УНИВЕРЗИТЕТ ПРИВРЕДНА АКАДЕМИЈА У НОВОМ САДУ
ФАКУЛТЕТ ЗА ПРИМЕЊЕНИ МЕНАЏМЕНТ, ЕКОНОМИЈУ И ФИНАНСИЈЕ, БЕОГРАД

ЗАВРШНИ РАД

РАЗВОЈ ДЕСКТОП АПЛИКАЦИЈЕ УПОТРЕБОМ ПЛАТФОРМЕ ЈАВАФХ

Ментор:

Проф. др Ивона Брајевић

Студент:

Огњен Томић, I004-02/2019

Београд, 2020. године

САДРЖАЈ

Увод 3
ЈАВА - КАРАКТЕРИСТИКЕ И КОНЦЕПТИ 4
1. Историја развоја Јаве 4
2. Кључне карактеристике и концепти Јаве 6
3. Јава програмски језик 9
4. Објектна оријентисаност Јаве 16
4.1. Класе, варијабле и метод 18
4.2. Типови података и оператори 21
5. Примери Јава програмирања 24
6. Практична примена апликације виртуелног здравственог картона 26
6.1. Шта је JavaFX 26
6.2. Шта је то Java scene builder? 29
6.3. Шта је то Icon Pichon? 30
6.4. Шта је то XAMPP? 30
7. ИТ ИСТРАЖИВАЊЕ 32
8. ПРОЈЕКАТ ВИРТУЕЛНОГ ЗДРАВСТВЕНОГ КАРТОНА 38
8.1. Кориснички ниво апликације 39
8.2. Административни мени апликације 41
8.3. Мени пацијента у апликацији 43
8.3.1. Додавање пацијената 43
8.3.2. Приказ пацијената 45
8.3.3 Брисање пацијента 45
8.4. Мени лекара у апликацији 47
8.4.1. Додавање лекара 47
8.4.2. Приказ лекара 49
8.4.3. Брисање лекара 49
8.5. Отварање картона 51
8.6. Заказивање прегледа 52
ЗАКЉУЧАК 54
Литература 55

Увод

Предмет писања рада представља израду JavaFX апликације у форми виртуелног здравственог картона. Мотивација за израду оваквог вида апликације јесте унапређење вештина за израду Јава апликација, као и могућност да апликација послужи као пројекат који ће аутору омогућити да поседује пројекат који може приложити при тражењу посла у струци. Зашто баш JavaFX? Аутор је одабрао баш тај алат за израду апликације из разлога што су узети у обзир сви аспекти на тржишту рада као и највећа предност коју JavaFX алат нуди, да приликом израде графичког приказа апликације алат омогућава потпуну аутоматизацију .css дизајнерског дела кода. Употреба апликације се односи на аспект предстваљања јавношћу приликом одбране дипломског рада. Могућност комерцијалне употребе треба искључити јер да би дошло до такве употребе апликацију треба усавршити до крајњих функционалности.

Јава програмски језик представља данас један од најпознатијих програмских језика, који се често користи у пракси услед специфичних карактеристика, функција и једноставности. У питању је објектно оријентисани програмски језик, који за разлику од других, односно процедуралних, није специфичан за одређен хардвер, већ се може применити на широком спектру уређаја. Услед наведеног, програмски језик Јава се карактерише већом безбедношћу, динамичношћу, а као кључне карактеристике јављају се и дистрибуираност, неутралност архитектуре, независна платформа, једноставност, лакоћа учења и савладавања, високе перформансе и тако даље. Јава своју историју везује за почетак деведесетих година прошлог века, када је овај програмски језик имао име ОАК, да би се 1995. године преименовао у данашњи назив. Већ 1996. године Јава је окарактерисана као велико достигнуће у свету програмирања, а интензивна популарност почиње расти са појавом интернета, будући да су поједини интернет претраживачи, најпре Netscape, интегрисали Јаву у свој систем.

Будући да је Јава независна од хардвера на којем се примењује, потребно је истаћи улогу Јава виртуелне машине, која представља апстрактни хардвер на којем се имплементира Јава програмски језик, а чија је улога да изворни програмски код преведе у бајткод путем компајлера и анализира и представи путем интерпретера. Када се користе програмски језици који нису објектно оријентисани, решење сваког проблема исказује се на основу бројева и знакова. У објектно оријентисаном програмирању, програмирање се и даље обавља бројевима и знацима, али програмер може да дефинише и друге врсте ентитета који су потребни за решавање проблема. У Јави је заправо све дефинисано објектима, а сам објекат може бити било шта. Поред објеката, као релевантни елементи Јава програмског језика истичу се класе, потклассе, варијабле, методи, типови података, оператори и друго. С обзиром на једноставаност и предности, Јава се врло често примењује у развоју апликација, софтвера и система, поготово оних на интернету. Један такав случај јесте и развој апликација које ће пружити подршку електронском здравственом картону, у оквиру целокупног здравственог информационог система. Управо је циљ овог рада израда и презентација апликације у оквиру IntelliJ окружења. Наведена апликација треба да обезбеди подршку раду здравственог електронског картона у смислу да омогући микро окружење које ће бити део здравственог информационог система. Технологије које ће се користити за развој аплиакције односе се на JavaFX, Java scene builder, Java UI design и Swing технологије.

Да би се испитао циљ, потребно је најпре извршити теоријски преглед Јава концепта, како би се спознале кључне карактеристике и елементи овог програмског језика, што је учињено у првом делу рада. У другом делу рада извршен је преглед програмског језика, односно развоја горе поменуте апликације. Након тога следи закључак рада.

ЈАВА - КАРАКТЕРИСТИКЕ И КОНЦЕПТИ

1. Историја развоја Јаве

Јава програмски језик један је од најпознатијих програмских језика данас, а његова историја је релативно новог датума и везује се последње деценије прошлог века. Пре свега, с обзиром да је Јава објектно оријентисани програмски језик, потребно је најпре указати на кључне развоје у овом домену. Наиме, идеја објектно оријентисаног програмирања није нова. Први програмски језик овог типа, познат као Simula, развијен је још у току шездесетих година двадесетог века од стране тројице скандинавских компјутерских стручњака. Иако су концепти Simula програмског језика доста коришћени, сам језик није имао значајну пажњу, осим у свету програмирања. С друге стране, програмски језик који је привукао већу пажњу јавности био је Smalltalk, програмски језик развијен од стране Xerox истраживачког центра. Међутим, највећи успех и комерцијализацију доживљава C програмски језик, а свакако његова верзија C++, која се увелико користи како у свету програмирања, тако и од стране аматерских програмера и шире јавности.[1]

Најновије остварење у свету објектно оријентисаног програмирања постигнуто је развојем Јава програмског језика. Јаву су првобитно дизајнирали 1991. године James Gosling, Patrick Naughton, Crish Warth, Ed Frank и Mike Sherdian из корпорације Sun Microsystem Inc. Аутори су радили 18 месеци на првој верзији програма, који је се до 1995. године није звао Јава, већ OAK. Али, од појаве иницијалне идеје до прве верзије, односно од 1992. до 1995. године, поред претходно наведених програмера, на развоју Јаве радили су и: Bill Joy, Arthur van Hoff, Jonathan Payne, Frenk Yellin и Tim Lindholm.[2] Сви наведени аутори били су познати као Green team, с обзиром да је сам пројекат развоја Јаве носио назив Green project, а сам циљ развоја Јаве био је да се креира језик који ће трошити мало меморије и радити на слабим процесорима. Иницијално, програм је био замишљен тако да се имплементира на малим технолошким уређајима, као што су телевизори, микроталасне пећи, даљински управљачи, сет топ бокс и други ситни уређаји у домаћинствима, али је касније дошло до имплементације Јаве на интернету и Netscape платформи.

Као што је претходно напоменуто, Јава се све до 1995. године звала OAK, а аутори су одабрали овај назив будући да OAK на енглеском означава храстово дрво, које репрезентује моћ и истрајност и симбол је јаких светских економских сила попут Сједињених Америчких Држава, Немачке, Канаде, Француске и слично. Међутим, аутори су 1995. године морали променити назив јер је већ постојало регистровано име OAK Technologies. Али, сада се поставља питање зашто су аутори као ново има одабрали "Јава"? Треба најпре напоменути да су међу поменутим именима били: dynamic, revolutionary, Silk, jolt, DNA и тако даље. Аутори су хтели назив који рефлектује технологију и означава динамизам, револуцију, живахност, јединственост, супериорност, а који се лако изговара. Поред назива Silk, у врху се нашло име Јава и многи су га преферирали. Ово име не представља акроним, већ представља острво у Индонезији и смислио га је James Gosling.[3]

Први програм у Јави звао се *7 и био је то програм за интелигентни даљински управљач. Међутим, велики значај Јава добија са почетком развоја интернета (World Wide Web). Већ 1994. године, двојица аутора, односно Jonathan Payne и Patrick Naughton развили су

[1] Roberts, E.S., 2006, *The art and ascience of Java*, Stanford University, California, стр. 14.
[2] Сарачевић, М., 2015, *Програмски језик Јава*, Универзитет у Новом Пазару, Нови Пазар, стр. 10.
[3] https://www.javatpoint.com/history-of-java, датум приступа 05.11.2020. године.

интернет претраживач који подржава Јаву на интернету и управо је ово био почетак периода интензивног развоја интернет сервиса. Следеће, односно 1995. године, чувени Netscape интернет претраживач у својем претраживачју инкорпорирао је Јава платформу.[4] Јава је 1996. године означена као једноставан, објектно оријентисани, робустни, једноставни, високо перформативни, динамични и архитектурно неутрални програмски језик. Основна карактеристика Јаве, која је управо омогућила велики успех овом програмском језику, јесте чињеница да он није развијен како би задржао специфични хардвер, већ се може применити на било којем систему. За разлику од других програмских језика, Јава преводи изворни програмски код у бајткод, који није машинско специфичан, већ посредује, односно представља интермедијатора између виртуелне и реалне машине.[5] Целокупна Јава филозофија била је замишљена на следећи начин:[6]

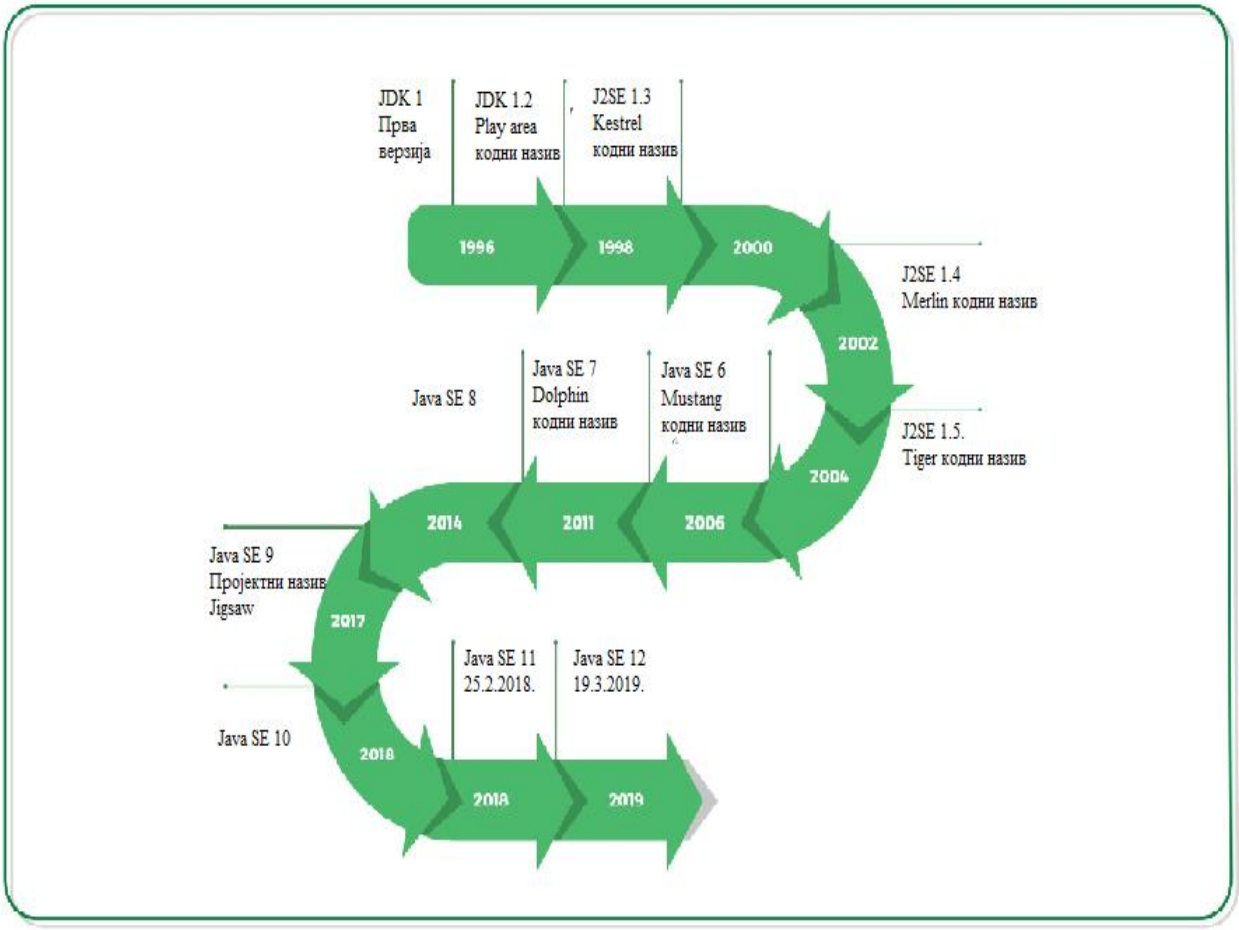
- програмски језик био је замишљен тако да користи објектно оријентисану методологију програмирања,
- циљ је да се развијени програм може користити на бројним оперативним системима и хардверима, а не само на једном, односно специфичном,
- програм се мора дизајнирати тако да пружа подршку интернет претраживачима,
- програм се мора дизајнирати тако да извршава код безбедно.

Уколико се посматра сам програмски језик, хронолошки развој Јаве може се посматрати на следећи начин:[7]

- прва верзија јаве (JDK) објављена је 23. јануара 1996. године, а потом следе наредне верзије:
    - JDK 1.1 - 19. фебруара 1997. године,
    - J2SE 1.2 - 8. децембар 1998. године,
    - J2SE 1.3 - 8. маја 2000. године,
    - J2SE 1.4 - 6. фебруара 2002. године,
    - J2SE 5.0 - 30. септембар 2004. године
    - Java SE 6 - 11. децембар 2006. године,
    - Java SE 7 - 28. јула 2011. године,
    - Java SE 8 - 8. марта 2014. године,
    - Java SE 9 - 21. септембар 2017. године,
    - Java SE 10 - 20. марта 2018. године.
    - Java SE 11 - 25. децембра 2018. године и
    - Java SE 12 - 19. марта 2019. године.

Данас се Јава успешно користи на Windows апликацијама и интернет претраживачима, компаманијски софтверима, мобилним апликацијама и тако даље. Детаљнија хронологија развоја приказана је на Слици 1, из које се може закључити да се кључни аспекти у развоју Јаве односе на 1996. годину, када је објављена прва верзија Јаве, а након тога се сукцесивно развија Јава са различитим кодним називима. Последња верзија Јаве појавила се 19. марта 2019. године.

[4] Sawitch, W., 2016, *Absolute Java 6th edition*, Pearson, Boston, стр. 35.
[5] Girau, L,, 2018, *Object oriented programming*, African Virtual University, Nairobi, стр. 19.
[6] https://www.freejavaguide.com/history.html, датум приступа 05.11.2020. године
[7] https://www.javatpoint.com/history-of-java, датум приступа 05.11.2020. године.



Слика 1. Хронологија развоја Јава програма

Извор: https://www.geeksforgeeks.org/the-complete-history-of-java-programming-language/, датум приступа 05.11.2020. године

2. Кључне карактеристике и концепти Јаве

Јава је програмски језик и платформа која је високо робустна, објектно оријентисана и безбедна. У питању је иновативни програмски језик који је постао незаобилазан када се ради о примени на различитим системима. Пре свега, Јава омогућава писање малих програма, такозваних аплета, који се могу уградити у интернет странице и тако обезбеде извесну функционалност. Поред тога, Јава омогућава писање великог броја програма који се неизмењени могу применити на различитим рачунарима, што ја данас најчешће и случај.

Најважнија карактеристика Јаве се односи на чињеницу да је она дизајнирана тако да је у самом старту машински независна, односно може се применити на различитим платформама. Апликација која је написана у Јави захтева само један скуп изворног кода, без обзира на колико се платформи жели применити, што је кључна карактеристика у односу на друге програмске језике. Овим путем се обезбеђује значајна уштеда времена, трошкова и других ресурса приликом развоја апликације.

Друга битна карактеристика се односи на то да је Јава програмски језик објектно оријентисан. Објектно оријентисани програми су разумљиви и лаки за одржавање, па се

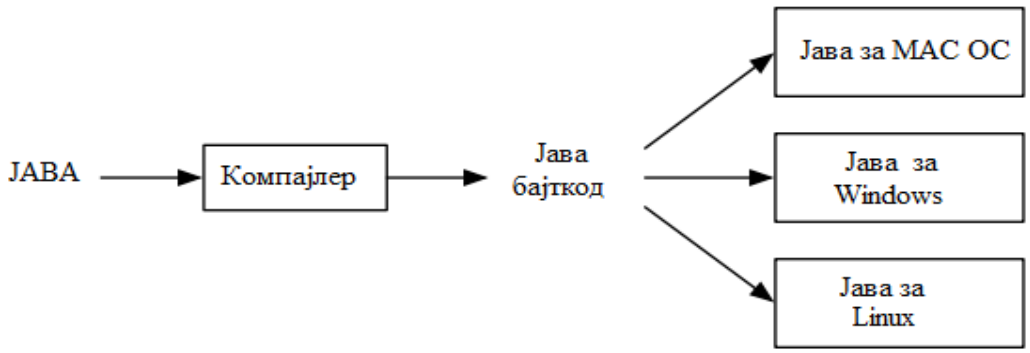
због тога Јава програмски језик лакше учи. Другим речима, у Јави не постоје замке у програмирању, што је карактеристично за друге програмске јзике. Коначно, битна карактеристика Јаве односи се и на постојање националних скупова знакова. То заправо значи да се у Јави програми лако пишу за коришћење у било којој земљи, без обзира да ли се у њима говори или не говори енглески језик.[8] Сходно томе, може се закључити да је Јава прилично моћан и компактан програмски језик. Као основне предности Јаве истичу се:[9]

- једноставност,
- дистрибуираност,
- објектна оријентисаност,
- робустност,
- безбедност,
- интерпретативност,
- динамичност,
- лакоћа учења и друго.

Јава поседује и извесних недостатака:[10]

- не постоји специфична применљивост на поједине хардвере,
- недостатак темплејта.

Јава програмски језик је статично писани језик, што значи да свака варијабла и сваки израз имају своје познато време компајлирања. Истовремено, сви типови програмских језика у Јави могу се поделити у две категорије: примитивну и референтну групу, о чему ће више речи бити касније. Сходно томе, програмери путем Јаве користе комбинацију компилације и интерпретације. Дакле, програмски језик се путем Јаве компајлира у машински језик, али заправо такав језик реално и не постоји, већ је познат као виртуелни рачунар, односно Јава виртуелна машина (Java Virtual Machine), чији је језик познат као Јава бајткод. Управо због тога Јава програми се могу применити на сваком систему, као што показује Слика 2.



Слика 2. Јава бајткод и оперативност

Извор:Eck, D.J., 2006, Introduction to programming using Java, Hobart and William Smith Colleges, Geneva, стр. 7.

[8] https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf, датум приступа 06.11.2020. године

[9] Ivanović, M., Budimac, Z., Mishev, A., Bothe, K., Jurca, L., 2013, *Java across different curricula, courses and countries using a common pool of teaching material*, Informatics in Education, 12(2), стр. 155.

[10] Ibid.

У Јави програми настају путем класа, а из дефиније класе може настати велики број објеката који су за дату класу карактеристични. Само програмирање може се извршити из било којег програма за писање текста (text editor). Генерално, Јавини програми могу се поделити у две групе:

- оне који се уграђују у интернет сајтове и који су познати као Јава аплети и
- независни Јава програми који се могу инсталирати и који су познати као Јава апликације.

Када је реч о другој групи, треба имати у виду да се Јава програми не извршавају директно на рачунару на којем су написани, већ путем стандардизованог окружења, познатог као Јава 2 Платформа, која има две компоненте:

1. Јава виртуелну машину и
2. Јава програмски интерфејс.

Процес функционише тако што Јава компајлер изворни код претвара у Јава бајткод, који се путем извршавања покреће путем Јава интерпретера који испитује и дешифрује код, проверава његову аутентичност и безбедност.[11]

Могу се издвојити следеће Јавине апликације:[12]

- стандардне апликације које се примењују на рачунарима,
- интернет (веб) апликације,
- апликације за предузећа,
- апликације за мобилне телефоне.

Тако написани програм мора поседовати следеће карактеристике:[13]

- мора бити једноставан,
- мора бити објектно оријентисан,
- мора бити познат,
- мора бити робустан,
- сигуран,
- мора поседовати независну архитектуру,
- мора бити портабилан,
- добијени програм мора имати високе перформансе,
- динамичност,
- интерпретативност и
- мора подржавати нити.

Заправо, целокупни сет карактеристика Јава програмског језика, онако како то представљају запослени Sun Microsystem корпорације, броји дванаест карактеристике, које су приказане на Слици 3.

[11] https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf, датум приступа 06.11.2020. године
[12] https://www.javatpoint.com/java-tutorial, датум приступа 06.11.2020. године
[13] Мићовић, Н., 2016, *Увод у програмски језик* Јава, SystemPro, Београд, стр. 4.

Слика 3. Карактеристике Јаве

Извор: https://www.javatpoint.com/features-of-java, приступљено 06.11.2020. године

3. Јава програмски језик

Пре него што се приступи анализи Јавиног програмског језика, потребно је најпре спознати саму идеју програма и термина повезаних са програмирањем. Компјутерски програм сет написаних инструкција које извршава рачунар уз помоћ одређеног хардвера и софтвера. Програм који за корисника извршава задатак представља апликативни софтвер, а програм који управља рачунаром познат је као системски софтвер. Сви рачунарски програми се конвертују у машински језик. Наведени машински језик, или машински код, представља сет инструкција које извршава рачунар и у питању је програмски језик нижег разреда (нивоа), који су веома комплексни за учење и рад и захтева специфичну машину за рад. Међутим, програмски језици су временом еволуирали и напредовали па је дошло до развоја вишеразредних програмских језика (вишег нивоа), који омогућавају креирање, читање, обраду, измену, односно лакши су за рад. Овакви језици имају своју синтаксу, односно сет правила како се програмски језици комбинују. У програмирања, програмери комбинују низ различитих програмских исказа, сличних реченицама, који треба да изврше одређен задатак. Такви програмски искази познати су и као команде, зато што наређују рачунару шта треба да уради. Након што се програмски искази напишу, програмери

користе специфичне програме, познате као компајлери и интерпретатори како би се написани језик превео у машински. Наведени компајлери и интерпретатори истичу грешку сваки пут када се јави проблем у раду програма, што је познато као синтаксна грешка (Syntax error).[14]

Поред објектно оријентисаног програмирања, које је карактеристично за Јаву, постоји и процедурално програмирање. Његова основна суштина огледа се у томе што се операције изводе једна након друге. Програмер у процедуралним апликацијама креира назив за сваку рачунарску меморијску локацију, које су познате као варијабле. Објектно оријентисано програмирање представља екстензију процедуралног програмирања, чије програмирање укључује:[15]

- креирање класа и објекта,
- креирање апликација које манипулишу класама,
- рачунарске симулације, који представљају модел реалног света у мини верзији,
- графички кориснички интерфејс (Graphical User Interface - GUI), који кориснику омогућавају управљање програмом, разумевање истог и манипулацију њиме.

У Јава програмирању користи се такозвани Java Development Kit - JDK, који ће касније бити описан у раду, који подразумева коришћење следећих алата:[16]

- прегледач аплета,
- Јава компајлер,
- Јава интерпретер,
- Јава бајткод,
- Javah (за C header фајлове),
- Javadoc (за HTML документа).

За писање програма путем Јаве, може се користити било који едитор текста, где се након писања фајл чува са .јава екстензијом. Путем Јава компајлера обавља се конверзија изворног кода програма у бајткод, а интерпретатор преводи из .јава фајла у .class фајл. За компајлирање Јава програма користи се следећа синтакса:[17]

C:\javac filename.java

Egg. If my filename is abc.java, then the syntax will be

C:\javac abc.java

Путем Јава интерпретатора, покретање програма подразуемева следећу синтаксу:[18]

C:\java filename

E.g. If my filename is abc.java then the syntax will be

C:\java abc

---

[14] Farrell, J., 2016, *Java programming 8th edition*, Cengage learning, Boston, стр. 2-3.
[15] Исто, стр. 6-7.
[16] Girau, L,, 2018, *Object oriented programming*, African Virtual University, Nairobi, стр. 16.
[17] Исто, стр. 17.
[18] Ibid.

Као што је претходно наведено, Јава програмирање је заправо дефиниција класе обављена путем main метода. Када се покрене програм, позива се main метода, односно main специфицирана акција се извршава. Основна садржина main метода написана је у заградама {}, па сваки пут када се програм покрене, извршавају се искази у заградама. Наредна линија истиче да је програмска класа названа FirstProgram (први програм):[19]

```
public class FirstProgram
{
```

Наредне две линије садрже дефиницију main метода:

```
public static void main(String[] args)
{
```

Уколико се као пример дијалога узме следеће:

Hello reader.

Welcome to Java.

Let's demonstrate a simple calculation.

2 plus 2 is 4

онда би Јава програм имао следећу форму:

```
1    public class FirstProgram
2   {
3    public static void main(String[] args)
4   {
5   System.out.println("Hello reader.");
6   System.out.println("Welcome to Java.");
7   System.out.println("Let's demonstrate a simple calculation.");
8   int answer;
9   answer = 2 + 2;
10  System.out.println("2 plus 2 is " + answer);
11  }
12  }
```

[19] Sawitch, W., 2016, *Absolute Java 6th edition*, Pearson, Boston, стр. 37-38.

У елементарном разумевању Јава програмског језика, углавном се полази од примера "Здраво свете" (Hello World). Да би се написао овај програм, који се можда чини једноставним, потребно је најпре:[20]

- прибавити програм за писање текста,
- компајлирати програм и
- покренути компајлирани програм.

Јава програм који каже "Hello World" приказан је у наставку текста, а његови основни елементи приказани су касније у раду:[21]

```
// A program to display the message
// "Hello World!" on standard output
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello World!");
}
} // end of class HelloWorld
```

Команда која представља поруку има следећу линију:

```
System.out.println("Hello World!");
```

Поред апликација, у Јави се могу писати и аплети. У питању су динамички интерактивни програми који се могу извршити у оквиру интернет странице путем одређеног интернет претраживача. Међутим, аплет се не извршава на исти начин као и апликација. Да би могао да се покрене, аплет мора бити уграђен у веб страну, а покреће се претраживачем. Уколико се, на пример, аплет покреже помоћу appletviewer претраживача, а компајлирани аплет је сачуван као MyApplet.html, онда се аплет покреће на следећи начин:[22]

```
appletviewer MyApplet.html
```

Узимајући у обзир претходно употребљени пример, додавање аплета у HTML документ обавља се на следећи начин:[23]

```
<html>
 <head>
```

[20] Eck, D.J., 2006, *Introduction to programming using Java*, Hobart and William Smith Colleges, Geneva, стр. 20
[21] Исто, стр. 21.
[22] https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf, датум приступа 06.11.2020. године
[23] Ibid.

```
 <title> A Simple Program </title>
</head>
<body>
 <hr/>
 <applet code = “MyFirstApplet.class” width = 300 height = 200 >
 </applet>
 <hr/>
</body>
</html>
```

Означеним линијама истакнут је бајткод у оквиру аплета, а име датотеке која садржи бајкод наводи се као вредност атрибута code контролног кога <applet>. Атрибути width и height одређују ширину и висину дела дисплеја који ће аплет користити у току свог извршења и ови атрибути се означавају за сваки аплет. Треба имати у виду да Јава прави разлику између великог и малог слова. Па тако уколико се реч public напише великим словом P, програм се неће компајлирати.

Потребно је такође и спознати кључне карактеристике Јав програмирања. Пре свега, Јава се може применити на мноштву рачунара, али се не извршава директно на њима, већ на хипотетичком рачунару познатом као Јава виртуелна машина. Програмски искази написани у високо разредном програмском језику познати су као изворни код (source code). Када се пише Јава програм, прво се конструише изворни код путем одређеног едитора текста и едитора изворног кода познатог као jGRASP. Развојно окружење јаве представља сет алата који пружају помоћ и одређена средства у процесу програмирања. Као што је претходно наведено, када се сачува фајл, путем компајлера се он преводи у Јава изворни код, а потом Јава интерпретер проверава бајкод и комуницира га са оперативним системом. Пошто је Јава издвојен, односно изолован од рачунара и хардвера на којем се примењује, овај програмски језик одликује одређен систем безбедности у степену већем него код других језика. Поред тога, Јава је доста једноставнија од других програмских језика, као што је на пример C++.[24]

Као што је претходно наглашено, Јава програмски језик је високо робустан. У процесу програмирања, ово заправо значи да Јава подразумева прецизно управљање меморијом, избегавају се проблеми са безбедношћу, аутоматски се прикупља смеће путем Јавине виртуелне машине, где се подразумева прикупљање и брисање објеката који се не користе у Јава апликацији. Јава програмски језик је знатно бржи од других, што му омогућава додатне перформансе у односу на пример на C и C++ програмски језик.[25] Када је реч о поређењу ова два језика, у Табели 1 приказане су кључне разлике Јава и C++ програмског језика.

[24] Farrell, J., 2016, *Java programming 8th edition*, Cengage learning, Boston, стр. 11.
[25] https://www.javatpoint.com/features-of-java, датум приступа 07.11.2020. године

Табела 1. Разлике између C++ и Јава програмског језика

Критеријум поређења	C++	Јава
Платформа	Завистан од платформе	Независтан од платформе
Где се најчешће користи	Код системског програмирања	Код великог броја уређаја
Циљ дизајна	Дизајниран је за системе и апликације програмирања	Олакшано коришћење уређаја од стране корисника
Показивачи	Подржани	Подржани, али се не могу написати у Јави
Компајлери и интерпретери	Користе се само компајлери.	Подржана су оба
Коментари докумената	Нису подржани	Подржан је коментар (/** ... */) како би се креирао документ за Јава изворни код
Виртуелна кључна реч	Подржана	Није подржана
Хардвер	Близак је хардверу	Није интерактиван са хардвером
Објектна оријентисаност	Постоји, али single root није могућа	Потпуна објектна оријентисаност

Извор: https://www.javatpoint.com/cpp-vs-java, датум приступа 07.11.2020. године

У програмирању путем Јава програмског језика, треба истаћи карактеристике и улогу више пута поменуте Јава виртуелне машине. Као што је претходно речено, Јава виртуелна машина представља апстрактни појам, односно замишљени (хипотетички) рачунар који креира окружење за извршење Јава бајткода. Улога Јава виртуелне машине огледа се у следећем:[26]

- учитавање кода,
- верификација кода,
- извршење кода и
- одређује време извођења.

Посебно је значајна улога учитавања класа (Classloader). Кад год се неки програм покреће, подразумева се најпре учитавање класних фајлова. Битни елементи који се јављају код учитавања класа су следећи:

- окружење класе (Class area) - где се чувају класне структуре,
- Heap - окружење података где се чувају објекти,
- Stack - место чувања фрејмова,
- Program Counter Register - садржи адресу инструкције Јава виртуелне машине која се тренутно извршава,
- Native Method Stack - представља употребљени метод у апликацији,
- машину извршења (Execution Engine) - представља виртуелни процесор, интерпретер и компајлер који се извршава у реалном моменту, који треба да унапреди перформансе,
- Java Native Interface - оквир који пружа интерфејс за комуникацију са другом апликацијом написаном на другом језику, попут C++.

[26] https://www.javatpoint.com/jvm-java-virtual-machine, датум приступа 07.11.2020. године

JavaFX представља библиотеку која се користи за развој десктоп апликација, као и богатих интернет апликација (RIA - Rich Internet Applications), а која се може применити на многобројним декстоп рачунарима, мобилним телефонима и веб технологијама.

Као основне карактеристике JavaFX истичу се:[27]

- Јава библиотека, која садржи многоштво класа и интерфејса.
- FXML - XML базирани декларативни језик.
- Scene Builder - генерише XML на који се може пренети IDE.
- веб преглед,
- уграђене UI компоненте,
- Canvas API - омогућава директно цртање у окружењу JavaFX сцене,
- интегрисана графичка библиотека,
- високо пеформативни медијски механизам.

Графички кориснички интерфејс сваке десктоп апликације подразумева такозване UI елементе, који кориснику омогућавају интеракцију и размену информација. Класе које у оквиру UI компоненте омогућавају контролу су следеће:[28]

- Label (ознака) - компоненте којом се дефинише текст на екрану.
- Button - контролише функцију апликације.
- RadioButton - омогућава вишеструке опције кориснику.
- TextField - служи за унос текста, а у Јава програмском језику представљено је у виду JavaFX.scene.control.TextField.
- PasswordField - служи за унос корисничке лозинке.
- HyperLink - упућује на интернет локацију и представљено је класом JavaFX.scene.control.HyperLink.
- Slider - пружа увид у опције кориснику у графичкој форми.
- ProgressBar - показује напредак процеса кориснику, представљено је класом JavaFX.scene.control.ProgressBar.
- мени,
- ToolTip - информације кориснику у погледу компоненти.

Јава Swing се користи за апликације засноване на прозорима (window-based applications). Swing платформа је такође независна и омогућава компоненте попут:

- табела,
- листа,
- одабир боја
- дизајнирање табова,
- дизајнирање опције за скрловање и тако даље.

Неки од метода Јава Swing су следећи:[29]

- public void add(Component c) - додавање компоненте другој компоненти,
- public void setSize(int width,int height) - одређивање величине компоненте,
- public void setLayout(LayoutManager m) - уређивање изгледа,
- public void setVisible(boolean b) - видљивост компоненте.

[27] https://www.javatpoint.com/javafx-tutorial, датум приступа 09.11.2020. године
[28] https://www.javatpoint.com/javafx-ui-controls, датум приступа 09.11.2020. године
[29] https://www.javatpoint.com/java-swing, датум приступа 09.11.2020. године

4. Објектна оријентисаност Јаве

Објектно оријентисано програмирање јесте облик програмирања, односно приступ истом, где се све компјутације обављају у контексту објеката, па се сходно томе програм креиран на основу објектно оријентисаног програмирања састоји од скупа објеката, међусобно повезаних како би се обавио одређен задатак. Овакав поступак програмирања разликује се од традиционалног, где је програмски код подразумевао значајну меморију како би се учитао програмски код, која често није била довољна. Због тога се један део података морао чувати на другој меморији, па је приликом покретања програма било потребно користити неколико меморијских јединица, чиме је укупан процес текао знатно споро. Додатно, од програмера се захтевало да поседује знање о имплементацији и структури фајлова, па се стварао проблем одржавања програма. Објектно оријентисано програмирање елиминише овај проблем, кореспондирајући са објектима реалног света, који се налазе у интеракцији како би се испунио задатак.[30]

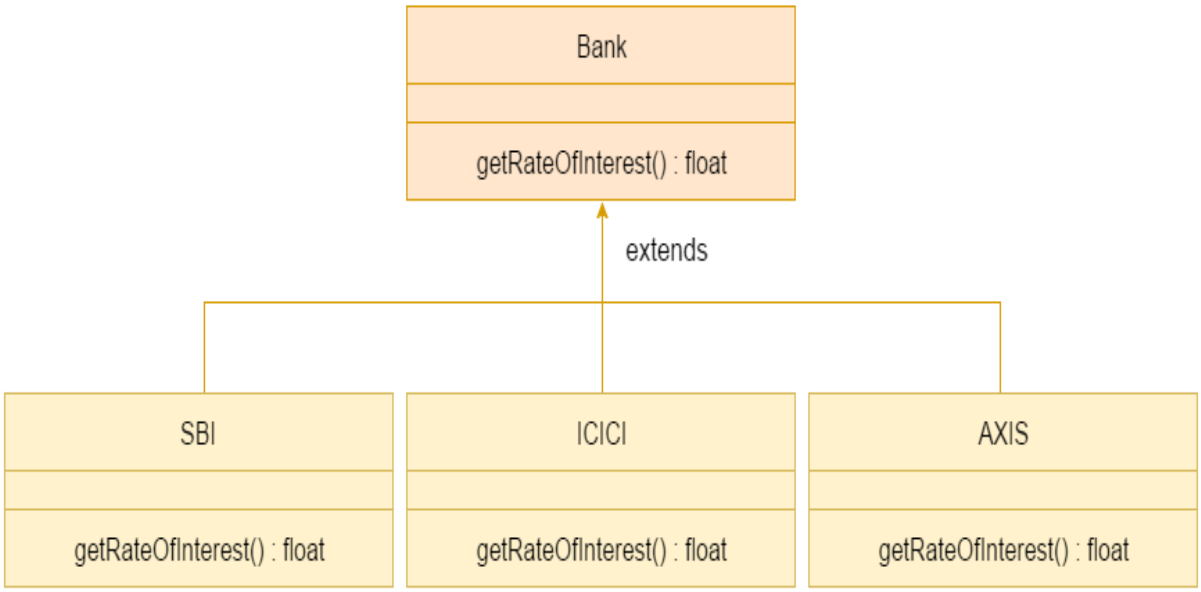
Када се користе програмски језици који нису објектно оријентисани, решење сваког проблема исказује се на основу бројева и знакова. У објектно оријентисаном програмирању, програмирање се и даље обавља бројевима и знацима - основним типовима података, али програмер може да дефинише и друге врсте ентитета који су потребни за решавање проблема. У Јави је заправо све дефинисано објектима, а сам објекат може бити било шта. У том погледу треба правити разлику између класе и потклase објекта. Наиме, класа објекта може представљати неки ентитет у целини. Нека се као пример узме дрво. Потклаcа објекта представља специфичну форму дрвета, на пример храст. Дакле, класа је спецификација у форми сегмента програмског кода која дефинише каква је структура одређене врсте објекта. Потклаcа је класа која наслеђује сва својства родитељске класе, са одређеним посебностима. Осим параметара који карактеришу објекат, путем класе се прецизира шта се може урадити са објектом, односно одређују се операције које се могу извести са објектима класе. Операције се одређују на основу такозваног метода. Сакривање података и метода унутар објекта познато је као енкапсулирање, које се постиже тако што се у дефиницији класе наведе уз сваку кључну реч private.[31]

Објектно оријентисано програмирање започиње идентификацијом објеката који се желе унети у проблем и порука на које ће се објекти односити. Објекти успостављају међусобну интеракцију и започињу међусобно слање порука, али различити објекти могу различито реаговати на поруке. Ово својство објекта познато је као полиморфизам. Сам термин полиморфизам грчког је порекла и састоји се од две речи: poly, што значи више - много и morphs, што значи форма. У Јава програмском језику и објектно оријентисаном програмирању, полиморфизам подразумева да се једна акција може извести на више различитих начина. Специфична форма полиморфизма у Јави позната је као Динамички метод отпремања (Dynamic Method Dispatch), који подразумева се примењен метод у процесу програмирања решава, односно замењује у току процеса извршавања, а не компајлирања. Наведено се извршава путем референтне варијабле суперкласе, а одабрани метод ће зависити од типа објекта на који упућује референтна варијабла. Нека се ради лакшег разумевања посматра пример три банке: SBI, ICICI, AXIS, које желе да одаберу метод обрачуна каматне стопе, које се међусобно разликују између банака: 8.4%, 7.3%, and 9.7% (Слика 4).

---

[30] Girau, L,, 2018, *Object oriented programming*, African Virtual University, Nairobi, стр. 14-15.

[31] https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf, датум приступа 06.11.2020. године



Слика 4. Полиморфизам на примеру банака и каматних стопа

Извор: https://www.javatpoint.com/runtime-polymorphism-in-java, датум приступа 08.11.2020. године

```
class Bank{
float getRateOfInterest(){return 0;}
}
class SBI extends Bank{
float getRateOfInterest(){return 8.4f;}
}
class ICICI extends Bank{
float getRateOfInterest(){return 7.3f;}
}
class AXIS extends Bank{
float getRateOfInterest(){return 9.7f;}
}
class TestPolymorphism{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
```

```
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Као резултат, односно output добија се:

```
SBI Rate of Interest: 8.4
ICICI Rate of Interest: 7.3
AXIS Rate of Interest: 9.7
```

Као карактеристике објектно оријентисаног програмирања треба навести:[32]

1. све је објекат - у процесу програмирања може се употребити било која концептуална форма објекта која је потребна да би се решио проблем,
2. програм се састоји од великог броја објеката који међусобно комуницирају, размењују поруке и говоре један другом шта треба урадити,
3. сваки објекат има своју меморију,
4. сваки објекат има свој тип,
5. сви објекти специфичног типа могу примити идентичну поруку.

Објекти у Јави имају своје стање, односно форму која репрезентује вредност објекта, као и специфично понашање и јединствени идентитет. Али поред тога, потребно је спознати и неке друге саставне делове објектно оријентисаног програмирања у Јави.

4.1. Класе, варијабле и метод

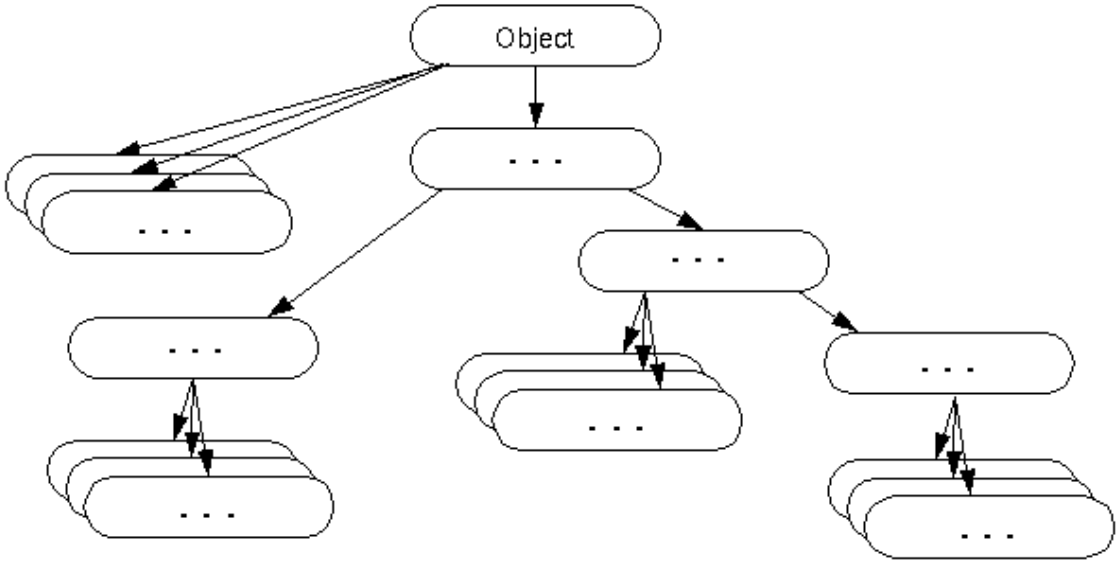
Као што је претходно поменуто, класа представља скуп објеката са идентичним карактеристикама. Као таква, класа у Јави садржи:[33]

- поља,
- методе,
- контрукторе,
- блокове,
- класу унутар класе (Nested class) и интерфејс.

Уколико се крене од претходно поменутог примера дрвета, онда би у том случају дрво представљају класу објекта, док би потклаcа била нека специфична форма дрвета, односно клас, чија је истовремено надкласа објекат дрво. Дакле, потклаcа наслеђује све карактеристике класе којој припада.

[32] Eckel, B., 2000, *Thinking in Java, 2nd edition*, Prentice Hall, New Jersey, стр. 3132.
[33] https://www.javatpoint.com/object-and-class-in-java, датум приступа 08.11.2020. године

Класе и потклаасе се даље делити на бројне потклаасе, као што то показује Слика 5.



Слика 5. Класе и потклаасе

Извор: https://www.javatpoint.com/object-class, датум приступа 08.11.2020. године

Имајући у виду наведену карактеристику, односно могућност да се класе даље деле на низ потклаасе, потребно је указати на одређене методе класа објеката и могућности које пружају (Табела 2).

Табела 2. Методе класа

Метод	Опис метода
public final Class getClass()	Враћа класу класу објекта специфичног објекта
public int hashCode()	Враћа hashcode објекта
public boolean equals(Object obj)	Пореди објекат са специфичним објектом
protected Object clone() throws CloneNotSupportedException	Клонира објекат
public String toString()	Враћа низ објекта
public final void notify()	Активира тему и чека мониторинг објекта
public final void wait(long timeout)throws InterruptedException	Активира одређену тему и привремено је зауставља док се не покрене друга тема
protected void finalize()throws Throwable	Активира сакупљање отпада

Извор: https://www.javatpoint.com/object-class, датум приступа 08.11.2020. године

Пакети класа који се најчешће користе су:[34]

- java.lang - ове класе подржавају основне језичке функције, руковање низовима и знаковима низова.
- java.io - класе за улазне и излазне операције са подацима,

[34] https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf, датум приступа 06.11.2020. године

- java.util - ручне помоћне класе
- javax.swing - компоненте за прављење графичких корисничких интерфејса,
- java.awt - оригиналне компоненте за графички кориснички интерфејс
- java.awa.geom - дводимензионални геометријски облици,
- java.ewt.event - класе овог пакета користе се у имплементацији апликација са прозорима за руковање догађајима у оквиру наших програма

Програми манипулишу подацима који се налазе у меморији. У машинском језику, подаци приказују нумеричку адресу локације где се они налазе. У високо програмском језику као што је Јава, уместо бројева користе се називи, односно имена, како би се означили подаци. Овакво име у Јава програском језику познато је као варијабла. При томе, варијаблу не треба схватити као назив одређеног податка, већ на локацију (фолдер, контејнер) где се тај податак налази. Другим речима, варијабла директно упућује на локацију, а индиректно на податак. У Јава програмском језику, варијабли, односно месту где се налази податак, се може приступити једино путем приписане изјаве (assigmnment statement), која има следећу форму:

(variable) = (expression) ;

где се (expression) односи на било који податак у рачунару. Када рачунар добије задатак, он обавља expression евалуацију и добијени податак претвара у варијаблу. На пример, нека се посматра пример камате (rate):

rate = 0,07 ;

варијабла се у овом случају односи rate, односно камату, а (expression) је каматни број 0,07. Рачунар овај задатак изводи тако што број 0,07 додаје камати (rate).[35]

У циљу бољег разумевања објеката, класе и метода, нека се посматра пример два купца. Нека се као атрибути објекта наведу назив, адреса и буџет два купца X и Y. Висина буџета два купца се разликује и та вредност указује на вредност атрибута. Нека купац X има просечан буџет од 2000 долара, купац Y вредност буџета од 1000 долара. Оба купца примењују понашање типично за купце, који се може окарактерисати као метод. У овом случају, метод је куповина purchase (). Посматрајући два купца из перспективе програмског језика, наведено се може представити на следећи начин:

Купац X као објект:

```
Attributes:
  name =  X
  address =  1, Robinson Road
  budget =  2000
Methods:
  purchase()   {send a purchase request to a salesperson}
  getBudget()  {return budget}
```

[35] Eck, D.J., 2006, *Introduction to programming using Java*, Hobart and William Smith Colleges, Geneva, стр. 24.

Купац Y као објект

Attributes:

name = Y

budget = 1000

Methods:

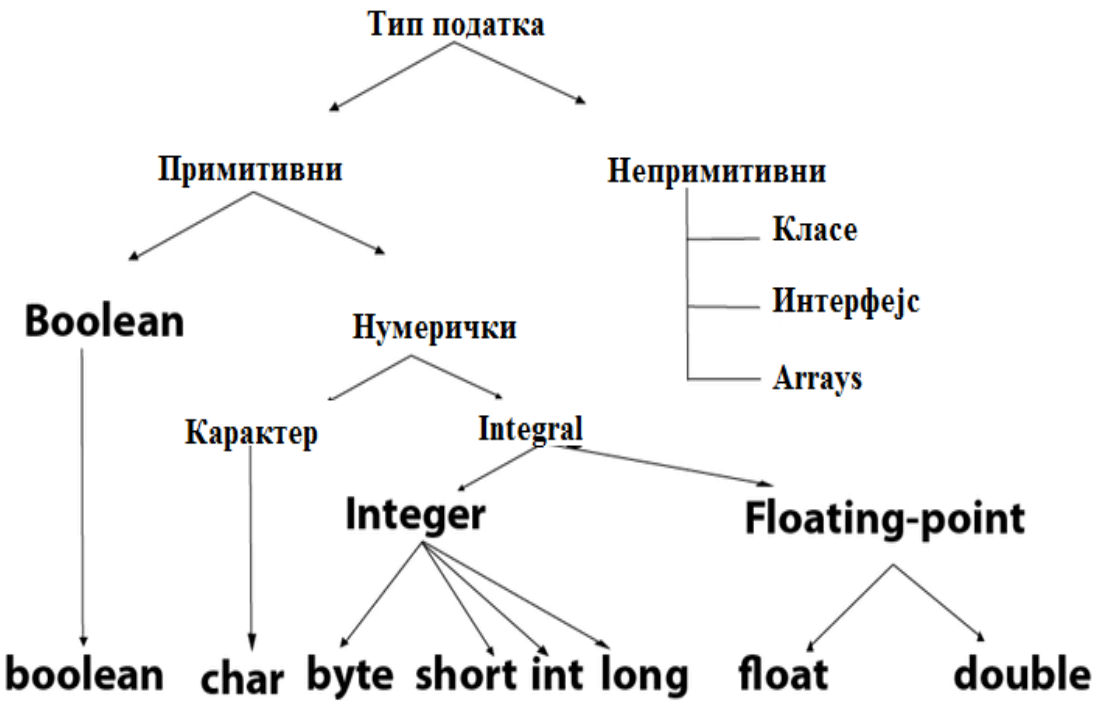
purchase() {send a purchase request to a salesperson}

getBudget() {return budget}

У наведеним линијама, name, address и budget представљају атрибуте, док purchase() и getBudget() представљају методе два објекта.[36]

4.2. Типови података и оператори

У Јави постоје две основне врсте типова података: примитивни и непримитивни типови (Класе, Интерфејс, Arrays) (Слика 4).



Слика 4. Јава типови података

Извор: https://www.javatpoint.com/java-data-types, датум приступа 08.11.2020. године

Када је реч о примитивним типовима, у Јави постоје више примитивних типова, при чему се под примитивним мисли на то да нису у питању класе већ општи подаци. У Јави

[36] Poo, D., Kiong, D., Ashok, S., 2008, *Object oriented programming and Java*, Springer, Berlin, стр. 7-8.

постоји осам примитивних типова података: byte, short, int, long, float, double, char и boolean (Табела 3).[37]

Табела 3. Примитивни типови података

Име података	Опис	Број битова	Подразумевана вредност
Byte	Обичан бајт	8	0
Short	Цели број	16	0
Int	Цели број	32	0
Long	Цели број	64	0
Float	Реални број	32	00f
Double	Реални број	64	00d
Char	Unicode карактер	16	'\u0000'
Boolean	Логичка вредност	1	false

Извор: Мићовић, Н., 2016, Увод у програмски језик Јава, SystemPro, Београд, стр. 5.

Као што се може приметити, прва четири примитивна типа садрже целе бројеве, float и double садрже целе бројеве, char садржи Unicode карактер, а boolean садржи логичку вредност true or false.

Јава програмски језик је објектно оријентисан, тако да подржава различите класе у оквиру непримитивних типова података. У том погледу треба најпре издвојити полиморфизам и енкапсулацију, који су претходно представљени у тексту. Као значани непримитивни подаци истичу се и String и Array. String представља секвенцу карактера, понаша се као класни тип и њиме се дозвољавају одређене наредбе, што није случај са класама и објектима. Array представља колекцију варијабли специфичног типа података који су организовани у контејнер којим се може управљати. Array се креира коришћењем једног основног типа податка, класе Јава библиотеке (Java library class), или посебне класе одређене од стране корисника - програмера. Уколико програмер жели у Јави да користи Array, потребно је унети заграде типу података у декларацији променљиве, што има следећи облик:[38]

```
int[] highScoreList;
```

Али, да би се претходно реализовало, потребно је спровести два корака. Прво, потребно је извршити дефинисање Array, а потом се мора алоцирати меморија:

```
int[] studentGrades;

studentGrades = new int[30];
```

Оператори се у Јави користе у виду симбола како би се извршиле одређене операције. Врсте Јава оператора приказане су у Табели 4.

[37] Eck, D.J., 2006, *Introduction to programming using Java,* Hobart and William Smith Colleges, Geneva, стр. 41.
[38] Harbour, J.S., 2012, *Begining Java: SE 6 game programming, 3rd edition*, Cengage learning, Boston, стр. 53-54.

Табела 4. Јава оператори

Тип оператора	Категорија	Симбол
Унарни	postfix	expr++ expr--
	prefix	++expr --expr +expr -expr ~ !
Аритметички	мултипликативни	* / %
	адитивни	+ / -
Shift	shift	<< >> >>>
Релациони	поређење	< > <= >= instanceof
	Једнакост	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Логички	logical AND	&&
	logical OR	||
Ternary	Ternary	? :
Assignment	Assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Извор: https://www.javatpoint.com/operators-in-java, датум приступа 09.11.2020. године

У наставку рада извршен је опис претходно поменутих оператора:[39]

- Унарни оператор повећава или смањује вредност за један, негира изразе и инвертује логичку вредност.
- Аритметички оператори омогућавају сабирање, одузимање, множење и дељење.
- Shift оператор помера све битове вредности на леву страну или десну (<< >>) страну одређено броја пута.
- Логички оператор && не проверава други услов уколико је први услов погрешан, односно врши проверу само ако је први услов тачан.
- Bitwise оператор & увек проверава услове, без обзра на тачност или нетачност првог.
- Ternary се користи као замена линије за изјаву if-then-else и користи се значајно у Јава програмирању. Једини је условни оператор који узима три операнда.
- Assignment оператор је један од најчешћих оператора у Јава програмирању. Користи се за додељивање вредности с десне стране операнду са леве стране.

[39] https://www.javatpoint.com/operators-in-java, датум приступа 09.11.2020. године

5. Примери Јава програмирања

Пример 1: Калкулатор

Путем Јава програмског језика може се развити калкулатор са базичним рачунарским операцијама: сабирање, одузимање, множење и дељење.

```
import java.util.Scanner;

public class Calculator {

public static void main(String[] args) {

Scanner reader = new Scanner(System.in);

System.out.print("Enter two numbers: ");

// nextDouble() reads the next double from the keyboard

double first = reader.nextDouble();

double second = reader.nextDouble();

System.out.print("Enter an operator (+, -, *, /): ");

char operator = reader.next().charAt(0);

double result;

//switch case for each of the operations

switch(operator)

{

case '+':

result = first + second;

break;

case '-':

result = first - second;

break;

case '*':

result = first * second;

break;

case '/':

result = first / second;

break;
```

```
// operator doesn't match any case constant (+, -, *, /)
default:
System.out.printf("Error! operator is not correct");
return;
}
//printing the result of the operations
System.out.printf("%.1f %c %.1f = %.1f", first, operator, second, result);
}
}
```

Када се покрене програм, то изгледа овако:

1. Enter two numbers: 20 98
2. Enter an operator (+, -, *, /): /
3. 20.0 / 98.0 = 0.2

Пример 2. Факторијел

Факторијел неког броја је производ свих позитивних бројева који су мањи или једнаки том броју. Писање програма у Јави има следећу форму:

```
import java.util.Scanner;
public class Factorial {
public static void main(String args[]){
//Scanner object for capturing the user input
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the number:");
//Stored the entered value in variable
int num = scanner.nextInt();
//Called the user defined function fact
int factorial = fact(num);
System.out.println("Factorial of entered number is: "+factorial);
}
```

```
static int fact(int n)
{
int output;
if(n==1){
return 1;
}
//Recursion: Function calling itself!!
output = fact(n-1)* n;
return output;
}
}
```

Извршење програма даје следећу поруку:

1. Enter the number:
2. 12
3. Factorial of entered number is: 47900160

6. Практична примена апликације виртуелног здравственог картона

У другом делу завршног рада биће представљен пројекат: „Виртуелног здравственог картона“. Пројекат је рађен у Јава програмском језику, користећи JavaFX технологијом. Додатни ппрограми који су коришћени у развоју примера су :

1. JavaFX
2. Java Scene builder
3. Icon Pichon
4. Xampp

6.1. Шта је JavaFX

JavaFX представља софтверску платформу за креирање апликација, на начин да програмерима пружа графичку платформу високих перформанси, а основна намера је да нове апликације користе JavaFX уместо Swing платформе како би се направио нови графички кориснички интерфејс, при чему то не значи да се Swing више не примењује. Карактеристике JavaFX су следеће:[40]

[40] https://www.javatpoint.com/javafx-tutorial, датум приступа 10.12.2020. године

- Јава библиотека (Java library) - састоји се од класа и интерфејса,
- FXML - XML декларативни језик,
- претходно поменути Scene builder,
- веб прегледач,
- UI контроле,
- Swing интероперабилност,
- интегрисана графичка библиотека,
- високо перформативни медијски систем,
- Canvas API - омогућава директно цртање.

Поред ЈавaФХ верзије 1.0 постоје још и следеће верзије:

- ЈавaФХ 1.0 (4. децембра 2008. године)
- ЈавaФХ 1.1 (12. фебруара 2009. године)
- ЈавaФХ 1.2 (2. јуна 2009. године)
- ЈавaФХ 1.3 (22. априла 2010. године)
- ЈавaФХ 1.3.1 (21. августа 2010. године)
- ЈавaФХ 2.0 (10. октобра 2011. године)
- ЈавaФХ 2.1 (27. априла 2012. године)
- ЈавaФХ 2.2 (14. августа 2012. године)
- ЈавaФХ 8 (. године)
- ЈавaФХ 9 (. године)

У првим верзијама ЈавaФХ (1.0, 1.1, 1.2, 1.3 i 1.3.1) се ослањао на једноставнији графички кориснички интерфејс али оно што је била основна њихова предност у односу на друге технологије јесте могућност да примене на скоро све технологије везане за:

- Десктоп
- Мобилне апликације
- као и Web.

ЈавaФХ се у самом почетку користио у главном за опширне интернет апликације популарно зване (РИА). У самој основи ЈавaФх је био структуиран и осмишљен као скриптни језик који је био најчешће коришћен за израду интернет апликација. Током почетка развоја ЈавaФХ платформе било је подељено мишљење међу искусним програмерима, да ли ће се ЈавaФХ потпуно заменити Свинг технологију или не. У тренутку када је Oracle показао иницитијативу за преузимањем Јава програмског језика од компаније Sun, сви ресурси су се преусмерили на унапређење ЈавaФХ алата и на крају његову потпуну имплементацију у Јава програмски језик са сироким дијапазоном могућности у развоју десктоп апликација[41].

Карактеристике које су издвајале ЈавaФХ у односу на Swing технологију су:

- Унапређење синатаксе у ондосу на претходну верзију ЈавaФХ,
- Убрзање брзине и квалитета графике,
- Имплементација помоћног ФХМЛ кода који омогућава корисницима боље управљање корисничким интерфејсом,
- Убрзање обраде приликом израде Web апликација и,
- Уградња Web компонената унутара ЈавaФХ апликације.

[41] https://bs.eferrit.com/sta-je-javafx/, приступљено: 12.12.2020 године.

Када је реч о структури ЈаваФХ апликације она се може поделити на:

- Главну страну (primaryStage) и,
- Чворове.

Главна страна (primaryStage) увек врши функцију наслеђивања класе Application и увек примењује методу *START*. Метода START прихвата један параметар типа *primaryStage*. Функције и класе које су поменуте у тексту се налазе у главној класи (Main). Почетни параметри које можете видети у коду који се налази испод слике јесте да је увек primaryStage.setTitle постављен на : "Hello World".

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root =
FXMLLoader.load(getClass().getResource("administrator.fxml"));
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
```

Следећа ставка која је потребна јесте дефинисање вредности везане за висину и ширину корисничког интерфејса. На слици кода који се налази изнад текста можете видети линију кода која поставља почетне вредности корисничког интерфејса на (300,275). Цела линија кода је : "primaryStage.setScene(new Scene(root, 300, 275));".

Креирање самог корисничког интерфејса каквог ЈаваФХ познаје не би било уопсте могуће без организационих класа панела који се програмира. Класе које су коришћене у пројекту су[42]:

- AnchorPane
- Label
- DatePicker
- ImageView
- Button
- ComboBox
- ScrollPane
- TextField
- GridPane(3,3)

Класа AnchorPane представља главно поље у коме се креира графички приказ апликације. Омогућава и обезбеђује потпуну слободу дизајнеру за употребу свих осталих класа како би досао до жељеног изгледа апликације.

[42] https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/package-frame.html, приступљено: 12.12. 2020 године

Класа GridPane омогућава дизајнеру апликације да постави као у случају апликације виртуелног здравственог картона панел са пољима која имају по 3 врсте и 3 колоне.

Класа Label омогућава дизајнеру апликације да у њу постави било какав вид садрзаја. Label опција најчешће слузи као дескриптивни приказ функције.

Класа DatePicker омогућава дизајнеру да постави у простор поље које ће кориснику апликације омогућити да одабере неки жељени датум. У апликацији виртуелног здравственог картона може се приметити у делу заказивања прегледа.

Класа ImageView омогућава дизајнеру апликације да у поље задатих димензија постави жељену слику, која ће унапредити визуелни изглед апликације.

Класа Button омогућава дизајнеру апликације да постави графички приказ дугмета како би омогућио кориснику апликације да у случају апликације виртуелног здравственог картона потврђивање датума прегледа.

Класа ComboBox омогућава дизајнеру апликације најчешће комбиновање поменутих опција корисник одабере неке од жељених информација из падајућег мени-а.

Класа ScrollPane омогућава дизајнеру апликације најчешће приказе кориснику апликације нпр. листу постојећих лекара у систему.

Класа TextField је класа која је јако слична класи Label само што се најчешће класа TextFiled поставља када се кориснику омогућава да у неко поље упише одређени текст, као нпр. своје име и презиме.

6.2. Шта је то Java scene builder?

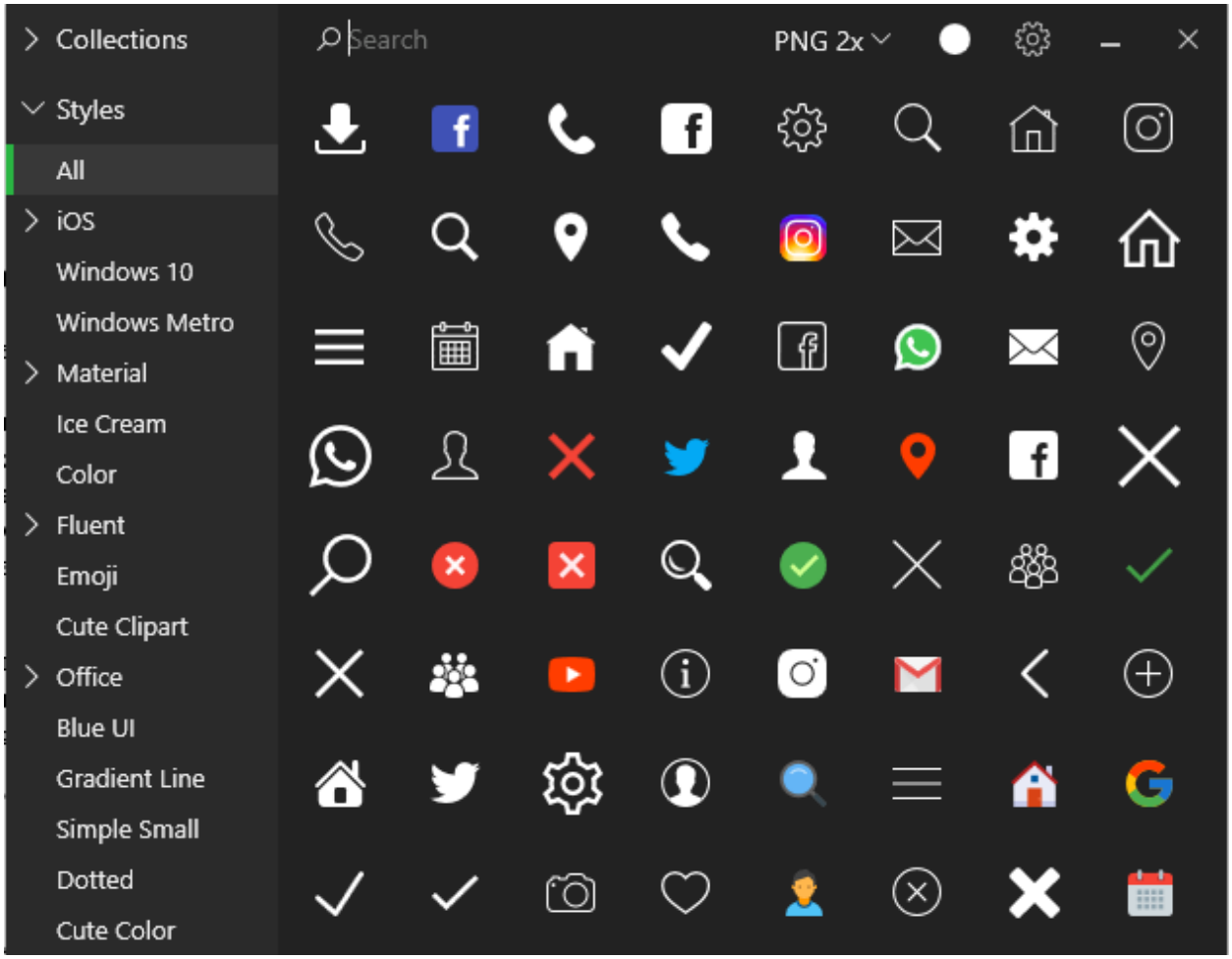
Java scene builder је алат за визуелни распоред који корисницима омогућава брзо дизајнирање корисничког интерфејса у јава апликацијама, без додатног кодирања. Корисници могу користити различите компоненте корисничког интерфејса на радно подручје које их занима и које им је потребно, изменити своја својства, применити различите табеле стилова, а FX код за изглед који креирају аутоматски се генерише у позадини тј у главном коду и генерише се новим фајлом са екстензијом (.fxml). Резултат је FXCML датотека која се затим може комбиновати са Јава пројектом везивањем корисничког интерфејса за логику и ток апликације апликације.[43]

Java scene builder вам омогућава да лако поставите JavaFX контроле, графиконе, облике и контејнере, тако да можете брзо прототипирати кориснички интерфејс. Анимације и ефекти могу се неприметно применити на софистицираније корисничке интерфејсе.

[43] https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html, датум приступа 08.11.2020. године

6.3. Шта је то Icon Pichon?

Icon Pichon представља додатни програм за дизајнирање графичког интерфејса за кориснике. Програм функционише тако што постоји листа графичких поинтера које можете додати свом пројекту како би помогао побољшању изгледа апликације.[44]



Слика 5. Pichon aplikacija

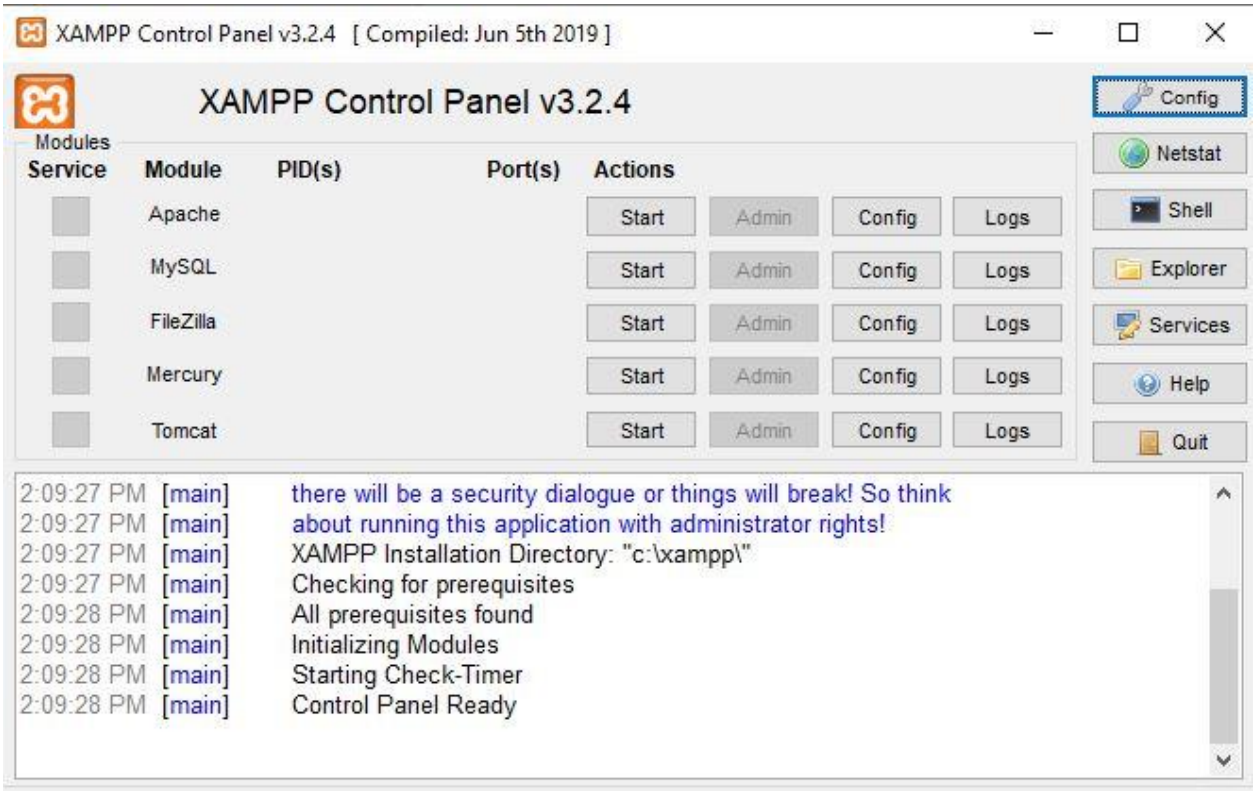
Извор: https://icons8.com/app/windows, датум приступа 20.11.2020. године

6.4. Шта је то XAMPP?

XAMPP је бесплатни пакет отворених укрштених платформи за веб сервере. Програм је развијен од стране Apache Friends. Састоји се углавном од Apache HTTP сервера, базе података MariaDB и тумача за скрипте написане на PHP и Perl програмским језицима. Будући да већина стварних примена веб сервера користи исте компоненте као xampp, то омогућава прелазак са локалног тест сервера на живи сервер.[45]

[44] https://icons8.com/app/windows, датум приступа 10.11.2020. године

[45] https://www.apachefriends.org/index.html, датум приступа 15.11.2020. године



Слика 6. Xampp aplikacija

Извор: https://www.apachefriends.org/index.html, датум приступа 20.11.2020. године

Једноставност примене xampp-а значи да програмер може брзо и једноставно инсталирати Wamp или Lamp на оперативни систем, с предношћу у односу на уобичајене програмске додатке као што су WordPress и Joomla.

7. ИТ ИСТРАЖИВАЊЕ

У Републици Србији је 2019. године спроведено треће истраживање Стартит-а у којем је 1.108 програмера учествовало у анкети и помогло аналитичарима око питања о условима у којима обављају своје пословне активности, начину на који се образују, искуствима која стичу у ИТ области и плановима које пројектују. Резултати истраживања биће приказани у наставку рада, а на основу њих можемо презентовати исходе који ће дати стручну слику о нашој програмерској сфери у садашњем времену.[46]

За почетак, једна пријатна вест је да је у 2019. години евидентиран извесни пораст броја припадница лепшег пола која су своју афирмацију пронашле у сектору програмирања, та бројка достиже 14,5% у односу на 12% у 2017. години, а 11% у односу на 2015. годину. Ови подаци доводе до закључка да је Република Србија у непосредној близини са светским просеком, а ако би се то приказало кроз бројке то би изгледало овако, по истраживању које спроводи компанија под називом Стацк Оверфлоњ, у Сједињеним Америчким Државама лепшу половину програмерске заједнице чине 11,7% испитаних.

Анализа показује да највише програмера и програмерки долази или се налази на територији града Београда. Према подацима који су приказани у анализи, треба обратити пажњу да се њихов удео знатно умањио у поређењу са анкетама које су спроведене претходних година са 56% на тренутних 39,9%. Ове године чак 20% учесника у анализи, свеукупно долази из градова који самостално не чине 1% програмерске сцене у Републици Србији, што нам говори о томе да су "измештањем" индустије из једног града и мањи градови у Републици добили шансу за учешћем на тржишној трци.

СТЕПЕН ОБРАЗОВАЊА ИТ СТРУЧЊАКА У РЕПУБЛИЦИ СРБИЈИ

Што се тиче образовних профила ИТ стручњака у Републици Србији, истраживање је показало да су испитаници са мастер дипломом знатно превазишли проценат људи који поседују само диплому средње стручне спреме. Проценат запослених са дипломом више школе или напредним степеном образовања је на прошлом истраживању достигао 70,9%, а 2019. године се увећао за 2,1%.

Генерално, не поседују сви запослени у овој области академске дипломе. Дефинитивно, као и у свакој професији постоје установе које производе потенционално најбоље студенте из те области у Републици Србији а то су:

- ФТН Универзитета у Новом Саду – 19,34%
- ЕТФ Универзитета у Београду – 13,13%
- ФОН Универзитета у Београду – 9,85 %
- ПМФ у Београду – 6,21%
- ЕТФ Универзитета у Нишу – 4,81%

[46] https://startit.rs/rezultati-istrazivanja-programerske-scene-u-srbiji-sve-je-vise-zena-u-struci-javascript-najpopularnija-tehnologija-plate-seniora-porasle/, датум приступа 20.11.2020. године.

СТЕПЕН ПОЗИЦИЈА У ПРОГРАМЕРСКОМ СЕКТОРУ У РЕПУБЛИЦИ СРБИЈИ

По расподели позиција у сфери програмирања у Србији, егзистира скоро идентична расподела процената — 33,2% припада јуниор девелоперима, 30,4% припада медиум девелоперима и на крају 36,4% припада уско стручним експертима у овој струци у Републици Србији тзв. Сениор девелоперима. Просечна старосна доб се може приказати кроз следеће податке:

• јуниор девелопери у Републици Србији у просеку имају 28 година (просечно време да би се стигло до ове позиције је 1,5 година)

• медиум девелопер у Републици Србији су у просеку искуснији за једну годину од јуниора што је 29 година (просечно време да би се стигло до ове позиције је 4,4 година)

• сениор девелопери у Републици Србији у просеку имају око 35 година (просечно време да би се стигло до ове позиције је 11 година).

Податак који није много репрезентативан јесте да програмерску индустрију у Републици Србији планира да напусти — 29,6% док је 6,6% већ почело да планира наредне кораке како би напустили Републику Србију и своју професионалну и индивидуалну каријеру наставили у некој другој земљи.

НАЈЧЕШЋЕ ПОЗИЦИЈЕ НА КОЈИМА ПРОГРАМЕРИ У РЕПУБЛИЦИ СРБИЈИ ПОСЛУЈУ?

Према истраживању најучесталије позиције које се појављују на Српској програмерској сцени могу се кроз проценте представити овако:

- back-end programeri - 46,31%
- front-end – 25,22%
- mobile – 7,48%
- data processing – 6,9%
- full-stack-developer – 4,7%

НАЈЧЕЋИ ПОСЛОВНИ ЗАХТЕВИ КОЈЕ ПРОГРАМЕРИ У РЕПУБЛИЦИ СРБИЈИ ОБАВЉАЈУ

Пословни захтеви као и у свакој озбиљној индустрији чине најразвноврснију лепезу пословних задатака које запослени у тој области требају да испуне. Наиме, тако је и у програмерској индустрији а она се кроз проценте може приказати овако:

- развијање за web – 71,6%
- стандалоне – 19,6%
- мобилни девелопмент – 8,8%

Развојни инжењери у Републици Србији су изузетно дисциплиновани традиционалним програмерским "писањем" кода, пошто ове године само 8,5% је оних који не користе ни једну другу технологију, у поређењу са 81,5% који своје пројетке публикују на Git-у као један од видова покушаја запошљавања или промене посла.

Грана програмирања у којој се знатно разликујемо у односу на друге земље које се баве оваквим видом пословања, јесте да се у свету и даље подстиче системско програмирање. Процентуално трка лидера у тој грани програмирања изгледа овако:

- Linux у свету је на примарној позицији (у Републици Србији се налази на трећем месту, 23,7%)
- Windows – 51,2%
- Mac Osa – 25,1%

СТАТИСТИЧКА ПОДЕЛА НАЈЧЕШЋИХ ПРОГРАМСКИХ ЈЕЗИКА У РЕПУБЛИЦИ СРБИЈИ

Својевремено истраживање приказало да су Java, PHP и JavaScript једни од најчешћих и најзаступљених програмских језика у Републици Србији. Према резултатима прошлих испитивања показало се да су процентуалне разлике између употребе наведених језика изузетно ниске. Међутим, 2019. године позиције су се мало промениле:

- JavaScript – 21%
- Java – 18,2% тенденција изузетно високога раста
- C# - 16,5%
- PHP – 12,5 % (izuzetno pozitivna novina)
- Python – 8,9%

Највеће задовољство је представити податак да поприличан број испитаника жели да настави са усавршавањем својих професионалних вештина у програмском језику који најбоље познаје. Оно што је изузетно интересантно, а може се навести је да су Python, Go, Kotlin и Swift за српске развојне инжењере изузетно привлачнији и преузимају примарне позиције у њиховим интересовањима него програмско окружење у коме заправо обављају своје пословне активности.

НАЧИНИ НА КОЈЕ СУ ЗАПОСЛЕНИ ДОМАЋИ РАЗВОЈНИ ИНЖЕЊЕРИ?

Подаци о начину запошљавања развојних инжењера у Републици Србији добијени истраживањем несумњиво говоре о томе да се највећи проценат њих запосли на тзв. традиционални вид запошљавања – 81%. Наравно, постоји и проценат оних који се самостално запосле тј. У програмерској сфери их називају фрелансерима, њих је око 8% учесника у анкети. Предузетници који у програмерској сфери пословања креирају свој пут и на том путу ангажују додатне колеге чине 5,5%, нажалост то је и исти број наших незапослених програмских развојних инжењера. Наравно, треба тежити што савршенијој слици али морамо се сложити да је то привредна грана која има мали број незапослених у Републици Србији. Постоји одређених број стручњака којима није проблем да раде "са стране" и тај број чини скоро 1/3. Следећи део истраживања базира се на део у коме се види колико наши програмерски стручњаци проводе времена у учењу и усавршавању својих вештина:

- мање од 5 часова недељно посвећује 42%
- између 5-10 часова недељно посвећује 25%
- мало више од 5% је оних који су жељни унапређења својих вештина и посвећују више од 20 часова недељно.

Податак који би највише обрадовао пореску управу Србије јесте да се скоро 30% наших запослених учесника у анкети изјаснило да би волело да отвори своју предузетничку радњу. Жељу да покрену свој посао, међу слободним програмерима (фрееланцерима) чини чак дупло већи број, што значи да би се чак 60% фрееланцера определило за предузетничке воде. Охрабрујући податак јесте да је чак нешто мало мање од 1/5 програмерских развојних инжењера ушло одмах по завршетку студија или у току студија у предузетничке воде.

Проценат развојних инжењера који су запослени у страним фирмама је у период иза нас порастао за 2% и то сада чини 40%, док је број оних који раде у фирмама Републике Србије опао за тај исти проценат од 2% и сада износи 47%, осталих 13% су компаније које су власнички мешовитог карактера. Развојни инжењери су се у Републици Србији показали као изузетно савесни и вредни па је најчешћи одговор испитаника био да не раде прековремено, али део пословања који би ту ажурност требао да прати и да се знатно поправи јесте да је само 56% оних којима је тај начин жтвовања свог личног времена укључен у њихову месечну зараду.

СТЕПЕН ЗАДОВОЉСТВА ПОСЛОМ КОД ПРОГРАМЕРА У РЕПУБЛИЦИ СРБИЈИ

Ово је један од најбитних фактора поред финансијске могућности зашто се млади опредељују за програмерску привредну грану. То је још један од услова који може обезбедити аргумент зашто је ова привредна грана у сталном успону јер се 72,7% програмера у Републици Србији изјаснило да своје задовољство на послу могу оценити

оценом 8, док је просечна оцена којом ова привредна грана може да се похвали 7,66 или висока просечна оцена.

Ово истраживање је спроведено и у већим градовима (центрима ИТ-а), а задовољство на послу кроз градове и технологије које користе приказано је кроз следеће просечне оцене:

- Суботица – 8,07
- Ниш – 7,79
- Нови Сад – 7,76
- Београд – 7,64
- Крагујевац – 7,5.

Просечне оцене добијене истраживањем за задовољство технологијом коју користе су:

- C++ - 8,62
- Python – 8,05
- JavaScript – 7,92
- SQL – 7,91
- Java – 7,58.

У оне који су задовољни својом пословном позицијом спадају и они о којима је малопре било приче, а то су они који имају урачунат прековремени рад у њихову месечну зараду. Њихова просечна оцена је 7,85, што је наравно боље у односу на другу групу којима није урачунат прековремени рад у њихову месечну зараду и где је та оцена значајно мања и износи 7,05.

Испитаници су од погодности које им послодавци током обављања својих пословних активности нуде навели су да им највише одговара флексибилно радно време компаније – 77%. Постоје и они запослени који су рекли да им атмосфера дома највише прија и они чине 2/3 испитаника и последња повлашћеност коју компаније нуде запосленима јесу службена путовања које воли више од ½ испитаника. Међу додатним погодностима у корак са горе поменутим иду:

- приватно здравствено осигурање
- приватно пензионо осигурање
- плаћене квалитетне обуке
- квалитетни тренинзи.

КАКО РАЗВОЈНИ ИНЖЕЊЕРИ НАЈЧЕШЋЕ ДОЛАЗЕ ДО ПОСЛА У РЕПУБЛИЦИ СРБИЈИ

Кроз текст који следи може се закључити да се у времену интернета и нових технологија, развојни инжењери највише запошљавају баш преко интернета. Статистика показује да се 73% програмера запосли преко ЛинкедИн-а. Почетнике у области програмирања тзв. јуниор девелопере најчешће контактирају преко емаила. Следећи вид доласка до перспективног посла јесте пријава на уско стручне семинаре које организују програмерске куће. Дефинитивно концепције као ЛинкедИна имају надмоћ у могућностима за контакт са жељеном компанијом. Програмери који се тренутно налазе у неким од компанија, најчешће су за ту позицију чули од некога ко је већ у тој индустрији тј. преко препоруке. Нажалост, јуниор развојни инжењери уколико нису стекли одређене контакте и конекције за време студија, мораће да уложе мало више напора како би дошли до првог посла. Прилике које се јуниор развојним инжењерима појављују најчешће се могу наћи преко портал ХеллоЊордл или Инфо Студ.

Како би компаније дошле у стадијум да буду изузетно интересантне за потенционалне запослене, пожељно је да испуне неке од услова које су издвојили испитаници, а у први план то су:

- разноврсност пројеката – 33%
- коректан вредносни систем између количине посла и плате – 25%
- могућност веће зараде – 18%
- сигурнија релација пословних односа између компаније и запосленог – 12%
- могућност бржег напредовања у тимском окружењу – 10%
- флексибилније радно време и могућност рада од куће – 2%.

ЗАРАДЕ РАЗВОЈНИХ ИНЖЕЊЕРА У РЕПУБЛИЦИ СРБИЈИ

Ово је део истраживања који верујем да су сви који су прошли кроз истраживање чекали са највећим нестрпљењем. Наиме, претходна истраживања из 2017. године показују податке који говоре о томе да су почетници (јуниор развојни инжењери) имали просечну плату од 700 еура, док су програмери који су се већ усталили у тој привредној грани медиуми имали зараду од око 1,299 еура, док су експерти из ове области за свој уложени труд и напор били плаћени са просечно 2,261 еура.

Истраживања о висини плата у програмском сектору у 2019. години приказују следеће податке:

- просечна месечна зарада јуниора пала је за 4 еура и сада износи 696 еура
- просечна месечна зарада медиума је порасла за 145 еура и сада износи 1.444 еура
- просечна месечна зарада сениор развојних инжењера је забележила највећи раст у износу од 358 еура и сада износи 2.619 еура.

Један јако битан показатељ за младе смо оставили на крају истраживања. Уколико узмемо да упоређујемо програмске језике у којима се обављају пословне активности наших развојних инжењера и просечну зараду коју поседују, највиша примања имају програмери који раде у Ц++. За оне који тек желе да се укључе у ову привредну грану истраживање је показало да је за почетнике Ц++ један од програмских језика који се најбрже учи. Оно што је јако занимљиво то је и један од језика који се предлаже и сугерише медиумима уз још један програмски језик, а то је Елиџир, а за сениоре тј. за оне који су овладали одређеним сегментом у програмерској привредној грани, а желе да и даље напредују препорука је усавршавање програмског решења Го.

Што се тиче опредељивања везаних за образовање, истраживање је показало да формула за већом платом најчешће изгледа овако:

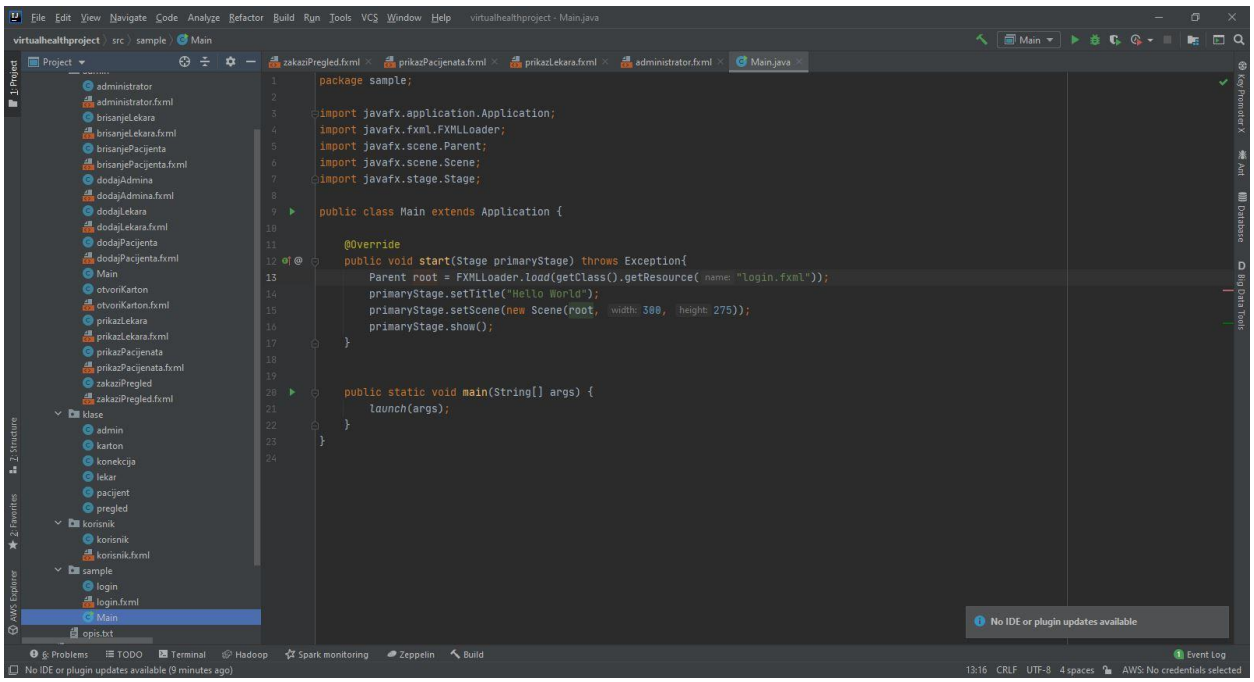
Што веће формално и практично знање + Напоран рад + Усавршавање = Боље плаћен посао

На основу свега наведеног долазимо до следећег закључка: Драги почетници, спремите се за напоран, узбудљив и динамичан рад који ће вам се исплатити у времену које је пред вама.

Зашто је приложено ово ит истаживање приложено раду? Одговор је изутено лак. У колико сте пажљиво читали истаживање сигурно сте приметили да се у сваком од грана најчешће помиње јава у високим процентима или у технологијама које су у порасту. Јава је један од програмских језика који су до сада били мало скрајнути у страну али долази време великог повратка програмског језика Јава. Кодирање у Јави се може применити на сваку од ових технологија (једино се не препоручује равој web сајтова) што Јава програмирање ставља на прво позицију нових старих пожељних језика за учење и усавршавање.

8. ПРОЈЕКАТ ВИРТУЕЛНОГ ЗДРАВСТВЕНОГ КАРТОНА

Пројекат је креиран у једном микро окружењу како би се приказао начин фукнционисања виртуелног здравствненог картона. Пројекат је могуће дефиницати као десктоп апликација. Подаци које пројекат користи (база података) налазе се у виртуелном серверу тј на (PHP Admin) страници. Овакав вид базе података је одабран из разлога што је то један од најпрактичнијих нацина за израду оваквих пројеката. Могуће унапређење апликације може се видети кроз развој (андроид или иос) апликације.



Слика 7. Приказ комплетног кода апликације

У даљем делу раду биће представљен шематски приказ самог програма и аналитички објашњене све функције које пројекат поседује као што су:

1. Неопходно логовање како би се одабрао ниво дозвола (кориснички или админ налог).
2. Додавање, брисање, и измена пацијента
3. Додавање, брисање, измена и приказ лекара.
4. Додавање, брисање и измена админа.
5. Отварање картона
6. Заказивање прегледа.

У колико се унесу параметри да се корисник улогује као гост (основни ниво административних дозвола) биће му приказан следећи прозор.

8.1. Кориснички ниво апликације

Подаци које корисник види на слици 8. су:

9. ИД,
10. Име,
11. Презиме,
12. Кориме (корисничко име),
13. Лозинка,

14. Шифра картона и

15. Приказ времена заказаног прегледа

На тексту који се налази испод слике 8 се може видети програмски код корисничког нивоа. Издвојени део кода који се налази испод слике приказује корисничке информације које је могуће видети приликом покретања програма. Најбитнија информација која би могла да се нагласи из кода који се налази испод слике 8 јесте Connection conn = new konekcija().vratiKonekciju();. Означава конекцију програма са базом података у којој су смештени сви потребни подаци који програм приказује.

Корисник у овом прозору не мозе ништа да мења већ само да има увид у податке.

Слика 8. Приказ корисничког нивоа

```
public class korisnik implements Initializable {

    public Label info1,info2;
    public Label info3;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        pacijent p = login.p;
        info1.setText(
                        "korisnicke informacije\n"+
                        "id: "+p.getId()+"\n"+
                        "ime: "+p.getIme()+"\n"+
                        "prezime: "+p.getPrezime()+"\n"+
                        "korime: "+p.getKorime()+"\n"+
                        "lozinka: "+p.getLozinka()
        );
```

```
        Connection conn = new konekcija().vratiKonekciju();


        karton k = karton.vratiKarton(p.getId());
        if(k!=null){
            info2.setText("sifra kartona: "+k.getSifra());
            lekar l = lekar.vratiLekara(k.getId_lekara());
            info3.setText(
                            "id: "+l.getId()+"\n"+
                            "ime: "+l.getIme()+"\n"+
                            "prezime: "+l.getPrezime()
            );
            ArrayList<pregled> niz = pregled.vratiPreglede(k.getId());
            String s = "pregledi\n";
            for(int i=0;i<niz.size();i++){
                s+=niz.get(i).getDatum()+"\n";
                System.out.println(niz.get(i).getDatum());
            }
            info3.setText(s);

        }else{
            info2.setText("nema otvorenog kartona");
        }


    }
}
```

8.2. Административни мени апликације

Административни ниво дозвола кориснику омогућава управљање комплетном апликацијом. Администатативном менију се приступа тако што се унесу одговарајући подаци за администратора и цекира се опција админ. Административне дозволе се односе као што је већ наведено на функције додавања, брисања и едитовања података везаним за пацијента или лекара као и заказивање прегледа као и отварање картона.

Слика 10. Приказ административног нивоа

```
public class administrator implements Initializable {
    @FXML
    public JavaFX.scene.control.Label podaci;

    private void loadWindow(String location,String title) throws IOException
{
        Parent root = FXMLLoader.load((getClass().getResource(location)));
        Scene sc = new Scene(root);
        Stage st = new Stage(StageStyle.DECORATED);
        st.setScene(sc);
        st.setTitle(title);
        st.show();
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        admin a = sample.login.a;
        podaci.setText(
                "id: "+a.getId()+"\nime: "+a.getIme()+"\nprezime:
"+a.getPrezime()+"\nkorime: "+a.getKorime()+"\nlozinka: "+a.getLozinka()
        );
    }


    public void dodavanjePacijenta(ActionEvent actionEvent) {
        System.out.println("dodavanje pacijenta");
        try {
            loadWindow("/admin/dodajPacijenta.fxml","ulogovani ste");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Кроз приказани део кода може се видети како административни профил прикупља информације, путање одакле долазе подаци у програм, као и поља у којима се приказују жељени скуп информација.

8.3. Мени пацијента у апликацији

Мени пацијента који се налази приказан на слици 11,налази се у административном новиу апликације, има за циљ да приказе кроз три подељене опције понуди жељене типове информација које програм може да понуди кориснику. Подаци који могу да се добију се односе на додавање, приказ и уклањање пацијената.

Слика 11. Приказ пацијентовог мени-а

```
public class pacijent {
    private int id;
    private String ime,prezime,korime,lozinka;

    public pacijent(int id, String ime, String prezime, String korime, String
lozinka) {
        this.id = id;
        this.ime = ime;
        this.prezime = prezime;
        this.korime = korime;
        this.lozinka = lozinka;
    }

    public pacijent(String ime, String prezime, String korime, String
lozinka) {
        this.ime = ime;
        this.prezime = prezime;
        this.korime = korime;
        this.lozinka = lozinka;
    }
```

Кроз део кода приказан је начин креирања класе пацијент.

8.3.1. Додавање пацијената

Приказ опције додавања пацијента се добије тако што се у менију пацијента корисник кликне опцију додај пацијента. Приказ менија се налази на слици 12. Поља која се налазе у овом менију ће кориснику омогућити да у програмску базу података унесе и сачува

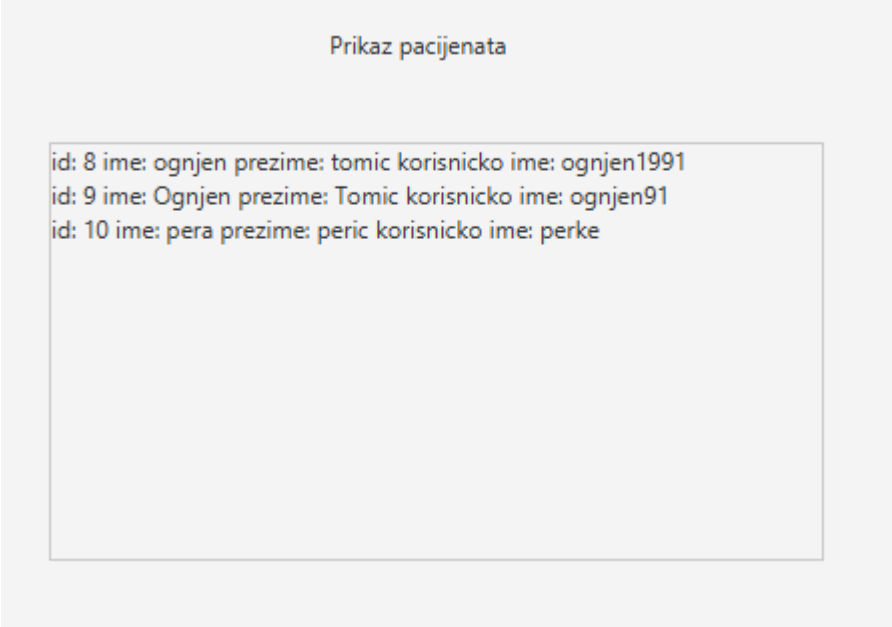
податке везане за пацијента. У даљем делу текста слици 12 ће бити припојен и део кода који се односи на додавање пацијента.

Слика 12. Приказ опције додавање пацијента

```
public class dodajPacijenta {

    public Button dodaj;
    public TextField ime;
    public TextField prezime;
    public TextField korime;
    public TextField lozinka;

    public void dodaj(ActionEvent actionEvent) {
        String imeText = ime.getText();
        String prezimeText = prezime.getText();
        String korimeText = korime.getText();
        String lozinkaText = lozinka.getText();

        pacijent p = new
pacijent(imeText,prezimeText,korimeText,lozinkaText);
        p.upisi();

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(null);
        alert.setContentText("pacijent dodat");
        alert.showAndWait();
        ((Node)actionEvent.getSource()).getScene().getWindow().hide();

        }

    }
```

8.3.2. Приказ пацијената

Опција приказ пацијента се као и опција додавање пацијената налази се у менију пацијента. Листи пацијената се приступа тако што корисник одабере опцију приказ пацијента. Након тога се појави графички приказ који се налази на слици 13 која се налази испод текста. Код који се налази испод слике приказује како програм долази до листе пацијената.



Слика 13. Приказ опције приказ пацијената

```
public class prikazPacijenata implements Initializable {
    public ScrollPane prikaz;


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        String s="";
        pacijent niz[] = klase.pacijent.vratiSvePacijente();
        for(int i=0;i<niz.length;i++){
            s+="id: "+niz[i].getId()+" ime: "+niz[i].getIme()+" prezime:
"+niz[i].getPrezime()+" korisnicko ime: "+niz[i].getKorime()+"\n";
        }
        Label l =new Label(s);
        prikaz.setContent(l);
    }
}
```

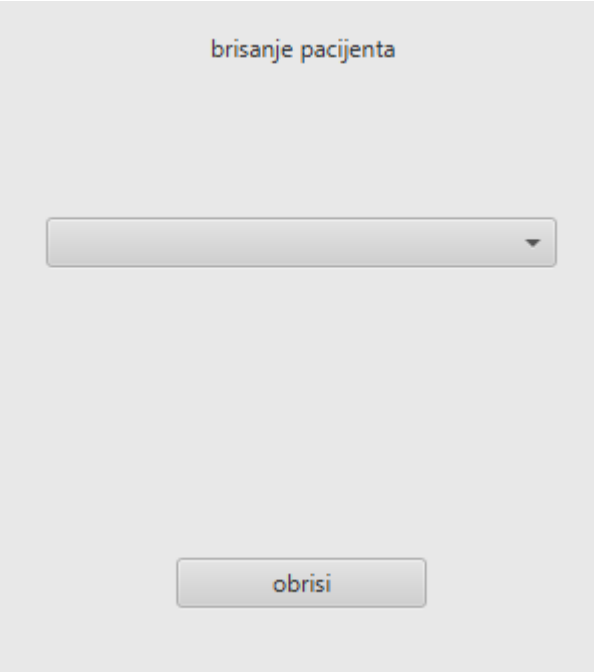
8.3.3 Брисање пацијента

Брисање пацијента јесте последња опција из менија пацијента. Сврха ове опције јесте да уклони пацијента који је нпр. завршио процес лечења. Начин на који ова опција програма функционише приказан је кроз код који се налази испод слике 14. Начин брисања је такав да из падајућег менија корисник може да одабере пацијента по ИД фактору, где се још може видети и:

1. Име пацијента,

2. Презиме пацијента,
3. Корисничко име пацијента и
4. Пасворд пацијента.

Пацијенти се бришу кроз све факторе низа типа (int).



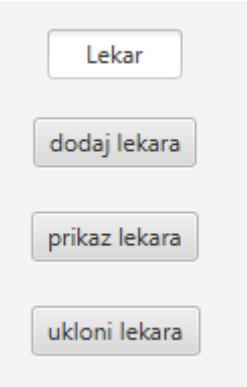
Слика 14. Приказ опције брисања пацијената

```
public class brisanjePacijenta   implements Initializable {
    //pacijent.getItems().add(pacijenti[i].getId()+"-"+pacijenti[i].getId());
    //Integer.parseInt(((String) pacijent.getValue()).split("-")[0])
    public ComboBox lekari;
    public Button obrisi;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        pacijent niz[] = klase.pacijent.vratiSvePacijente();
        for(int i=0;i<niz.length;i++){
            lekari.getItems().add(niz[i].getId()+"-"+niz[i].getIme()+"-
"+niz[i].getPrezime()+"-"+niz[i].getKorime()+"-"+niz[i].getLozinka());
        }
    }

    public void obrisi(ActionEvent actionEvent) {
        String[] niz = ((String)lekari.getValue()).split("-");
        pacijent l = new pacijent(
                Integer.parseInt(niz[0]),
                niz[1],
                niz[2],
                niz[3],
                niz[4]
        );
        l.obrisi();
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setHeaderText(null);
        alert.setContentText("pacijent obrisan");
        alert.showAndWait();
```

```
        ((Node)actionEvent.getSource()).getScene().getWindow().hide();
    }
}
```

8.4. Мени лекара у апликацији

Мени lekara који се налази приказан на слици 14,налази се у административном нивоу апликације, има за циљ да приказе кроз три подељене опције понуди жељене типове информација које програм може да понуди кориснику. Подаци који могу да се добију се односе на додавање, приказ и уклањање лекара.



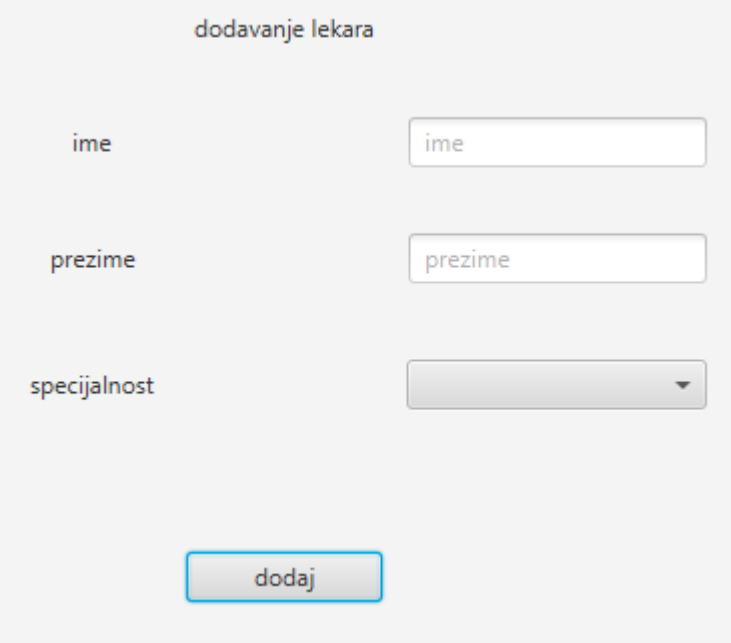
Слика 14. Приказ опције брисања пацијената

```
public class lekar {
    private int id;
    private String ime,prezime,specijalnost;

    public lekar(int id, String ime, String prezime, String specijalnost) {
        this.id = id;
        this.ime = ime;
        this.prezime = prezime;
        this.specijalnost = specijalnost;
    }

    public lekar(String ime, String prezime, String specijalnost) {
        this.ime = ime;
        this.prezime = prezime;
        this.specijalnost = specijalnost;
    }
```

Кроз део кода приказан је начин креирања класе лекар.

8.4.1. Додавање лекара

Приказ опције додавање лекара се добије тако што се у менију лекар корисник кликне опцију додај лекара. Приказ менија се налази на слици 15. Поља која се налазе у овом менију ће кориснику омогућити да у програмску базу података унесе и сачува податке везане за лекара, као један од најбитнијих делова јесте бирање специјалности лекара који се додаје. У даљем делу текста слици 15 ће бити припојен и део кода који се односи на додавање лекара.



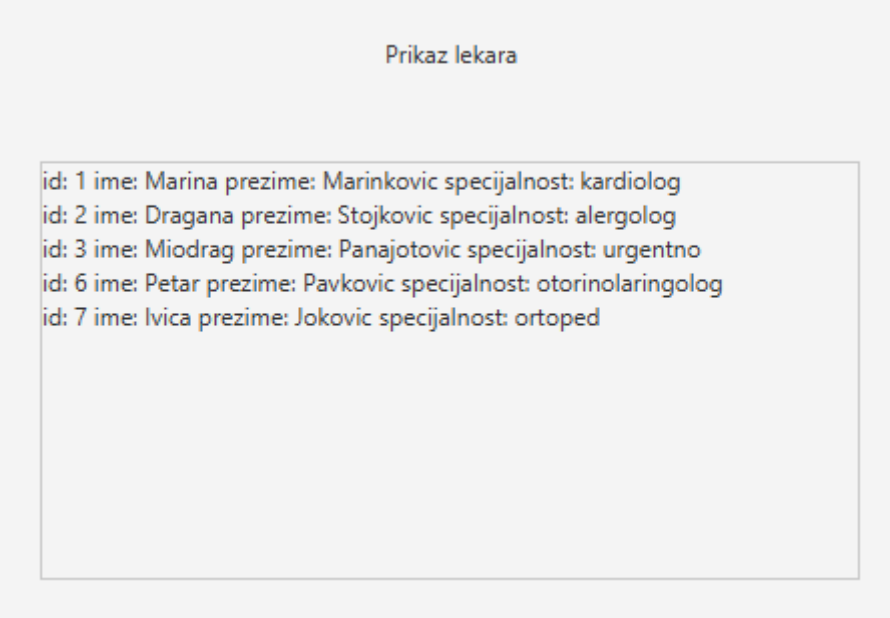
Слика 15. Приказ опције додавања лекара

```
public class dodajLekara implements Initializable {
    public ComboBox specijalnost;
    public TextField prezime;
    public TextField ime;
    public Button dodaj;

    public void dodajLekara(ActionEvent actionEvent) {
        String textSpecijalnost = (String) specijalnost.getValue();
        String textPrezime = prezime.getText();
        String textIme = ime.getText();

        lekar l = new lekar(textIme,textPrezime,textSpecijalnost.split("-
")[1]);
        l.upisi();

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setHeaderText(null);
        alert.setContentText("lekar dodat");
        alert.showAndWait();
        ((Node)actionEvent.getSource()).getScene().getWindow().hide();

    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        Connection conn = new konekcija().vratiKonekciju();
        try {
            Statement stmt = conn.createStatement();
            String sql = "SELECT * FROM specijalnosti";
            ResultSet rs = stmt.executeQuery(sql);
            while(rs.next()){
                specijalnost.getItems().add(rs.getInt("id")+"-
"+rs.getString("specijalnost"));
            }
            conn.close();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
```

```
    }
}
```

8.4.2. Приказ лекара

Опција приказ лекара се као и опција додавање лекара налази се у менију лекара. Листи лекара се приступа тако што корисник одабере опцију приказ лекара. Након тога се појави графички приказ који се налази на слици 16 која се налази испод текста. Код који се налази испод слике приказује како програм долази до листе лекара.



Слика 16. Приказ опције листање лекара

```
public class prikazLekara implements Initializable {
    public ScrollPane prikaz;


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        String s="";
        lekar niz[] = klase.lekar.vratiSveLekare();
        for(int i=0;i<niz.length;i++){
            s+="id: "+niz[i].getId()+" ime: "+niz[i].getIme()+" prezime:
"+niz[i].getPrezime()+" specijalnost: "+niz[i].getSpecijalnost()+"\n";
        }
        Label l =new Label(s);
        prikaz.setContent(l);
    }
}
```

8.4.3. Брисање лекара

Брисање лекара јесте последња опција из менија лекара. Сврха ове опције јесте да уклони лекара који је нпр. дао отказ у једној здравственој јединици. Начин на који ова опција програма функционише приказан је кроз код који се налази испод слике 17. Начин брисања је такав да из падајућег менија корисник може да одабере лекара по ИД фактору, где се још може видети и:

1. Име лекара,

2. Презиме пацијента,
3. Специјалност.

Лекари се бришу кроз све факторе низа типа (int).

Слика 17. Приказ опције брисања лекара

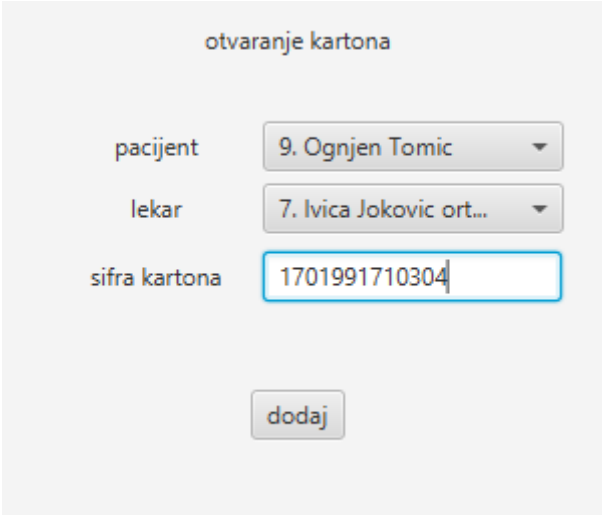
```
public class brisanjeLekara  implements Initializable {
    //pacijent.getItems().add(pacijenti[i].getId()+"-"+pacijenti[i].getId());
    //Integer.parseInt(((String) pacijent.getValue()).split("-")[0])
    public ComboBox lekari;
    public Button obrisi;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        lekar niz[] = klase.lekar.vratiSveLekare();
        for(int i=0;i<niz.length;i++){
            lekari.getItems().add(niz[i].getId()+"-"+niz[i].getIme()+"-
"+niz[i].getPrezime()+"-"+niz[i].getSpecijalnost());
        }
    }

    public void obrisi(ActionEvent actionEvent) {
        String[] niz = ((String)lekari.getValue()).split("-");
        lekar l = new lekar(
                Integer.parseInt(niz[0]),
                niz[1],
                niz[2],
                niz[3]
        );
        l.obrisi();
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setHeaderText(null);
        alert.setContentText("lekar obrisan");
        alert.showAndWait();
        ((Node)actionEvent.getSource()).getScene().getWindow().hide();
    }
}
```

8.5. Отварање картона

Опција отварања картона омогућава кориснику административних дозвола да отвори нови здравствени картон пацијенту. Подаци које корисник може да одабере јесу:

1. Пацијент (из падајуцег менија корисник бира ком пацијенту се отвара нови здравствени картон),
2. Лекар (из падајућег менија корисник бира ком лекару ће нови пацијент бити додељен, где треба обратити пажњу на специјалност лекара коме се пацијент додељује) и
3. Шифра картона (корисник додељује сифру пацијентовог здравственог картона по којој ће се он на даље водити).

Испод текста се налази слика 18 која приказује пример отварања једног здравственог картона са свим параметрима.



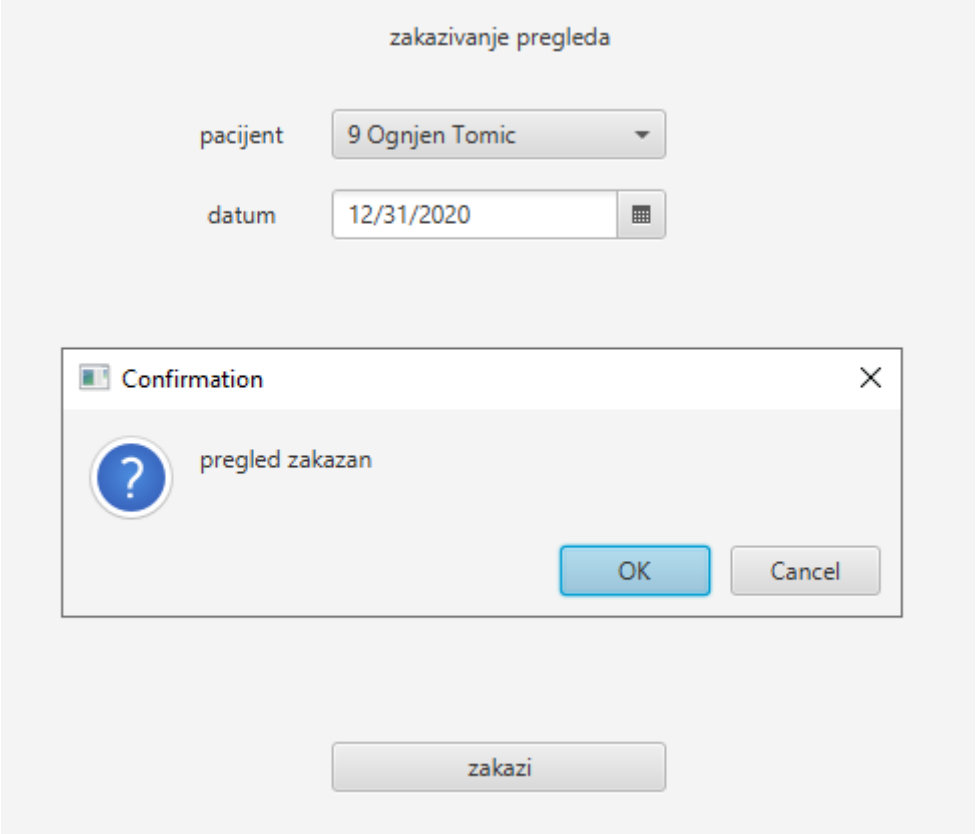
Слика 18. Приказ примера отварања картона

```
public class otvoriKarton implements Initializable {
    public ComboBox pacijent;
    public ComboBox lekar;
    public TextField sifra;
    public Button dodaj;
    //.getItems().add

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        pacijent[] pacijenti = klase.pacijent.vratiSvePacijente();
        lekar[] lekari = klase.lekar.vratiSveLekare();

        for(int i=0;i<pacijenti.length;i++){
            pacijent.getItems().add(pacijenti[i].getId()+". "
+pacijenti[i].getIme()+ " " +pacijenti[i].getPrezime());
        }

        for(int i=0;i<lekari.length;i++){
            lekar.getItems().add(lekari[i].getId()+". " +lekari[i].getIme()+
" " +lekari[i].getPrezime()+ " " +lekari[i].getSpecijalnost());
        }
    }
```

```
    public void dodaj(ActionEvent actionEvent) {
        karton k = new karton(
                Integer.parseInt(((String) pacijent.getValue()).split("-
")[0]),
                Integer.parseInt(((String) lekar.getValue()).split("-")[0]),
                sifra.getText()
        );
        k.upisi();
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setHeaderText(null);
        alert.setContentText("karton otvoren");
        alert.showAndWait();
        ((Node)actionEvent.getSource()).getScene().getWindow().hide();
    }
}
```

8.6. Заказивање прегледа

Последња опција програма коју он нуди зове се заказивање прегледа. Овом опцијом је кориснику административних дозвола омогућено да закаже преглед одређеном пацијенту (којег бира из падајућег менија), одрађеног датума и наравно на крају је потребно да само одабере опцију ок да би се процес завршио. Процес је приказан на слици 19 која се налази испод текста, као и кроз део кода.



Слика 18. Приказ процеса заказивања прегледа

```
public class zakaziPregled  implements Initializable {
    public ComboBox pacijenti;
    public DatePicker datum;
    public Button zakazi;

    public void zakazi(ActionEvent actionEvent) {
        String[] niz = ((String)pacijenti.getValue()).split(" ");
        int id = Integer.parseInt(niz[0]);

        LocalDate localDate = datum.getValue();
        Instant instant =
Instant.from(localDate.atStartOfDay(ZoneId.systemDefault()));
        Date date = Date.from(instant);

        String sql = "INSERT INTO pregled VALUES(null,(SELECT id FROM karton
WHERE id_pacijenta="+id+"),'"+date+"')";
        Connection conn = new konekcija().vratiKonekciju();
        try {
            Statement stmt = conn.createStatement();
            stmt.executeUpdate(sql);
            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
            alert.setHeaderText(null);
            alert.setContentText("pregled zakazan");
            alert.showAndWait();
            ((Node)actionEvent.getSource()).getScene().getWindow().hide();
        } catch (SQLException throwables) {
            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
            alert.setHeaderText(null);
            alert.setContentText("greska proverite da li korisnik ima otvoren
karton");
            alert.showAndWait();
            ((Node)actionEvent.getSource()).getScene().getWindow().hide();
        } catch (Exception e){

        }
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        pacijent niz[] = klase.pacijent.vratiSvePacijente();
        for(int i=0;i<niz.length;i++){
            pacijenti.getItems().add(niz[i].getId()+" "+niz[i].getIme()+"
"+niz[i].getPrezime());
        }
    }
}
```

ЗАКЉУЧАК

Као један од најраспрострањенијих програмских језика, Јава, чије су основне карактеристике објектна оријентисаност, висока робустност и релативна једноставност управљања и функционисања, најпре је примењивана на интернет претраживачима, да би се временом почела користити и на рачунарима и другим информационим технологијама. За разлику од процедуралних програмских језика, Јава није специфична за конкретни хардвер, већ је машински независна, што заправо значи да се може применити на бројним уређајима. Имплементација се обавља на основу такозване Јава виртуелне машине, која преводи изворни програмски код у Јава бајткод. Дакле, Јава је дизајнирана тако да је у самом старту машински независна, па апликација која је написана у Јави захтева само један скуп изворног кода, без обзира на колико ће се хардвера, односно платформи та апликација применити. Треба нагласити и то да је Јава статични језик, што подразумева да свака варијабла и њен израз имају своје време компајлирања. За писање програма може се користити било који програм, односно text editor, што додатно умањује време и трошкове програмирања у овом језику, у односу на друге програмске језике.

Пре него што је извршен приказ Јава апликације која треба да обезбеди подршку, у раду је извршен теоријски приказ концепта Јаве, односно њен развој, карактеристике, објектна оријентисаност и кључни концепти програмирања. Након тога је извршен приказ програмирања апликације за потребе здравственог картона, заједно са приказом ИТ истраживања.

Користећи се свим чињеницама и уз коришћену литературу која је била потребна како би се рад завршио као и истраживање које је само још један од аргумената који могу само да потврде колики се равој јава програмирања може очекивати. Уз скромне могућности пројекта који је коришћен у овом дипломском раду могао се видети само делић једног јако великог и изузетно сложеног система јава програмирања. Једна од најбитнијих могућности које овај програмски језик нуди јесте да се до развоја једне апликације као и њеног усавршавања може доћи на много начина а сви они зависе само и искључиво од знања особе која користи неке од развојних центара за кодирање. Сам значај овог дипломског рада се своди на могућност и шансу која омогућава програмеру да је приложи као пројекат приликом конкурисања за неко јуниорско програмерско место и да му омогући даље и боље усавршавање које Вам је неопходно како би остали и опстали у програмерској струци.

Што се тиче начина унапређења апликације, сматрам да то у највећој мери зависи од знања и уложеног труда. Унапређење видим и кроз побољшање у редизајну базе података, како њене брзине тако и података који се налазе у њој. Побољшања се могу применити и на сам графички дизајн аликације, као и на унапређење постојећих и додавање нових функционаности и као последњи и најбитнији облик јесте комерционализација производа и могућност примене у реалном окружењу и тржишту.

Литература

1. Eck, D.J., 2006, Introduction to programming using Java, Hobart and William Smith Colleges, Geneva
2. Eckel, B., 2000, Thinking in Java, 2nd edition, Prentice Hall, New Jersey
3. Farrell, J., 2016, Java programming 8th edition, Cengage learning, Boston
4. Girau, L,, 2018, Object oriented programming, African Virtual University, Nairobi.
5. Мићовић, Н., 2016, Увод у програмски језик Јава, SystemPro, Београд.
6. Harbour, J.S., 2012, Begining Java: SE 6 game programming, 3rd edition, Cengage learning, Boston
7. Ivanović, M., Budimac, Z., Mishev, A., Bothe, K., Jurca, L., 2013, Java across different curricula, courses and countries using a common pool of teaching material, Informatics in Education, 12(2)
8. Poo, D., Kiong, D., Ashok, S., 2008, Object oriented programming and Java, Springer, Berlin
9. Roberts, E.S., 2006, *The art and ascience of Java*, Stanford University, California.
10. Sawitch, W., 2016, Absolute Java 6th edition, Pearson, Boston.
11. Сарачевић, М., 2015, *Програмски језик Јава*, Универзитет у Новом Пазару, Нови Пазар.

Интернет извори:

1. https://cet.rs/wp-content/uploads/2017/06/Java_2_JDK_5_Pog_01_Od_pocetka.pdf
2. https://www.freejavaguide.com/history.html
3. https://www.geeksforgeeks.org/the-complete-history-of-java-programming-language
4. https://www.javatpoint.com/cpp-vs-java
5. https://www.javatpoint.com/features-of-java
6. https://www.javatpoint.com/history-of-java
7. https://www.javatpoint.com/java-data-types
8. https://www.javatpoint.com/JavaFX-tutorial
9. https://www.javatpoint.com/JavaFX-ui-controls
10. https://www.javatpoint.com/java-swing
11. https://www.javatpoint.com/java-tutorial
12. https://www.javatpoint.com/jvm-java-virtual-machine
13. https://www.javatpoint.com/object-and-class-in-java
14. https://www.javatpoint.com/object-class
15. https://www.javatpoint.com/operators-in-java
16. https://www.javatpoint.com/runtime-polymorphism-in-java
17. https://www.javatpoint.com/JavaFX-tutorial
18. https://www.oracle.com/java/technologies/javase/JavaFXscenebuilder-info.html
    https://icons8.com/app/windows
19. https://www.apachefriends.org/index.html
20. https://startit.rs/rezultati-istrazivanja-programerske-scene-u-srbiji-sve-je-vise-zena-u-struci-javascript-najpopularnija-tehnologija-plate-seniora-porasle/

21. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/package-frame.html
22. https://bs.eferrit.com/sta-je-javafx/
23. https://www.vps.ns.ac.rs/Materijal/mat23263.pdf
24. http://poincare.matf.bg.ac.rs/~kartelj//nastava/DPJ2015/01/1.JavaFX.Uvod.pdf