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 opisuje svojstva i ponašanje objekta. Sadrži atribute i metode koje opisuju što ta klasa predstavlja, tj. koje vrijednosti sprema i koje funkcije može izvoditi. |
| Objekt | Objekt je instanca klase. |
| Metoda | Metoda je funkcija definirana unutar klase. |
| Varijabla | Varijabla se koristi za spremanje neke vrijednosti. Vrijednost koju sprema se može izmjenjivati |
| Sučelje | Sučelje opisuje funkcije/metode koje objekt/klasa mora sadržavati. |
| Nasljeđivanje | Nasljeđivanje podrazumijeva da jedna klasa (podklasa) može potpuno ili djelomično naslijediti svojstva i ponašanja neke druge klase (nadklase), tj. podklasa će sadržavati iste atribute i metode kao i nadklasa. |
| Overload | Overload se koristi kod definiranja metoda istih naziva, ali s različitim argumentima (ulaznim parametrima). Razlikuju se po broju i tipu argumenata. |
| API | API (Application Programming Interface) pruža aplikaciji sučelje prema nekom sistemu/serveru. Aplikacija šalje API-ju neki zahtjev, zatim će API proslijediti taj zahtjev sistemu/serveru i vratiti odgovor aplikaciji. |
| JDK | Java Development Kit. Razvojno okruženje pomoću kojeg se razvijaju Java aplikacije. |
| JRE | Java runtime environment. Skup komponenti za stvaranje i pokretanje Java aplikacije. JRE je dio JDK-a. |
| JVM | Java virtual machine. Interpretira i kompajlira Java kod. |
| jar | Java ARchive. Tip datoteke koji se temelji na ZIP datotekama. Koristi se za grupiranje više datoteka u jednu. |
| j2se | Java Platform, Standard Edition. Osnovna verzija Jave. Na njemu se temelje sva ostala izdanja. Uglavnom se koristi za razvoj Desktop aplikacija. |
| j2ee | Java Platform, Enterprise Edition. Koristi se u razvoju web aplikacija i servisa. Koristi HTML, CSS i JavaScript. |
| OpenJDK vs OracleJDK | |  |  | | --- | --- | | OracleJDK | OpenJDK | | * Novo izdanje svake 3 godine uz dugoročnu podršku. * Zahtjeva komercijalnu licencu * U potpunosti razvija Oracle | * Novo izdanje svakih 6 mjeseci i podška traje dok ne izađe novo izdanje * Open source * Razvija Oracle, OpenJDK i Java zajednica | |
| Maven | Alat za upravljanje Java projektima. Koristi se POM (Project Object Model) datoteka. Ona je XML datoteka koja sadrži sve informacije o projektu i njegovoj konfiguraciji. |
| Ant | Slično kao Maven. Ant je sličniji alatu dok je Maven framework. |
| Gradle | Open source alat za automatizaciju izgradnje aplikacija. |
| Koja su prva 4 byte-a svake klase (class file-a) u hex formatu i zašto? | 0xCAFEBABE. James Gosling je nazivao jedan lokal cafe dead zato što je tamo navodno u prošlosti nastupao rock bend Grateful Dead. Bili su mu potrebni brojevi za trajnu objektnu datoteku i za klase. Primijetio je da je CAFE DEAD heksadecimalni broj pa je to koristio za trajnu objektnu datoteku i genijalnom domišljatosti je smislio CAFE BABE za klase. |
| **Konvencija naziva** | |
| Naziv klasa | * Imenica * Opisuje što klasa predstavlja * Veliko prvo slovo svake riječi unutar naziva * Izbjegavati skraćenice osim ako su one poznatije od punog naziva (npr. URL). |
| Naziv metoda | * Glagol * Malo prvo slovo prve riječi, a svaka iduća riječ ima veliko prvo slovo |
| Naziv varijabli | * Malo prvo slovo prve riječi, a svaka iduća riječ ima veliko prvo slovo * Ne bi trebalo započinjati posebnim znakovima kao što su „\_“, „$“ i „&“ * Izbjegavati nazive od samo jednog slova |
| **Tipovi podataka** | |
| Primitivni tipovi | Primitivni tipovi podataka sadržavaju neku vrijednost, to su: int, byte, short, long, float, double, boolean i char. |
| Referencirajući tipovi | Referencijalni tipovi podataka sadržavaju referencu na neku vrijednost, to su sve ostale varijable osim onih navedenih iznad pod „Primitivni tipovi“ (najčešće objekti). Primjer reference: „Ime@56aa4b56“, „Ime“ je naziv varijable, a „56aa4b56“ njezin ID koji Java automatski stvori. |
| **Operatori** | |
| Aritmetički | Koristi se za izvođenje aritmetičkih operacija nad operandima.   * Zbrajanje „+“ * Oduzimanje „-“ * Množenje „\*“ * Dijeljenje „/“ * Modul (ostatak od dijeljenja, npr. 6 % 5 = 1) „%“ * Inkrement (povećava vrijednost za 1) „++“ * Dekrement (smanjuje vrijednost za 1) „--“ |
| Relacijski | Koristi se za provjeru odnosa između operanda.   * Jednako „==“ * Različito „!=“ * Manje „<“ * Veće „>“ * Manje ili jednako „<=“ * Veće ili jednako „>=“ |
| Operatori uvjeta | Koristi se za provjeru uvjeta između boolean vrijednosti i kao rezultat vraća boolean (true ili false).   * Logički i „&&“ * Logički ili „||“ * Uvjetni/ternarni operator „?=“   Ternarni operator: (izraz) ? (izraz istinit) : (izraz nije istinit) |
| Operatori pridruživanja | Koristi se za pridruživanje vrijednosti varijablama.   * Obično pridruživanje „=“ * Pridruživanje uz zbrajanje „+=“ * Pridruživanje uz oduzimanje „-=“ * Pridruživanje uz množenje „\*=“ * Pridruživanje uz dijeljenje „/=“ * Pridruživanje uz modul „%=“   Primjer za pridruživanje uz zbrajanje (odnosi se i na ostala pridruživanja uz aritmetičku operaciju): „a += b“ je isto kao da je napisano „a = a + b“. |
| **Kontrola toka: odlučivanje** | |
| if | Koristi se za definiranje uvjeta te ako je uvjet ispunjen izvršava se definirani kod.  if(uvjet)  {kod} |
| if – else | Ako je definirani uvjet if-a ispunjen izvršava se blok koda ispod njega, a ako nije izvršava se blok koda ispod else-a. Else uvijek mora imati prije sebe if, a if ne treba imati else nakon sebe.  if(uvjet)  {kod 1}  else  {kod 2} |
| if – else if – else | Korištenjem „else if“ može se definirati više od jednog uvjeta. Uvjeti se provjeravaju sekvencijalno. Prvo se provjerava if uvjet, pa else if uvjet ispod njega, itd.  if(uvjet)  {kod 1}  else if(uvjet)  {kod 2}  else  {kod 3} |
| switch – case | Switch - case je sličan kao if - else, ali za razliku od njega switch - case radi provjeru samo u ovisnosti o jednoj varijabli dok if unutar sebe može imati uvjet vezan za više varijabli.  switch(izraz)  case 1: kod 1;  break:  case 2: kod 2;  break;  default: kod 3;  break;  Ako je izraz jednak 1, onda izvrši kod 1, ako je izraz jednak 2 onda izvrši kod 2, a za sve ostalo izvrši kod 3. „Default“ je u ovom slučaju kao „else“. Potreno je koristiti „break“ nakon svakog izvršenog koda kako bi se izašlo iz switch-a. |
| **Kontrola toka: petlje** | |
| for | For petlja se sastoji od tri izraza i bloka koda ispod njega. Prvi izraz se izvodi samo jednom, drugi izraz definira uvjet za izvođenje bloka koda, treći izraz se izvršava svaki put nakon izvršavanja bloka koda.  for(izraz 1; izraz 2; izraz 2)  {kod} |
| while | While petlja provjerava zadani uvjet te ako je uvjet zadovoljen izvršava se blok koda. Skroz dok je uvjet zadovoljen blok koda će se ponovno izvršavati.  while(uvjet)  {kod} |
| do – while | Kod do - while blok koda se izvršava jedno i nakon toga se ponaša isto kao obični while (izvodi blok koda ponovno skroz dok je uvjet zadovoljen).  do{kod}  while(uvjet) |
| **Kontrola toka: bezuvjetni skok** | |
| break | Break zaustavlja izvođenje petlje u potpunosti. |
| continue | Continue preskače samo trenutnu iteraciju petlje. |
| return | Return završava izvođenje metode/funkcije. Također možemo vratiti neku vrijednost, npr. „return 1;“ će završiti izvođenje funkcije i vratiti vrijednost 1. |
| **Kontrola toka: nepravilnosti** | |
| try – catch | Koriste se za provjeru je li kod ima grešku i što učiniti ako ju ima. U „try“ blok se piše kod koji želimo testirati, a u „catch“ blok pišemo što želimo da se dogodi ako se pojavila greška. |
| try – catch – finally | Kao i „try - catch“, ali imamo dodatni blok koda „finally“. U „finally“ blok se piše kod koji će se neovisno o pojavi greške uvijek izvesti nakon „try - catch“ blokova. |
| try – finally | Prvo se izvodi „try“ blok, a zatim „finaly“ blok. Nije bitno je li došlo do greške, oba bloka koda će se izvršiti. |
| throw | Koristi se za rukovanje iznimkama. „throw“ baca iznimku iz neke metode ili bloka koda. Većinom se koristi unutar „try“ bloka. |
| **Ključne riječi** | |
| abstract | Koristi se za kreiranje apstraktne klase ili apstraktne metode. Ako klasa sadrži apstraktnu metodu onda i sama klasa mora biti apstraktna. Apstraktne klase se ne mogu instancirati, ali se mogu nasljeđivati. Apstraktna metoda se samo deklarira (nema implementaciju). |
| assert | Koristi se za provjeru je li neki uvjet istinit, ako nije onda se izbacuje greška. |
| boolean | Tip podatka koji može imati samo dvije vrijednosti, true ili false. |
| break | Break zaustavlja izvođenje petlje u potpunosti. |
| byte | Tip podataka koji se sastoji od 8 bitova. Može imati vrijednost od -128 do 127. |
| case | Koristi se unutar switch - case. Predstavlja jedan od mogućih slučajeva unutar switch bloka. |
| catch | Koristi se u kombinaciji s „try“ blokom (prvo ide „try“ blok, a nakon njega „catch“ blok). U „try“ blok se piše kod koji želimo testirati, a u „catch“ blok pišemo što želimo da se dogodi ako se pojavila greška. |
| char | Tip podataka koji predstavlja jedan znak. |
| class | Koristi se za defnirianje klase.  public class NazivKlase  {svojstva i metode} |
| const | Predstavlja konstantu. Jednom kada joj se definira vrijednost više se ne može mijenjati. |
| continue | Continue preskače samo trenutnu iteraciju petlje. |
| default | Koristi se unutar switch - case. Default blok se izvršava ako se nije izveo niti jedan od case slučajeva. |
| do | Koristi se u do - while. Predstavlja blok koda koji petlja izvodi. |
| double | Decimalni tip podataka veličine 8 bajta. |
| else | Koristi se u kombinaciji s if naredbom. Else blok se izvršava uvjet prijašnje navedene if naredbe nije ispunjen. |
| enum | Enum se koristi za spremanje liste konstanti. |
| extends | Extends se koristi kod nasljeđivanja klasa. |
| final | Final varijable su jednake konstantama, tj. ne može im se mijenjati vrijednost. Ako je metoda final onda ju ne možemo override-ati. Ako je klasa final onda ju ne možemo nasljeđivati. |
| finally | Koristi se u kombinaciji s „try“ ili „try - catch“. Izvodi se nakon „try“ i „catch“ blokova neovisno o tome je li došlo do greške ili ne. |
| float | Decimalni tip podataka veličine 4 bajta. |
| for | For petlja se sastoji od tri izraza i bloka koda ispod njega. Prvi izraz se izvodi samo jednom, drugi izraz definira uvjet za izvođenje bloka koda, treći izraz se izvršava svaki put nakon izvršavanja bloka koda.  for(izraz 1; izraz 2; izraz 2)  {kod} |
| goto | Koristi se za skok u programu. Omogućuje nastavak rada programa od navedene naredbe. |
| if | Koristi se za definiranje uvjeta te ako je uvjet ispunjen izvršava se definirani kod.  if(uvjet)  {kod} |
| implements | Implement se koristi kod implementiranja sučelja. |
| import | Koristi se za pristupanje nekom paketu i njegovim klasama unutar programa. |
| instanceof | Koristi se za testiranje je li objekt određenog tipa. Vraća „true“ ili „false“. |
| int | Cjelobrojni tip podataka veličine 4 bajta. |
| interface | Koristi se za definiranje sučelja koje sadrži samo deklaracije funkcija. Kada klasa implementira neko sučelje mora implementirati sve funkcije deklarirane sučeljem. |
| long | Cjelobrojni tip podataka veličine 8 bajta. |
| native | U Javi „native“ metode su Java metode koje imaju implementaciju napisanu u nekom drugom programskom jeziku. |
| new | Koristi se za kreiranje instance klase, tj. objekta. |
| package | Paket se koristi za grupiranje povezanih klasa. |
| private | Privatnom članu se može pristupiti samo unutar klase kojoj pripada. |
| protected | Protected članu se može pristupiti samo unutar paketa kojem pripada. |
| public | Public članu se može pristupiti bilo gdje. |
| return | Return završava izvođenje metode/funkcije. Također možemo vratiti neku vrijednost, npr. „return 1;“ će završiti izvođenje funkcije i vratiti vrijednost 1. |
| short | Cjelobrojni tip podataka veličine 2 bajta. |
| static | „static“ član klase ne pripada instanci klase, nego samoj klasi, tj. „static“ član je dijeljen između svih instanci te klase. |
| super | Referentna varijabla koja pokazuje na objekt roditeljsku klasu. Može se koristiti za pristup atributima i metodama roditeljske klase. |
| switch | Switch - case je sličan kao if - else, ali za razliku od njega switch - case radi provjeru samo u ovisnosti o jednoj varijabli dok if unutar sebe može imati uvjet vezan za više varijabli. |
| synchronized | Koristi se za sinkronizaciju kod pristupa dijeljenim podatcima iz različitih niti. Dijelu koji je označen kao „synchronized“ se može pristupiti samo iz jedne niti u bilo kojem trenutku. |
| this | Referenca na trenutni objekt. |
| throw | Koristi se za rukovanje iznimkama. „throw“ baca iznimku iz neke metode ili bloka koda. Većinom se koristi unutar „try“ bloka. |
| throws | Kao i „throw“, ali može se definirati više od jedne iznimke. |
| transient | „transient“ varijabla nije serijalizirana. |
| try | Koristi se u kombinaciji s „catch“ blokom (prvo ide „try“ blok, a nakon njega „catch“ blok). U „try“ blok se piše kod koji želimo testirati, a u „catch“ blok pišemo što želimo da se dogodi ako se pojavila greška. |
| void | Označuje da metoda nema povratni tip podatka. |
| volatile | Svako čitanje „volatile“ varijable će se čitati iz glavne memorije (neće se vrijednost spremati u cache memoriju pa od tamo čitati) i svaka izmjena će se odmah zapisivati u glavnu memoriju. |
| while | While petlja provjerava zadani uvjet te ako je uvjet zadovoljen izvršava se blok koda. Skroz dok je uvjet zadovoljen blok koda će se ponovno izvršavati. |
| false | Laž. Jedna od dvije mogućih „boolean“ vrijednosti. |
| null | Defaultna vrijednost svih referentnih tipova podataka. Pokazuje da referentni podatak ne pokazuje na nikakvu vrijednost, tj. da nije inicijaliziran. |
| true | Istina. Jedna od dvije mogućih „boolean“ vrijednosti. |
| **SQL** | |
| **Općenito** | |
| table | Tablica za spremanje podataka. Svaki stupac označava jednu vrstu podataka. Prvi red su nazivi/tipovi podataka koji se čuvaju u stupcima. Svaki red nakon prvog sadrži vrijednosti svakog stupca. |
| column | Svaki stupac označava jednu vrstu podataka. |
| view | Prikaz nekih ili svih polja jedne ili više tablica. |
| index | Koristi se za lakše dohvaćanje podataka iz baze podataka. Prave se indexi za brži pristup određenim podatcima koji se često pretražuju (uklanja potrebu za pisanje upita za dohvaćanje tih podataka). |
| constraint | Koristi se za postavljanje pravila za podatke unutar tablice. Npr. „NOT NULL“ znači da podatak ne može imati NULL vrijednost, „DEFAULT“ postavlja defaultnu vrijednost ako nije specificirana neka vrijednost. |
| stored procedure | Spremljeni SQL kod kako bi se mogao ponovno koristiti. |
| trigger | Pokreće se na neki događaj. Npr. možemo pokrenuti neki upit ako se neki podatak obrisao. |
| create | Koristi se za stvaranje nove baze podataka, tablice, view-a, procedure ili indeksa. |
| drop | Koristi se za brisanje baze podataka, tablice, view-a, stupca ili constraint-a. |
| select | Koristi se za dohvaćanje/odabir podataka iz baze podataka koji zadovoljavaju određene uvjete. |
| insert | Koristi se za umetanje novih redaka u tablicu. |
| update | Koristi se za ažuriranje već postojećih redaka u tablici. |
| delete | Koristi se za brisanje redaka iz tablice. |
| commit | Završava transakciju i trajno pohranjuje sve izmjene baze podataka. |
| inner/outer/left/right join - objasniti | JOIN - spaja tablice.  Tipovi JOIN-a:   * INNER - vraća podatke koji imaju iste vrijednosti u obje tablice. * OUTER - vraća sve podatke koji imaju vrijednost u bilo kojoj tablici. * LEFT - vraća sve podatke iz prve/lijeve tablice i podatke iz druge/desne tablice koji se podudaraju s podatcima iz lijeve tablice * RIGHT - vraća sve podatke iz druge/desne tablice i podatke iz prve/lijeve tablice koji se podudaraju s podatcima iz desne tablice |
| **Vrste relacija između tablica** | |
| 1 – 1 | Zapis iz prve tablice može imati vezu sa samo jednim zapisom u drugoj tablici i obrnuto. |
| 1 – više | Zapis iz prve tablice može imati vezu s više zapisa iz druge tablice, a zapis iz druge tablice može imati vezu samo s jednim zapisom iz prve tablice. |
| više - više | Zapis iz prve tablice može imati vezu s više zapisa iz druge tablice i zapis iz druge tablice može imati vezu s više zapisa iz prve tablice. |
| **SVN** | |
| **Općenito** | |
| Check-out | Naredba za kopiranje datoteka sa SVN repozitorija u radni direktorij. |
| Commit | Naredba za sprema izmjene koje smo napravili u SVN repozitorij (izmjene koje smo radili lokalno spremamo na server). |
| trunk | Glavno razvojno područje. Ovdje se sprema cijeli projekt/program. |
| branch | Područje za paralelni razvoj. Koristi se za razvoj različitih funkcionalnosti koje će se pridodati trunk-u. |
| **UNIX** | |
| **Osnovne naredbe** | |
| Ls | Ispisuje datoteke koje se nalaze unutar direktorija. |
| Cd | Mijenja trenutni direktorij. |
| Mkdir | Stvara novi direktorij. |
| Grep | Koristi se za pretraživanje sadržaja datoteke. |
| Cp | Kopira jedan ili više dokumenata na drugu lokaciju. |
| Rm | Briše dokumente ili direktorije. |
| Mv | Premješta ili mijenja naziv datoteka. |
| Rmdir | Briše prazne direktorije. |
| Touch | Mijenja vremenske oznake datoteke (vrijeme izmjene, vrijeme pristupa). |
| Chmod | Mijenja razinu pristupa (read, write, execute). |
| Chown | Mijenja vlasnika ili grupa vlasnika datoteke. |
| Man | Prikazuje priručnik za korištenje bilo koje komande. |
| Ps | Ispisuje sve trenutno pokrenute procese. |
| Kill | Prekida proces. |
| Kako izlistati stanje servisa |  |
| Gdje se definiraju mount pointi da budu perzistentni |  |
| Vrste distribucija |  |
| Koja je zadnja verzija referentnog kernela |  |
| **Specijalni direktoriji** | |
| / | Glavni direktorij, unutar njega se nalaze svi ostali direktoriji/datoteke. |
| . | Trenutni direktorij. |
| .. | Roditeljski direktori, tj. direktorij u kojem se nalazi trenutni direktorij. |
| ~ | Početni korisnički direktorij. |
| **Putanje** | |
| Apsolutna putanja | Puna putanja datoteke, sadrži sve direktorije od root direktorija do same datoteke. |
| Relativna putanja | Sadrži putanju od trenutnog direktorija do datoteke. |
| **Osnovni preglednici** | |
| Vrste osnovnih preglednika |  |
| Naredbe u „less“ pregledniku |  |
| **vi editor** | |
| Uređivanje teksta |  |
| Snimanje promjena |  |
| Brisanje retka |  |
| Pretraga teksta i navigacija (next/previous match) |  |
| **General know-how** | |
| SOAP | Simple Object Access Protocol. Komunikacijski protokol baziran na XML-u. Neovisan je o programskom jeziku. |
| RegExp | Regular Expression. Koristi se za opisivanje uzorka za pretraživanje niza znakova. |
| Node-red | Razvojni alat za vizualno programiranje. Koristi se za spajanje sklopovlja, API-ja i online usluga. |
| GMT / UTC / CEST | GMT (Greenwich Mean Time) - vremenska zona Londona.  UTC (Coordinated Universal Time Standard) - vremenski standard po kojem se definiraju vremenske zone.  CEST (Central European Summer Time) - vremenska zona UTC+2. |
| RabbitMQ | Aplikacija sa slanje poruka. |
| MosqitoMQ | Implementira MQTT protokol za slanje poruka. |
| Cassandra | Open source NoSQL baza podataka. |
| NoSQL / NewSQL | Baza podataka koja se NE sastoji od tablica. Svaki podatak je većinom spremljen kao JSON dokument. |
| ETSI | Neprofitna organizacija koja razvija telekomunikacijske standarde za korištenje u Europi. |
| 3GPP | Sastoji se od sedam organizacija za razvijanje telekomunikacijskih standarda. |
| LTE | Standard za bežičnu i širokopojasnu komunikaciju mobilnih uređaja. |
| 5G | Standard pete generacije za širokopojasne mobilne mreže. |
| Što je podatak, a što informacija | Podatak je sirova kolekcija nekih činjenica koje trebaju biti obrađene, dok je informacija obrađeni podatak. Informacija je korisna, podatak nije. |
| Što je REGEXP, napisati regexp za IPv4 adresu i objasniti ga |  |
| Što je RFC | Request for Comments. Tehnički dokument koji izdaje IETF (Internet Engineering Task Force). Koristi se za razvoj standardnog mrežnog protokola. |
| Kojim RFCom je opisan HTTP protokol (v 1.1), kojim HTTP2? | HTTP = RFC2616  HTTP2 = RFC7540 |
| JSON | JSON je objekt koji sadrži podatke u obliku parova (ključ i vrijednost). Sintaksa se zasniva na JavaScript-u. |
| REST | Representational state transfer. |
| WS | Web Service. Usluga koju pruža jedan uređaj drugome preko internetske veze. |
| FYI | For your information? |
| IIRC, IMHO | If I recall correctly? In my honest opinion? |

# Praktični dio

Riješite navedene zadatke vodeći računa o točnosti rješenja, čitljivosti i razumljivosti koda. Svi java programi koje budete kreirali tijekom priprema trebaju biti dio istog Eclipse projekta pod nazivom PripremaPrakse\_<ImePrezime> (kreira se unutar zadatka 02). Neka rješenje svakog zadatka bude u zasebnom java paketu. Pakete nazovite ovako:

hr.atos.praksa.<imeprezime>.zadatak<broj zadatka>

gdje <imeprezime> predstavlja vaše ime i prezime, a <broj zadatka> dvoznamenkastu brojčanu oznaku zadatka (npr. 03 ili 13).

Primjer: rješenje petog zadatka studenta Ivana Horvata treba biti u java paketu:

hr.atos.praksa.ivanhorvat.zadatak05

## Zadatci

1. Instaliraj program Eclipse. Na stranici <http://www.eclipse.org/downloads/> nalazi se popis različitih alata iz Eclipse porodice. Za pripremu prakse bit će dovoljan Eclipse Standard. Odaberi verziju operacijskog sustava koji koristiš i preuzmi alat. Eclipse se ne instalira, nego jednostavno „raspakira“ na željenu lokaciju.
2. Kreiraj Java projekt i nazovi ga PripremaPrakse\_<ImePrezime>, pri čemu <ImePrezime> trebate zamijeniti svojim imenom i prezimenom (npr. PripremaPrakse\_IvanHorvat)
3. Kreiraj tzv. "Hello World" aplikaciju tj. aplikaciju koja će nakon pokretanja na konzolu ispisati riječi "Hello World, ovo je prva java aplikacija.".
4. Kreiraj program koji ima definirano polje pet različitih cjelobrojnih vrijednosti. Program treba provjeriti je li svaka od pet vrijednosti paran ili neparan broj i da li je višekratnik od jednog od idučih brojeva: 3, 5 ili 11, te ispisati poruku na konzolu u obliku „XY je paran broj“ ili „XY je neparan broj“ te „XY je višekratnik od 3 “ (ili 5 ili 11, zamijeniti broj u tekstu).
5. Kreiraj program koji ima definiran početak i kraj intervala te jedan brojač. Neka početak intervala bude manji od 10, a kraj intervala veći od 100. Program treba provjeriti jesu li početak i kraj intervala ispravno definirani te ispisati poruku o pogrešci ukoliko nisu. Ukoliko jesu program treba za svaki broj u zadanom intervalu napraviti sljedeće:
   * ukoliko je broj manji ili jednak 18, brojač treba povećati za 4
   * ukoliko je broj veći od 18, brojač treba umanjiti za 1
   * ukoliko je broj djeljiv s 20, treba prekinuti procesuiranje tog broja i prijeći na sljedeći
   * ukoliko je broj jednak ili veći od 75, treba prekinuti procesuiranje intervala te ispisati vrijednost brojača
6. Kreiraj program koji će koristeći for petlje automatizirati ispis tablice množenja u ovom obliku:

-------------------------------

: : : TABLICA MNOZENJA : : :

-------------------------------

\* | 1 2 3 4 5 6 7 8 9

-------------------------------

1 | 1 2 3 4 5 6 7 8 9

2 | 2 4 6 8 10 12 14 16 18

3 | 3 6 9 12 15 18 21 24 27

4 | 4 8 12 16 20 24 28 32 36

5 | 5 10 15 20 25 30 35 40 45

6 | 6 12 18 24 30 36 42 48 54

7 | 7 14 21 28 35 42 49 56 63

8 | 8 16 24 32 40 48 56 64 72

9 | 9 18 27 36 45 54 63 72 81

-------------------------------

: : : : : : : : :by Ime

-------------------------------

Umjesto "Ime" treba ispisati ime uneseno s konzole i pri tome pripaziti da zadnje slovo imena bude poravnato s desnim rubom tablice.

1. Kreiraj program koji će ispisati koliko ima brojeva djeljivih sa 6 u intervalu između neka dva prirodna broja unesena s konzole.
2. Kreiraj program koji će s konzole učitati redni broj mjeseca u nekoj prijestupnoj godini. Napraviti provjeru je li uneseni redni broj mjeseca ispravan (ne postoji npr. 15. mjesec), a zatim ispisati kalendar tog mjeseca u obliku:

P U S Č P S N

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30 31

Pretpostavlja se kako prvi u mjesecu pada na ponedjeljak. Također, treba ispisati onoliko dana koliko ih taj mjesec po klasičnom kalendaru doista ima.

1. Luka je vaš dobar prijatelj. Nedavno vam je spomenuo kako bi volio imati grafičko rješenje prikaza njegovog troška struje njegove „krušne“ peći kroz godinu (on je naime pizza majstor i ima svoj restoran). Račun mu stalno varira, ali nikada ne prelazi 4500kn. Prvo ste mu rekli da još ne znate raditi grafički prikaz u Javi, ali vam je bilo žao ne ponuditi mu nikakvo rješenje pa ste na papiru nacrtali ovakav graf i rekli da znanje za napraviti ovakav prikaz imate:

5000kn - |

| x

4000kn - | x

|

3000kn - | x x

|

2000kn - | x

|

1000kn - | x x x

|

0kn - | x x

-- -- -- -- -- -- -- -- -- -- -- --

1 2 3 4 5 6 7 8 9 10 11 12

Također ste mu pojasnili kako će morati unijeti plaću za svaki pojedini mjesec, a onda će program iscrtati graf. Upozorili ste ga kako će vrijednosti plaće biti aproksimirane. Primjerice mjesečna plaća koja iznosi od 1501kn pa sve do 2500kn, na grafu će biti obilježena iksićem uz vrijednost 2000kn. On se naposljetku složio s vašim prijedlogom i od srca zahvalio. Stoga kreirajte program kojim ćete pomoći Luki.

1. Kreiraj java program koji će pretražiti sve datoteke s ekstenzijama "csv" i "txt" unutar jednog direktorija (foldera) kako bi provjerio nalazi li se u njima definirana tekstualna fraza. Nazive datoteka u kojima se fraza nalazi, ispisat će na konzolu. Putanju direktorija za pretraživanje kao i tekstualnu frazu treba unijeti s konzole.

Popis klasa koje bi mogle biti korisne:

java.io.File (za manipuliranje datotekama / direktorijima)

java.io.BufferedReader (za čitanje sadržaja datoteke)

java.lang.String (za manipulaciju sa tekstom)

Fraza usput može biti regular expression.

1. Kreiraj program koji će omogućiti interakciju s korisnikom na jednom od tri ponuđena jezika. Interakcija se treba sastojati od nekoliko međusobno izmijenjenih poruka. Prijevodi na pojedini jezik trebaju biti dostupni unutar java properties datoteka za svaki jezik odvojeno.
2. Napraviti program koji će omogućiti korisniku ispis statistike broja pojedinih riječi unutar datoteke koju odabere. Riječi trebaju biti poredane po abecedi i treba ignorirati velika/mala slova. Ispod se nalazi primjer ispisa:

U datoteci XX nalaze se sljedece rijeci:

------------------------

Rijec (broj ponavljanja)

------------------------

auto (17)

auta (2)

boje (19)

cipele (3)

je (39)

su (110)

trava (1)

zeleno (2)

------------------------

1. Kreiraj program koji se sastoji od jedne klase i sučelja koje klasa nasljeđuje. Sučelje treba definirati dvije različite metode po vlastitom izboru.
2. Kreiraj program kojim će korisnik izračunati površinu ispod neke krivulje opisane funkcijom y=f(x)=A\*funk(x)+B od neke početne do krajnje točke (T1,T2). Izračun izvesti numeričkom metodom. Ulazni parametri su A, B, T1i T2 te funkcija koja može biti sinus, kosinus, tangens ili kotangens. Pri rješavanju zadatka obavezno koristiti više klasa i(li) sučelja te nasljeđivanje.
3. Za jednu tvrtku potrebno je napraviti demo aplikaciju u kojoj će moći raditi administraciju zaposlenika te njihbovih radnih zadataka. Ovo su im najbitniji podatci:

Zaposlenici

ime

prezime

radno mjesto

oib

Zadaci

naziv

opis

tip (bug, task)

trenutni status (otvoren, zatvoren, u tijeku)

kompleksnost (broj)

potrošeno vrijeme (sati)

početni datum i vrijeme

završni datum i vrijeme

(napomena, početak i završetak ne znače nužno da je potrošeno vrijeme njihova razlika)

Potrebno je omogućiti administraciju na nekoliko razina tj. potrebno je definirati korisnička prava korisnika aplikacije po grupama: "admin", "superuser" i "user" tako da se omoguće sljedeće akcije:

* + kreiranje zadataka i zaposlenika (admin, superuser)
  + izlistanje zadataka i zaposlenika (admin, superuser, user)
  + izmjena zadataka i zaposlenika (admin)
  + brisanje zadataka i zaposlenika (admin)

Naposljetku aplikacija mora omogućiti kreiranje izvještaja:

* + po radnom mjestu tj. koliko radnika radi na pojedinom radnom mjestu
  + utrošeno vrijeme po osobi
  + najduže otvoren zadatak

Izvještaje mogu kreirati samo korisnici s admin ili superuser pravima.

Sučelje za komunikaciju s korisnicima neka bude konzola, a podaci se trebaju spremati u tekstualne datoteke ili bazu podataka po izboru.

# Upute za završni izvještaj

Završni izvještaj sadrži rezultat svladavanja teorijske i praktične grupe zadataka:

* Završni izvještaj teorijskog dijela podrazumijeva da u drugi stupac tablice iz poglavlja 1. Teorijski dio unesete vaše razumijevanje pojedinog pojma navedenog u prvom stupcu te tablice. Preporuča se pojam opisivati vlastitim riječima na hrvatskom jeziku.
* Za završni izvještaj praktičnog dijela trebate napraviti eksport Eclipse projekta PripremaPrakse\_<ImePrezime> (koji sadrži java programe tj. rješenja svih zadataka koje ste riješili tijekom priprema) u zip datoteku jednakog naziva (PripremaPrakse\_<ImePrezime>.zip). Tako kreiranu zip datoteku trebate umetnuti na kraj ovog word dokumenta.

Nakon što ste odradili navedena dva koraka spremite ovaj word dokument. Prije nego ga pošaljete provjerite jeste li promijenili ImePrezime iz naziva dokumenta u vaše ime i prezime.

