Priprema za studentsku praksu

Zadatci – Software DEVELOPMENT Engineer

v2.1

# Teorijski dio

Prođite kroz listu pojmova iz prvog stupca tablice ispod s ciljem da upoznate, proučite i razumijete što je moguće više pojmova.

|  |  |
| --- | --- |
| **PROGRAMSKI JEZIK JAVA** | |
| **Uvod** | |
| Klasa | Klasa je predložak za stavaranje podataka i metoda za objekt. Objekt je isntanca klase. Klasa se instancira sa ključnom riječi 'class'. |
| Objekt | Objekt je dio klase. Stanje objekta je pohranjeno u varijablama, dok metode služe da se prikaže ponašanje objekta. Stvoraju se pri pokretarnju programa iz predložaka(klasa). |
| Metoda | Metoda ima nekakvu funkcionalnost, koja se provodi tako da se metoda pozove kako bi se izvršila, i ako je potrebno predaju joj se potrebni paramteri. Najveća korist im je da se kod može više puta izvršiti, jednom se napiše a može se pozivati koliko god treba. |
| Varijabla | Varijable su „kontejneri“ koji spremaju vrijednost podataka. Moraju imati određeni tip podatka koji spremaj kao npr: String, Int, Float, Double, Char... |
| Sučelje | Kao i klasa može imati varijable i metode, ali metode nemaju funkcionlnost, kada klasa nasljedi sučelje mora implementirati i metode te im dati funkionalnost. |
| Nasljeđivanje | Klase mogu nasljediti dio ili cijelu funkcionalnost neke druge klase. |
| Overload | Može se postići ako imamo više metoda istog imena, ali da primaju drugačije argumente. |
| API | Služi da se mogu koristiti već napisani programi za rješenje problema. |
| JDK | Java Development kit je razvojno okruženje za izgradnju aplikacija, te sadrži potrebne alate. |
| JRE | Java Runtime Enviroment nam pruža biblioteke i resurse za naš program. |
| JVM | Java Virtual Machine nam pretvara byrecode u strojni kod. |
| jar | Java Archive nam sprema i čuva potrebne datoteke za pokretanje programa. |
| j2se | Java 2 Platforma(Core Java) je najjednostavnija i standardna verzija jave, većinom se koristi za kreiranje aplikacija za „desktop“. |
| j2ee | Java 2 Platform Enterprise Edition je nagorađena verzija J2SE i koristi se za izgranju „enterprise“ aplikacija. |
| OpenJDK vs OracleJDK | OpenJDK je „open source“ i koristi Javu s „GNU General Public License“, dok Oracle JDK nije besplatan i koristi komercionalnu licencu. |
| Maven | To je framework koji sadrži sve potrebne biblioteke koje su nam potrebne pri razvoju aplikacije u Javi. |
| Ant | Apache Ant je alat koji se koristi za automatizaciju izgradnje aplikacija. |
| Gradle | Sličan je kao Ant, ali koristi i Maven, to je osnovni alat za igradnju aplikacija za Android sučelje, isto tako podržava izgradnju više projekata. |
| Koja su prva 4 byte-a svake klase (class file-a) u hex formatu i zašto? | Prva 4 byte-a su CA FE BA BE, po James Gosling-u to je magičan broj, da ne pišem sve možete kliknuti [ovdje](https://www.quora.com/The-first-4-bytes-of-every-Java-class-file-have-the-magic-value-of-0xCAFEBABE-How-was-this-value-chosen) za cijelu pricu. |
| **Konvencija naziva** | |
| Naziv klasa | Trebaju biti imenice napisane velikim početnim slovom |
| Naziv metoda | Trebaju biti glagoli napisani malim slovom, ali ako naziv sadrži više riječi onda se koristi „camel case“ to je npr: racunanjePostotkaOcjena. |
| Naziv varijabli | Isto kao i metode idu malim slovom i koristi se „camel case“ u slučaju više riječi. |
| **Tipovi podataka** | |
| Primitivni tipovi | Oni specificiraju veličinu i tip varijable, ima ih 8: byte, short, int, long, float, double, boolean i char |
| Referencirajući tipovi | To su tipovi koji se stvaraju od primitivnih tipova npr. od više char-ova možemo dobiti String |
| **Operatori** | |
| Aritmetički | Služe za float i int tipove podataka, ima ih 12: +, ++, +=, -, --, -=, \*, \*=, /, /= , %, %= |
| Relacijski | Služe da se provjeri odnos 2 varijable, ima ih 6: ==, !=, <, >, >=, <= |
| Operatori uvjeta | Služe da se provjere nekakvi uvjeti, ima ih 3: &&, || i tercijarni operator ?: |
| Operatori pridruživanja | Služe za pridruživanje vrijednosti varijablama ima ih 6: =, +=, -=, \*=, /=, %= |
| **Kontrola toka: odlučivanje** | |
| if | Zadaju se uvjeti, ako su oni zadovoljeni izvršava se određena radnja |
| if – else | If je isti, a ako uvjet kod if-a nije zadovoljen onda se izvršava određena radnja pod else-om |
| if – else if – else | If je isti, else if se izvodi ako se ne ulazi u if i on isto mora kao if imati uvjete za izvođenje koda, else je isti |
| switch – case | Predaje mu se određeni argument i na temelju njega se izvodi određeni „case“ u kojemu je nekakva funkcionalnost, na kraju ima „default case“ koji ima istu svrhu kao else kod if-a. |
| **Kontrola toka: petlje** | |
| for | To je petlja u kojoj imamo iterator u kojemu imamo zadano koliko će se puta petlja izvrtiti, kada uvjeti u iteratoru nisu više zadovoljeni izalzi se iz petlje, npr: (int i = 0; i < 10; i++), for s ovakvim iteratorom će se izvrtiti 10 puta dok je uvjet zadovoljen, nakon toga se izlazi iz petlje. |
| while | Dok je uvjet zadovoljen kod se izvršava, ali ako uvjet odmah nije zadovoljen kod u while-u se neće niti jednom izvršiti. |
| do – while | Slično kao i while, ali kod se uvijek izvrši barem jednom te se tek onda provjera uvjet u while-u, pa ako uvjet ne odgovara petlja se više ne izvršava. |
| **Kontrola toka: bezuvjetni skok** | |
| break | Služi kako bi se petlja prekinula, najčešće se koristi u switch case-u. |
| continue | Služi nam za preskakanje iteracije, npr ako imam uvjet (int i = 0; i < 5; i++) i if(i==2) continue, i na kraju ispis System.out.println(i), dobit ćemo ispis 0 1 3 4 |
| return | Služi nam da se vrati određena vrijednost iz metode, kada se izvrši sav kod iza return-a se ne izvodi. |
| **Kontrola toka: nepravilnosti** | |
| try – catch | Pokušamo izvršiti određeni kod, pa ako se pojave greške imamo catch nakon toga da „uhvati“ greške i izvrši kod u slučaju greške. |
| try – catch – finally | Isti princip kao try-catch ali kao default u switch case-u finnaly se uvijek izvodi, neovisno o ishocu try-catch-a. |
| try – finally | Pokušava se provesti određeni kod, i neovisno o njegovom ishodu nakon toga se izvrši finally. |
| throw | Služi kako bi se deklarirala iznimka koja bi se mogla dogoditi tijekom izvođenja programa. |
| **Ključne riječi** | |
| abstract | Služi za kreiranje apstraktnih klasa i metoda, klase ne mogu bit instancirane, nego samo nasljeđivane. |
| assert | Provjerava pretpostavke u programu, ako rezultat na kraju bude false, onda dolazi do Assertion error. |
| boolean | To je tip podatka koji može imati samo 2 stanja, true ili false. |
| break | Služi kako bi se petlja prekinula, najčešće se koristi u switch case-u. |
| byte | Ima raspon u [-127,128], to je 8-bitni tip podatka |
| case | To je blok koda u switch-u koji će se izvesti ovisno o tome odgvara li predani paramterat tom case-u. |
| catch | To je dio try-catch-a koji hvata grešku ako se pojavi u try bloku. |
| char | To je tip podatka koji sadrži samo jedan znak. |
| class | Ključna riječ pri kreiraju klase. |
| const | Const nam znači da se varijabla nakon deklaracije neće moći više mjenjati. |
| continue | Služi nam za preskakanje iteracije, npr ako imam uvjet (int i = 0; i < 5; i++) i if(i==2) continue, i na kraju ispis System.out.println(i), dobit ćemo ispis 0 1 3 4. |
| default | Dio je switch-case-a i uvijek se izvodi ako niti jedan case nije bio zadovoljen. |
| do | Prvi dio do-while petlje, uvijek se izvrši barem jednom |
| double | Tip podatka koji služi za spremanje decmalnog broja s puno decimala(do 15). |
| else | Dio if-else-a, izvršava se ako uvijek u if-u nije zadovoljen. |
| enum | Enum je posebna „klasa“ koja predstavlja grupu konstanti, tj. varijabli koje imaju const kljucnu riječ i ne mogu biti promjenjene |
| extends | To je ključna riječ za nasljeđivanje klase. |
| final | To je atribut za klase, atribute i metode radi kojeg se one više ne mogu mjenjati. |
| finally | Dio try-catch-finally i uvijek se izvodi neovisno o ishodu prije njega. |
| float | Isto kao double sprema decimale, ali samo do 6-7 decimali. |
| for | To je petlja u kojoj imamo iterator u kojemu imamo zadano koliko će se puta petlja izvrtiti, kada uvjeti u iteratoru nisu više zadovoljeni izalzi se iz petlje, npr: (int i = 0; i < 10; i++), for s ovakvim iteratorom će se izvrtiti 10 puta dok je uvjet zadovoljen, nakon toga se izlazi iz petlje. |
| goto | Naredba koja nas vodi na neki drugi dio programa, ali java ga ne podrzava. |
| if | Zadaju se uvjeti, ako su oni zadovoljeni izvršava se određena radnja |
| implements | Koristi se za nasljeđivanje interface-a. |
| import | Koristi se da uvođenje dodatnih paketa, klasa i interface-a. |
| instanceof | Provjerava je li objekt instanca neke klase ili ne. |
| int | Služi za pohranjivanje cijelobrojnih vrijednosti, raspon od 32-bita |
| interface | Opisuje nam što klasa koja ga nasljeđuje treba raditi, ali klasa mora implementirati sve metode i dati im funkcionalnost. |
| long | Isto kao int za cjelobrojne podatke, ali ima duplo veći raspon |
| native | To nam odmah daje do znanja da je metoda izvorno implementirana u JNI (Java Native Inerface) |
| new | Ovaj operator nam služi za kreiranje novih objekata. |
| package | Služi za grupiranje više klasa u jedan paket. Klase mogu nositi isto ime ako su u drugim paketima. |
| private | On nam ograničava pristup atrbutima i meotdama, jer su vidljivi samo unutar te klase. |
| protected | Isto kao i private, ali atributi i metode su vidljive svima unutar tog paketa. |
| public | Ne ograničava pristup atributima i metodama, nego su vidljivi svim klasama. |
| return | Služi nam da se vrati određena vrijednost iz metode, kada se izvrši sav kod iza return-a se ne izvodi. |
| short | Kao int to je cijelobrojni podatak, ali duplo manjeg raspona (16bita). |
| static | Modifikator za metode i atrbute, pristupiti im se može bez instanciranja objekta klase. |
| super | Upućuje nam na superklasu (roditeljsku) objekta. Najviše se koristi kako bi se raspoznala razlika između podklase i superklase istog imena. |
| switch | Dio switch-case-a, predaje mu se određeni argument i na temelju njega se izvodi određeni „case“. |
| synchronized | Kotrolira pritup za više dretvi, tj. oraginizira pristup resursima kako nebi došlo do prekapanja pto dovodi do nepotupunih rezultata. |
| this | Referita na trenutni objekt unutar metode. |
| throw | Služi za „bacanje“ pogrešaka, većinom su to prilagođene pogreške. |
| throws | Odnosi se na to da metoda može „baciti“ pogrešku. |
| transient | Ključna riječ koja označava varijablu da ne bude serijalizirana. |
| try | Služi da se pokuša izvesti određeni kod, nakon njega dolazi catch/finnaly. |
| void | To nam govori da metode void tipa nemaju nikakvu povratnu vrijednost. |
| volatile | Volatile varijabla mora biti spremljana u glavu memoriju. |
| while | Dok je uvjet zadovoljen kod se izvršava, ali ako uvjet odmah nije zadovoljen kod u while-u se neće niti jednom izvršiti. |
| false | Jedna od dvije boolean vrijednosti. |
| null | Može se dodjeliti svim ne primitivnim tipovima podataka. |
| true | Jedna od dvije boolean vrijednosti. |
| **SQL** | |
| **Općenito** | |
| table | Može sadržavati razne tipove podataka, predstavlja jedan objekt unutar baze podataka. |
| column | To je jedan stupac u tablici. |
| view | Služi da se korisniku prikažu podaci iz više tablica, organizirani u jednu tablicu. |
| index | Koriste se za brzo dohvaćanje, upite i pretragu podataka, korisnici ih ne vide. |
| constraint | Služe nam da zadamo pravila u tablici, koristi se s alter table. |
| stored procedure | To je već napisan kod koji se može koristiti više puta. |
| trigger | To je poseban tip spremljenih procesa koji se automatski pokrene na određene akcije kao npr. Update i Insert. |
| create | Koristi se da se stvori nova tablica u bazi. |
| drop | Koristi se za brisanje tablice ili čak cijele baze. |
| select | Služi za odabiranje podataka iz baze. |
| insert | Koriste se da se unesu podaci u tablicu. |
| update | Korisi se da se ažurira tablica, uz određene uvjete. |
| delete | Koriste se pri brisanju podataka iz tablice, mogu se obrisati sv ili samo određeni ( uz uvjet). |
| commit | Koristi se da se primjeni trajna promjena iz transakcije na bazu. |
| inner/outer/left/right join - objasniti | Inner – odabire zapise iz obje tablice koji imaju podudarajuće podatke  Outer – odabire sve zapise koji imaju vrijesnot u bar jednoj tablici  Left – odabire sve zapise iz lijeve tablice, a zapise iz desne  Right – odabire sve zapise iz dense tablice, a zapise iz lijeve |
| **Vrste relacija između tablica** | |
| 1 – 1 | Rijetko se koristi, to znači da jednom entitetnu može pripradati samo jedan entitet, npr. osoba i OIB. |
| 1 – više | Jednom entietnu može pripadati više entietnu, npr. jedna majka može imati više djece. |
| više - više | Više entiteta može imati više entiteta, npr. više osoba može koristiti više auta u obitelji, i svaki od tih auta je korišten od više osoba. |
| **SVN** | |
| **Općenito** | |
| Check-out | Služi da kopira datoteke iz SVN repozitroija u trenutni projekt. |
| Commit | Spremaju se promjene u repozitoriji. |
| trunk | To je glavno područje za razvoj. |
| branch | Provodi se paralelno uz „trunk“ direktoriji. „Branch“ kopira „trunk“ i dopušta da se na njemu rade promjene, kada su promjene stabilne, spoje se nazad. |
| **UNIX** | |
| **Osnovne naredbe** | |
| Ls | Izlistava nam datoteke u trenutnom direktoriju. |
| Cd | Služi da se promjeni trenutni direktoriji. |
| Mkdir | Služi da se stvori novi direktoriji. |
| Grep | Služi za pretraživanje stringova ili znakova u datotekama. |
| Cp | Služi za kopiranje datoteka ili direktorija. |
| Rm | Služi za brisanje datoteka ili direkotrija. |
| Mv | Služi za pomicanje jedne ili više datoteke/direktorija. |
| Rmdir | Služi za uklanjanje praznih direktorija. |
| Touch | Služi za kreiranje direktorija/datoteka, a ako postoje onda ih ažurira. |
| Chmod | Služi da se promjeni pravo pristupa datotekama. |
| Chown | Služi da se promjeni vlasnik za određenu datoteku/direktoriji. |
| Man | Služi da nam se otvore upute za korištenje. |
| Ps | Služi da se ispišu informacije o trenutno pokrenutim procesima. |
| Kill | Služi da se ugase procesi. |
| Kako izlistati stanje servisa | service --status-all |
| Gdje se definiraju mount pointi da budu perzistentni | /etc/fstab |
| Vrste distribucija | Linux, OS x, Solaris, OpenSolaris, OpenServer, Minix |
| Koja je zadnja verzija referentnog kernela | 5.7 |
| **Specijalni direktoriji** | |
| / | To nam predstavlja put root direktorija. |
| . | Trenutni direktorij. |
| .. | Direktoriji prije tj. roditeljski direktoriji. |
| ~ | Početni „home“ direktoriji. |
| **Putanje** | |
| Apsolutna putanja | To je putanja od root direktorija „/“ sve do datoteke ili direkotirja kojem pristupamo. |
| Relativna putanja | To je putanja od direktorija u kojemu se trenutno nalazimo. |
| **Osnovni preglednici** | |
| Vrste osnovnih preglednika | Google Chrome, Safari, Microsoft Edge, Mozilla Firefox... |
| Naredbe u „less“ pregledniku | Neke od njih su: Strijelica dole, Enter, e ili j – pomakne se naprijed jednu liniju  Strijelica gore, y ili k – pomakni se nazad jednu liniju  Space ili f – naprijed stranicu  b – stranicu nazad  h – prikaže pomoć |
| **vi editor** | |
| Uređivanje teksta | Neke od komandi su: dd – briše liniju  dw – briše riječ  cw – promijeni riječ |
| Snimanje promjena | :w – snime se promjene, ali se nastavi s uređivanjem  :wq(ZZ) – snime se promjene, ali se izađe iz editoa |
| Brisanje retka | Uz pomoć esc moramo „osloboditi“ pokzivać, njega stavimo na red koji želimo brisati i unesemo „dd“ komandu. |
| Pretraga teksta i navigacija (next/previous match) | N – idemo po tekstu unaprijed dok se ne pojavi string  n- idemo po tekstu unazad dok se ne pojavi string  /tekst – traži string po tekstu unaprijed  ?tekst – traži string po tekstu unaprijed |
| **General know-how** | |
| SOAP | Simple Object Access Protocl, to je XML protocol za razmjenu informacija između računala |
| RegExp | Pretražuje stringove putem parametara. |
| Node-red | To je alat kojeg je proizveo IBM i služi da poveže API, hardware i online servise. |
| GMT / UTC / CEST | Vremenske zone. |
| RabbitMQ | Softvare za posredovanje porukama otvorenog koda. |
| MosqitoMQ | Sličan princip kao RabbitMQ, ali se on fokusira na uređaje niske potrošnje u područijima slabog signala. |
| Cassandra | To je NoSQL sustav za upravljenjm bazom podatka. |
| NoSQL / NewSQL | To su baze podataka, mogu radtiti s većom količinom podataka, i podaci se spremaju drugačije od realacijskih baza podataka. |
| ETSI | Europski Institut Za Telkomunikacijske Strandarde |
| 3GPP | To je krovni pojam za brojne organizacije za standardizaciju razvoja protokola za mobilne tlekomunikacije. |
| LTE | Bežična Telekomunikacijska Tehnologija kojoj pripada četvrta generacija(4G) |
| 5G | To je peta generacija mobilne mreže koja nasljeđuje 4G, može biti i do 1000 puta brža. |
| Što je podatak, a što informacija | Podatak je jednostavna neobrađena izolirana misaona činjenica koja ima neko znaćenje  Informacija je rezultat analize i organizacije podatak ana način da daje novo znanje korisniku |
| Što je REGEXP, napisati regexp za IPv4 adresu i objasniti ga | REGEXP je regularni izraz.  IPv4 - ^(?:[0-9]{1-3}\.}{3}[0-9]{1-3}$  Provjerava brojeve 3 puta odjene točkom |
| Što je RFC | Request for Comments, opsuje nam inovacnije u internetu, standarde na int i metode istraživanja. |
| Kojim RFCom je opisan HTTP protokol (v 1.1), kojim HTTP2? | HTTP v1.1 je opisan s RFC 2616  HTTP v2 je opisan RFC 7540 |
| JSON | To je otvoreni standardni format datoteke i format za razmjenu podatka između web aplikacija. |
| REST | Representational State Transfer je framework koji olakšava stvaranje web servisa. |
| WS | To je oznaka objekta koji je predstavljen s WebSocketom, to nam olakšava povezivanje na server te slanje i priamnje podataka. |
| FYI | For Your Information |
| IIRC, IMHO | If I Remember Correctly  In My Humbe Opinion |