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 | Osnova objektno-orijentiranog programa; instanca klase je objekt; varijable definirane unutar klase su dostupne svakom objektu te klase; ponašanje klase opisuju metode |
| Objekt | Objekt je instanca klase; ima pristup varijablama i metodama te klase (primjer:  klasa Životinja, objekt te klase može biti pas, mačka, jelen, konj…) |
| Metoda | Metoda opisuje ponašanje neke klase; funkcionalni i reciklabilni blok koda; ovisno o povratnu tipu može, a ne mora vraćati vrijednost, te može, a ne mora primati argumente |
| Varijabla | Podatak (ovisno o tipu) zapisan u memoriji (String, Intgere, složeniji tip podataka…) |
| Sučelje | Implementira se od strane klase; ima svoje varijable, metode, te može imati zadanu (default) metode; klasa mora implementirati sve što sučelje opisuje |
| Nasljeđivanje | Klasa može naslijediti dio ili pak cijelu strukturu i ponašanja druge klase |
| Overload | Odnosi se na klasne metode istog imena, no različitih argumenata |
| API | Služi pri komunikaciji između više aplikacija; poboljšava kvalitetu života programa već implementiranim rješenjima; javni (public) API se može koristiti uz prethodnu autorizaciju; primjer, Twitter API koji omogućava integriranje Vašeg Twitter „feeda“ u razne desktop i web aplikacije |
| JDK | Java Development Kit; sadrži razvojne alate kao i JRE; služi razvoju Java aplikacija |
| JRE | Java Runtime Environment; osigurava runtime okruženje; „stvara“ JVM |
| JVM | Java Virtual Machine; apstrakcija; izvršava „bytecode“ |
| jar | Java arhiva datoteka; ekstenzija za istu; čuva fileovi potrebne za pokretanje neke aplikacije |
| j2se | Pruža bazično iskustvo za razvoj standardnih aplikacija; razvoj „samostalnih desktop aplikacija (eng. „applets“)“ |
| j2ee | Ima sve funkcionalnosti „J2SE“; koristi se za izradu „poduzetničkih (eng. „enterprise“) aplikacija“; dodatno funkcionalnosti su EJB, JSP, Servelts i XML tehnologijom |
| OpenJDK vs OracleJDK | OpenJDK je „open source“ sa [GNU General Public License, version 2](https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html), dok „OracleJDK“ koristi komercijalnu licencu („pay to use“, „subscription“) |
| Maven | Razvojni okvir (engl. „framework“); repozitorij koji sadrži mnoštvo knjižnica koje pomažu, doprinose i poboljšavaju razvoju Java aplikacija |
| Ant | Programski alat za automatizaciju izrade aplikacija; nastao pod Apache Tomcatom ranih 2000.; zamjena za Unix-ov popularni „Make“; zamjena za „Makefile“ format je XML |
| Gradle | Slično kao Ant, no koristi i Maven; podržava izgradnju više projekata; osnovni alat za izradu Andorid OS aplikacija |
| Koja su prva 4 byte-a svake klase (class file-a) u hex formatu i zašto? | CAFEBABE(16) = 3405691582(10); magični broj; James Gosling je trebao par „magičnih brojeva“, te se odlučio za CAFEDEAD(16); zašto CAFEBABE(16)? [Cijela priča](https://en.wikipedia.org/wiki/Java_class_file#Magic_Number). |
| **Konvencija naziva** | |
| Naziv klasa | Veliko početno slovo; imenica; npr. Auto, Osoba, Računalo… |
| Naziv metoda | Malo početno slovo; glagol; npr. vozi();, hodaj();, upaliSe();…; u slučaju više riječi se koristi „camel case -> camelCase“ |
| Naziv varijabli | Malo početno slovo; također „camel case“; prviIzbor;, zbroj;, brojSlova;… |
| **Tipovi podataka** | |
| Primitivni tipovi | Sprema određenu vrstu podataka; osnovni tipovi su: byte, short, int, long, float, double, char, boolean |
| Referencirajući tipovi | Tipovi koji se tvore od drugih primitivnih tipova; npr. String (od char tipova) |
| **Operatori** | |
| Aritmetički | Ima ih 12: +, -, \*, /, %, ++, --, +=, -=, \*=, /=, %= |
| Relacijski | Ima ih 6: ==, !=, >, <, >=, <= |
| Operatori uvjeta | Jedan (ternarni operator): ?: |
| Operatori pridruživanja | Ima ih 6: =, +=, -=, \*=, /=, %= |
| **Kontrola toka: odlučivanje** | |
| if | Ako; zadaje se i provjerava uvjet, ukoliko je zadovoljen, izvodi se blok koda |
| if – else | Ako, inače; ako je uvjet zadovoljen, izvedi blok koda (od if), inače izvedi blok koda (od else) |
| if – else if – else | Ako, inače ako, inače; nizanje gornja dva; else if, za razliku od elsa, prima argument |
| switch – case | Sklopka, slučaj(evi); kroz slučajeve se provodi argument predan sklopki; ukoliko je u jednom slučaju zadovoljen, izvodi se blok unutar tog slučaja; slično nizanju if-else if-else-ova; switch ima default koji se izvodi ako ni jedan slučaj nije zadovoljen |
| **Kontrola toka: petlje** | |
| for | Petlja; zadaje se početna vrijednost iteratora, krajnja vrijednost i korak; ovisno o kodu unutar petlje, kod se izvodi onoliko puta koliko je specificirano u iteratoru; npr.  for (int i = 0; i < 10; i++) { kod }, kod se izvodi deset puta; iterator ide od nula do 9 (kada je iterator 10, ne ulazi u for) te se povećava za jedan (korak i++) |
| while | Dok; dok je uvjet zadovoljen, izvršavaj kod; kod se ne mora izvršiti ni jednom; npr.  while (isDone) { kod }; u slučaju da je isDone = false; kod se ne izvršava ni jednom; |
| do – while | Radi, dok; slično kao dok, no kod se uvijek izvrši barem jednom; npr. (sličan kao dok)  do { kod } while (isDone); iako je isDone = false; kod se izvrši jedno, te se provjerava uvjet |
| **Kontrola toka: bezuvjetni skok** | |
| break | Prekid; prekida se izvođenje petlji |
| continue | Pri iteraciji, kod one vrijednosti iteracije kod koje je postavljen, iteracija se preskače;  for (int i = 1; i <= 5; i++) { if (i == 3) continue; System.out.println(i); } dobijemo ispis 1 2 4 5 |
| return | Povrat vrijednosti iz metode; kod iza returna se ne izvodi; ovisno o metodi, vraća primitivni ili složeni tip podatka |
| **Kontrola toka: nepravilnosti** | |
| try – catch | Pokušaj i uhvati (error); pokušavamo izvesti kod, ukoliko se pojavi pogreška, zbrinemo ju |
| try – catch – finally | Isto kao try-catch; no finally se uvijek izvršava neovisno o ishodima try-catcha |
| try – finally | Finally se u ovom slučaju uvijek izvršava |
| throw | Većinom služi za „bacanje“ prilagođenih (od strane programemra) pogrešaka |
| **Ključne riječi** | |
| abstract | Abstract class; može biti samo naslijeđena, ne i instancirana; može služiti kao „opisna klasa“, sadržavati djelomičan opis koji onda klase koje ju nasljeđuju proširuju |
| assert | Koristi se kako bi se testirao program; ukoliko uvjet nije zadovoljen, dolazi do „AssertionError-a“ |
| boolean | Tip podatka koji može biti jedan od dva stanja, true ili false; npr.  System.out.println(9 > 10); - false  System.out.println(9 < 10); - true |
| break | Prekid; prekida se izvođenje petlji |
| byte | 8bitni tip podatka; seže u rasponu [-127, 128] |
| case | Blok kod koji se izvodi (ili ne izvodi) ovisno o uvjetu u „switch“ |
| catch | Ukoliko se pojavi pogreška kod „try“ izvođenja, „catch“ hvata tu pogrešku i dozvoljava nam da izvedemo kod, tako da zbrinemo pogrešku |
| char | Tip podatka koji može sadržavati samo jedan znak (npr. slovo ili broj) |
| class | Tip podatka koji koristimo kada kreiramo klasu (npr. public class Auto() {} ) |
| const | Const varijabla ne bi trebala biti mijenjana nakon deklaracije |
| continue | Pri iteraciji, kod one vrijednosti iteracije kod koje je postavljen, iteracija se preskače;  for (int i = 1; i <= 5; i++) { if (i == 3) continue; System.out.println(i); } dobijemo ispis 1 2 4 5 |
| default | U „switch“ služi da se izvede blok koda ukoliko ni jedan slučaj ne zadovoljava uvjet |
| do | Čini; prvi dio do-while petlje; izvodi blok koda unutar sebe uvijek barem jednom |
| double | Tip podatka za spremanje decimalnog broja (do 15 decimala); visoka preciznost |
| else | Inače; dio if-else; izvršava se ukoliko uvjet (argument) u if-u nije zadovoljen |
| enum | Deklarira se kao nepromjenjivi tip; enum je ključna riječ, deklariraju se konstante |
| Extends | Ključna riječ za nasljeđivanje klasa; npr. Pas exteneds Životinja |
| final | Final varijabli se vrijednost ne može mijenjati; ukoliko pokušamo, dobijemo „compile time error“ |
| finally | Kod try-catcha se uvijek izvodi, ne ovisi o ishodu sitih |
| float | Slično kao double, služi za decimalne brojeve (6 do 7 decimala preciznosti) |
| for | Petlja; zadaje se početna vrijednost iteratora, krajnja vrijednost i korak; ovisno o kodu unutar petlje, kod se izvodi onoliko puta koliko je specificirano u iteratoru; npr.  for (int i = 0; i < 10; i++) { kod }, kod se izvodi deset puta; iterator ide od nula do 9 (kada je iterator 10, ne ulazi u for) te se povećava za jedan (korak i++) |
| goto | Skok na drugi dio programa; ne koristi se jer ometa preglednost programa i dolazi do bug-ova |
| if | Ako; zadaje se i provjerava uvjet, ukoliko je zadovoljen, izvodi se blok koda |
| implements | Koristi se za nasljeđivanje „implementiranje“ interfacea |
| import | Uvoz; dodavanje dodatnih paketa, klasa i interfacea koji su nam potrebni |
| instanceof | Provjerava je li objekt instanca određene klase ili interfacea |
| int | Cjelobrojni tip podatka; uobičajeno 32bita u rasponu [-2147483648, 2147483647] |
| interface | Opisuje što klasa (koja nasljeđuje) radi, ali ne i kako; klasa mora implementirati (i overrideati) sve metode |
| long | Cjelobrojni tip podatka; uobičajeno 64bita u rasponu [-9223372036854775808, 9223372036854775807] |
| native | U Javi označava metoda izvorno implementiranu u JNI |
| new | Koristimo pri stvaranju instance klase; memorija za objekt se dodijeli, te se vraća referenca na tu memoriju |
| package | Više klasa grupiramo u jedan paket; čist i održiv kod; možemo koristiti ista imena metoda u različitim paketima (što inače ne možemo u istom paketu) |
| private | Modifikator pristupa; koristi se za atribute, metode i konstruktore; vidljivo samo unutar klase |
| protected | Modifikator pristupa; koristi se za atribute, metode i konstruktore; vidljivo unutar paketa i svim podklasama |
| public | Modifikator pristupa; koristi se za atribute, metode i konstruktore; vidljivo svim klasama |
| return | Povrat vrijednosti iz metode; kod iza returna se ne izvodi; ovisno o metodi, vraća primitivni ili složeni tip podatka |
| short | Cjelobrojni podatak; uobičajeno 16bita u rasponu [-32768, 32767] |
| static | Modifikator koji se koristi za metode i atribute; možemo im pristupiti bez instanciranja objekta klase |
| super | Upućuje na objekt roditeljske klase |
| switch | Sklopka; kroz slučajeve se provodi argument predan sklopki; slično nizanju if-else if-else-ova |
| synchronized | Kod višedretvenosti služi za sprječavanje preklapanja izvršavanja metoda |
| this | Referira se na trenutan objekt unutar metode ili konstruktora; koristi se, najčešće, da bi se izbjegli konflikti; npr.  public Auto(String naziv) { this.naziv = naziv } |
| throw | Većinom služi za „bacanje“ prilagođenih (od strane programemra) pogrešaka |
| throws | Govori da metoda može „throwati“ neke od navedenih pogrešaka |
| transient | Kod serijalizacije objekata, ključna riječ sprječava istu |
| try | Služi za pokušaj izvođenja određenog dijela koda; slijedi ga catch i / ili finally |
| void | Metode sa void tipom ne vraćaju nikakvu vrijednost |
| volatile | Govori da varijabla mora biti spremljena u glavnu memoriju |
| while | Dok; dok je uvjet zadovoljen, izvršavaj kod; kod se ne mora izvršiti ni jednom; npr.  while (isDone) { kod }; u slučaju da je isDone = false; kod se ne izvršava ni jednom; |
| false | Netočno; boolean vrijednost |
| null | Može se dodijeliti svim ne-primitivnim tipovima podataka kako bi se pokazalo da ta varijabla ne upućuje ni na jedan objekt (ili nešto drugo) |
| true | Točno; boolean vrijednost |
| **SQL** | |
| **Općenito** | |
| table | Tablica; jedan objekt unutar baze podataka koji može sadržavati razne tipove podataka |
| column | Jedan stupac u tablici; npr. jedan stupac iz tablice „Osoba“ sadržava sve informacije definirane u tablici o toj jednoj osobi (ovisno o uvjetu) |
| view | Ograničen pogled na tablicu / tablice; prikazujemo korisniku podatke iz više tablica kao jednu |
| index | Korisnici ne vide indekse; koriste se za brzo dohvaćanje, pretrage i upite |
| constraint | Koristi se za određivanje pravila unutar tablice; pri samom kreiranju tablice ili nakon (sa „alter table“) |
| stored procedure | Već pripremljeni kod koji možemo koristiti iznova ukoliko je potrebno |
| trigger | Okidač; pokreće se na određenu akciju (INSERT, UPDATE, DELETE) |
| create | Ključna riječ pri kreiranju različitih dijelova baze; npr. create table <imetablice>; create view <imeviewa> |
| drop | Drop se može koristiti kako bise obrisala cijela baza ili samo jedna tablica |
| select | Koristi se za odabir podataka iz baze; podaci se spremaju kao set rezultata (tablica) |
| insert | Koristimo pri unošenju podataka u tablicu, sa ili bez navođenja vrijednosti koje unosimo |
| update | Koristimo pri ažuriranju tablica uz određene uvjete |
| delete | Koristimo pri brisanju određenih (ili svih) podataka iz tablice (redova) |
| commit | Koristi se u transakcijama; primjenjuje se trajna promjena iz transakcije na bazu |
| inner/outer/left/right join - objasniti | Inner – odaberi zapise koji imaju podudarajuće podatke u obje tablice  Outer – odaberi sve zapise koji imaju vrijednost u barem jednoj od tablica  Left – odaberi sve zapise iz lijeve, i odgovarajuće zapise iz desne  RIght – odaberi sve zapise iz desne, i odgovarajuće zapise iz lijeve |
| **Vrste relacija između tablica** | |
| 1 – 1 | Veza jedan na jedan; rijetko korištena; jedan entitet može pripadati SAMO jednom entitetu; npr. osoba i osobni dokument (osobna iskaznica, vozačka dozvola, x-ica…) |
| 1 – više | Veza jedan na više; jedan entitet ima više pripadajućih entiteta; npr. osoba može imati više mobilnih telefona, automobila, nekretnina… |
| više - više | Veza više na više; jedna osoba može imati više mobilnih telefona, te svaki od tih mobilnih telefona može pripadati više osoba (za izvedbu se koristi posredni entitet) |
| **SVN** | |
| **Općenito** | |
| Check-out | Pristup kopiji repozitorija; kopiranje SVN repozitorija u trenutnu kopiju projekta |
| Commit | Spremanje promjena u repozitorij; dobra praksa je dodati „-m“ i poruku o promjenama koje su napravljene u trenutnoj verziji (koju šaljemo / spremamo) |
| trunk | Glavna linija proizvodnje na SVN-u |
| branch | Dio na kojemu se radi; mogu se spremati razne verzije, bug-fixing, work-in-progress… |
| **UNIX** | |
| **Osnovne naredbe** | |
| Ls | Trenutni sadržaj direktorija u kojemu se nalazimo („list“) |
| Cd | „Kretanje“ po direktoriju („change directory“) |
| Mkdir | Kreiranje novog direktorija („make directory“) |
| Grep | Pretraživanje određenog teksta (zadanog parametrima) u datoteci |
| Cp | Kopiranje dokumenata i direktorija |
| Rm | Ovisno o OS-u, uklanjanje dokumenata, direktorija i simboličnih poveznica sa datoteka |
| Mv | Prebacivanje (micanje) jednog ili više direktorija i / ili datoteka; kod micanja na isto mjesto dolazi do promjene imena („rename“) |
| Rmdir | Na različitim OS-ovima uklanjanje praznih direktorija |
| Touch | Ažuriranje datoteke / direktorija ukoliko postoji; ako ne postoji, kreiranje |
| Chmod | Promjena razine pristupa određenoj skupini korisnika ili korisniku |
| Chown | Promjena vlasnika („ownera“) određenog direktorija ili datoteke |
| Man | Otvara upute za korištenje |
| Ps | Prikaz informacija o procesima koji su trenutno pokrenuti |
| Kill | „Ubija“, tj. gasi određene procese |
| Kako izlistati stanje servisa | service --status-all |
| Gdje se definiraju mount pointi da budu perzistentni | /etc/fstab |
| Vrste distribucija | Minix, Linux, Mac OS X, BSD (FreeBSD, DragonflyBSD, NetBSD, OpenBSD), AIX, Solaris, OpenSolaris, HP-UX, OpenServer |
| Koja je zadnja verzija referentnog kernela | Radna verzija: 5.16.8; 08.02.2022.  „Preview“: 5.17-rc3 |
| **Specijalni direktoriji** | |
| / | Najviša točka u sustavu; sastoji se od svih ostalih direktorija |
| . | Trenutni direktorij |
| .. | „Roditelj“ trenutnog direktorija (ekvivalentno „Back“ na Windows-u) |
| ~ | „Home“ direktorij na sustavu ($HOME varijabla) |
| **Putanje** | |
| Apsolutna putanja | Putanja od „/“ sve do datoteke / direktorija kojem želimo pristupiti |
| Relativna putanja | Putanja od trenutnog direktorija; npr.  Nalazimo se u /home/hrvoje/ i hoćemo ući u „photos“ direktorij, dovoljno je napisati „cd photos“ |
| **Osnovni preglednici** | |
| Vrste osnovnih preglednika | Najpopularniji su Google Chrome, Internet Explorer, Safari, Microsoft Edge i Mozilla Firefox |
| Naredbe u „less“ pregledniku | Strelica dolje, Enter, e ili j – naprijed jednu liniju  Strelica gore, y ili k – nazad jednu liniju  Razmak ili f – naprijed jednu stranicu  B – nazad jednu stranicu  Samo su neke od naredbi, ostale se mogu pronaći [ovdje](https://linuxize.com/post/less-command-in-linux/) |
| **vi editor** | |
| Uređivanje teksta | u – natrag zadnju promjenu  dd – obriši liniju  dw – obriši riječ  cw – promijeni riječ  Samo su neke od naredbi, ostale se mogu pronaći [ovdje](http://www.atmos.albany.edu/daes/atmclasses/atm350/vi_cheat_sheet.pdf) |
| Snimanje promjena | :w – snimi i nastavi uređivatei  :wq (ZZ) – snimi i izađi iz vi editora |
| Brisanje retka | Prvo „Esc“, potom staviti pokazivač na redak koji se želi obrisati, te unijeti „dd“ i potvrditi „Enterom“ |
| Pretraga teksta i navigacija (next/previous match) | /nekistring – traži tekst unaprijed za zadani string  ?nekistring – traži tekst unazad za zadani string  n – navigiramo po tekstu prema naprijed do pojave stringa  N – navigiramo po tekstu u nazad do pojave stringa |
| **General know-how** | |
| SOAP | Komunikacijski protokol između aplikacija koristeći HTTP |
| RegExp | Regularni izraz; pretražujemo stringove putem parametara |
| Node-red | IBM-ov alat kojim se povezuje API, hardware i online servisi |
| GMT / UTC / CEST | Oznake za razne vremenske zone |
| RabbitMQ | „Open source message broker“ |
| MosqitoMQ | Slično kao RabbitMQ; open source je; fokusira se na uređaje niske potrošnje u okolinama slabog signala |
| Cassandra | NoSQL baza podataka |
| NoSQL / NewSQL | Baze podataka; podaci se spremaju drukčije od relacijskih baza; brži pristup i rad sa većom količinom podataka |
| ETSI | Neovisna, neprofitna organizacija za standardizaciju u području informacija i komunikacija |
| 3GPP | „Umbrella term“ za brojne organizacije koje razvijaju protokole mobilne komunikacije |
| LTE | Bežična telekomunikacijska tehnologija |
| 5G | Peta generacija standarda za „broadband“; |
| Što je podatak, a što informacija | Podatak – jednostavna neobrađena izolirana misaona činjenica koja ima neko značenje  Informacija – rezultat analize i organizacije podataka na način da daje novo znanje primatelju |
| Što je REGEXP, napisati regexp za IPv4 adresu i objasniti ga | REGEXP (IPv4) - ^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$  Provjera brojeva 3 puta odvojena točkom; završava brojem |
| Što je RFC | Publikacija u seriji; opisuje standarde interneta, metode istraživanja i ponašanja, te inovacije na internetu |
| Kojim RFCom je opisan HTTP protokol (v 1.1), kojim HTTP2? | HTTP v1.1 je opisan RFC 2616, a HTTP v2 je opisan RFC 7540 |
| JSON | Format za razmjenu datoteka i podataka između web aplikacija |
| REST | REST je okvir („framework“) koji olakšava stvaranje web servisa |
| WS | Objekt predstavljen WebSocketom; olakšava nam povezivanje na server, te slanje i primanje podataka |
| FYI | For Your Information |
| IIRC, IMHO | IIRC – If I Recall (Remember) Correctly  IMHO – In My Humble Opinion |