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 predstavlja osnovni tvorbeni blok u OOP-u te služi za kreiranje objekata.  Neki primjer jednostavne klase bi bio „auto“ koji može imati objekte „VW Polo“ , „BMW“ itd. |
| Objekt | Objekt je neka realizacija klase, kao što sam napisao gore, ako je klasa „auto“, objekt je npr. „BWM“. |
| Metoda | Funkcija koja pripada nekoj klasi ( Funkcija != Metoda). |
| Varijabla | Skladište za spremanje podataka, u kojem se tijekom rada programa, isti podaci mogu mijenjati. |
| Sučelje | Omogućuje interakciju između korisnika i sustava/uređaja. |
| Nasljeđivanje | Jedna klasa može naslijediti metode i dio strukture ili cijeli strukturu od neke druge klase. |
| Overload | Kad klasa ima više metoda istog imena, ali različitih parametara. |
| API | Služi za komunikaciju između više aplikacija, također omogućuje korištenje već napravljenih funkcija kako bi skratili kompleksnost i vrijeme programiranja. Npr. prilikom izrade jednog od projekata koristio sam Spotify API kako bi pauzirao pjesmu. Spotify je napravio funkciju za to i objasnio kako se koristi, a ja sam ju nakon autorizacije mogao koristiti. |
| JDK | Razvojno okruženje za Java aplikacije i programe (Java Development Kit). |
| JRE | Java Runtime Environment je softverski sloj koji omogućuje pokretanje Jave na računalu na kojem je instaliran. |
| JVM | Java virtual machine je dio JRE i aplikacija koja služi za interpretaciju i „prevođenje“ Java bytecoda. |
| jar | Format datoteke specifičan za Javu. |
| j2se | Standardna platforma za razvoj i upotrebu Jave. |
| j2ee | Enterprise platforma napravljena kao dodatak standardnoj platformi. Pruža API i JRE za razvoj velikih, fleksibilnih, pouzdanih i sigurnih mrežnih aplikacija. |
| OpenJDK vs OracleJDK | OpenJDK je open source implementacija Java platforme sa slobodnom licencom dok OracleJDK ima komercijalnu licencu. Također koliko sam shvatio OracleJDK ima nešto bolje performanse, ali ništa preznačajno. |
| Maven | Apache Maven je framework za upravljanje projektima te pomaže sa izgradnjom projekta, dokumentacijom i izvještavanjem. |
| Ant | Isto kao i Apache Maven , Apache Ant služi za olakšavanje samog procesa izrade projekta, više je „tool box“ za razliku od Maven-a. Maven je više orijentiran na project management dok je Ant build automation tool. Također se više ne koristi toliko. |
| Gradle | Radi sličnu stvar kao Ant samo bolje zbog bolje integracije sa IDE-om, napravljen je da bi popravio mane Maven-a i Anta te podržava multi-project builds. |
| Koja su prva 4 byte-a svake klase (class file-a) u hex formatu i zašto? | 0xCAFEBABE( CA FE BA BE )  James Gosling (otac Jave) je trebao „couple of magic numbers“ za class file, pošto je objekt file imao CAFEDEAD odlučio je da će class file biti CAFEBABE.  Izvor, te bolje objašnjenje:  https://stackoverflow.com/questions/2808646/why-are-the-first-four-bytes-of-the-java-class-file-format-cafebabe |
| **Konvencija naziva** | |
| Naziv klasa | Nazivi klasa trebaju biti imenice, počinju velikim početnim slovom, trebaju biti jednostavni, ali također dovoljno razumljivi. Npr. class Car. |
| Naziv metoda | Metode trebaju biti glagoli, prvo slovo prve riječi je malo dok ostale riječi započinju velikim slovom. Npr. driveFast(); |
| Naziv varijabli | Nazivi varijabli trebaju biti kratki, ali smisleni, također kao metode započinju malim slovom dok ostale riječi unutar naziva započinju velikim slovom. Npr. float carMileage; |
| **Tipovi podataka** | |
| Primitivni tipovi | Tipovi podataka : Boolean, Char, Int, Float 🡪 predstavljaju neku vrijednost. |
| Referencirajući tipovi | Tipovi podataka : Class, Array, String, Interface 🡪 složeni, tvore se od drugih referentnih tipova ili primitivnih tipova. |
| **Operatori** | |
| Aritmetički | +,-,\*,/,%,++,-- |
| Relacijski | ==,!=,<,>,<=,>= |
| Operatori uvjeta | ? |
| Operatori pridruživanja | =,+=,-=,\*=,/=,%= |
| **Kontrola toka: odlučivanje** | |
| if | Ako je uvjet zadovoljen napravi nešto. |
| if – else | Ako je uvjet zadovoljen napravi nešto, ako nije napravi else. |
| if – else if – else | Dva uvjeta, pa else. |
| switch – case | Unutar switcha imamo expression koji provjeravamo kroz cases, ako zadovoljava neki case izvrši ga. |
| **Kontrola toka: petlje** | |
| for | Iskreno ne znam kako da opišem for pa ću napisati primjer:  Npr. Ispis arraya  String[] names = {"Java","C","C++","Python","JavaScript"};  **for**(int i = 0; i<5; i++) {  System.out.println(name);  } |
| while | Koristi se najčešće kada ne znamo broj iteracija unaprijed. |
| do – while | Kada broj iteracija nije poznat, a mi moram izvršiti petlju barem jednom, možemo koristiti do-while. |
| **Kontrola toka: bezuvjetni skok** | |
| break | Koristi se za izlazak iz petlje. |
| continue | Koristi se za prekidanje jedne iteracije petlje, ako je npr. zadovoljen neki uvjet. |
| return | Služi za završavanje metode i može se koristiti za vraćanje vrijednosti. |
| **Kontrola toka: nepravilnosti** | |
| try – catch | Unutar try možemo napisati dio koda koji testiramo za probleme, ukoliko naiđemo na neki error izvršava se catch dio. |
| try – catch – finally | Finally služi za izvršavanje koda neovisno o ishodu try-catch. |
| try – finally | Razlika između try-catch i try-finally je to što u try-finally, finally dio je uvijek izvršen nakon try neovisno je li bio error ili ne. |
| throw | Možemo ga koristiti za kreiranje custom errora. |
| **Ključne riječi** | |
| abstract | Koristi se kao modifier za klasu ili metodu. Abstract klasa se ne može koristiti za kreiranje objekata, a apstract metoda može biti korištena samo u abstract klasi te nema body. |
| assert | Koristi se tijekom debuganja, te izbacuje AssertionError, ako uvjet nije zadovoljen. |
| boolean | Varijabla koja ima dva stanja, TRUE ili FALSE. |
| break | Koristi se za izlazak iz petlje. |
| byte | Varijabla koja može spremiti vrijednost u intervalu od -128 do 127. |
| case | Case je dio switch-case statementa . |
| catch | Dio try-catch, koristi se za izvršavanje nekog koda ukoliko se pojavi error. |
| char | Varijabla u koju se sprema jedan znak. |
| class | Referentni tip podatka koji se koristi za izradu klase. |
| const | Služi kako bi rekli kompajleru da vrijednost neke varijable ne bi trebala biti modificirana nakon deklaracije. |
| continue | Koristi se za prekidanje jedne iteracije petlje ako je npr. zadovoljeni neki uvjet. |
| default | Pruža backwards compatibility za stare interface-ove tako da mogu imati nove metode bez utjecaja na postojeći kod. |
| do | Dio do-while petlje, izvršava do dio sve dok je uvjet u while zadovoljen. |
| double | Tip varijable za decimalne brojeve gdje je potrebna visoka preciznost. |
| else | Dio if-else, izvršava se ako nije zadovoljen if uvjet. |
| enum | Klasa koji predstavlja grupu konstanti  Npr.  enum Level {  LOW,  MEDIUM,  HIGH  } |
| extends | Služi za nasljeđivanje, koristi se kod klasa npr. class Car extends Vehicle. |
| final | Ako neka varijabla ima final, vrijednost te varijable je konstantna. |
| finally | Finally služi za izvršavanje koda neovisno o ishodu try-catch. |
| float | Varijabla za spremanje decimalnih brojeva sa preciznošću od 6-7 znamenki. |
| for | String[] names = {"Java","C","C++","Python","JavaScript"};  **for**(int i = 0; i<5; i++) {  System.out.println(name);  }  Primjer for petlje u javi. |
| goto | Omogućava prijelaz na drugi dio programa, najčešće se ne koristi zbog toga što kod postaje nepregledniji i postoje veće šanse za bug-ove. |
| if | Ako je uvjet zadovoljen napravi nešto. |
| implements | Koristi se za implementiranje interface-a. |
| import | Import koristimo za dodavanje dodatnih paketa, klasa ili interface-a. |
| instanceof | Koristi se za testiranje pripada li instanca objekta nekoj klasi ili interface-u. |
| int | Cjelobrojna varijabla. |
| interface | Predstavlja opis onoga što klasa radi, ali ne i kako radi. |
| long | Varijabla za spremanje iznimno velikih cjelobrojnih brojeva. |
| native | Native u Javi primjenjuje se na metodu kako bi se naznačilo da je metoda implementirana u izvornom kodu koristeći JNI. |
| new | Koristi se za stvaranje objekta iz klase. |
| package | Služi za grupiranje klasa u jedan modul. |
| private | U slučaju dodjeljivanja privatnog prava pristupa metodi ili varijabli tada samo “kod” unutar iste klase može pristupiti toj varijabli ili pozvati tu metodu. |
| protected | Dodjeljivanje ove vrste prava pristupa znači da “kod” unutar klase, podklase, kao i “kod” unutar klasa koje se nalaze u istom paketu može pristupiti klasi, varijablama, konstruktoru ili metodama. |
| public | Modifikator pristupa za klase, metode, konstruktore i varijable koji ih čini dostupnima bilo kojoj drugoj klasi. |
| return | Služi za završavanje metode i može se koristiti za vraćanje vrijednosti. |
| short | Cjelobrojna varijabla koja sprema brojeve u intervalu od -32,768 to 32,767. |
| static | Koristi se ponajviše u svrhe memory managmenta, static varijable dobiju memoriju samo jednom prilikom učitavanja klase. |
| super | Služi za upućivanje na objekt roditeljske klase. |
| switch | Unutar switcha imamo expression koji provjeravamo kroz cases ako zadovoljava neki case, izvrši ga. |
| synchronized | Koristi se kako bi se spriječilo preklapanje izvršavanja metoda. |
| this | Koristi se za referenciranje trenutnog objekta u metodi ili konstruktoru. |
| throw | Služi za kreiranje custom errora. |
| throws | Govori nam kakav exception može biti bačen metodom. (eng. may be thrown by a method) |
| transient | „Sprječava“ serijalizaciju varijable. |
| try | Koristi se u try-catch i ostalim sličnim metodama kontrole toka, izvršava neki dio koda. |
| void | Return type kada metoda/funkcija ne vraća ništa (ne može vraćati ništa). |
| volatile | Koristi se kako bi varijabla bila spremljena u glavnu memoriju, a ne CPU cache. |
| while | Petlja, koristi se najčešće kada ne znamo broj iteracija unaprijed. |
| false | Jedna od vrijednosti boolean varijable te označava točnost nekog izraza. |
| null | Referentna vrijednost bilo koje varijable prije nego što joj se pridoda neka vrijednost. |
| true | Jedna od vrijednosti boolean varijable te označava točnost nekog izraza. |
| **SQL** | |
| **Općenito** | |
| table | Tablice su objekti u koji sadrže sve podatke iz baze podataka. |
| column | Stupac unutar tablice. |
| view | Služi za kreiranje pogleda na tablicu čime ograničavamo pristup podacima. |
| index | Indeks se koristi za brže povlačenje podataka iz baze. |
| constraint | Koriste se za postavljanje raznih ograničenja unutar baze. |
| stored procedure | Već pripremljeni SQL kod koji se može spremiti i kasnije ponovo koristiti. |
| trigger | Posebna vrsta procedure koja se aktivira nakon nekog događaja. |
| create | Koristi se kao naredba za stvaranje različitih dijelova baze npr. CREATE TABLE; CREATE VIEW; itd. |
| drop | Briše tablicu. |
| select | Koristi se za odabir podataka iz baze. |
| insert | Koristi se za unos novih podataka u tablicu. |
| update | Koristi se za uređivanje podataka u tablici. |
| delete | Koristi se za brisanje podataka u tablici. |
| commit | Koristi se za trajnu promjenu baze podataka trenutnom transakcijom. |
| inner/outer/left/right join - objasniti | INNER JOIN - odabire podatke koji su isti u obje tablice  LEFT JOIN – odabire sve podatke iz prve tablice i iste podatke iz obje tablice  RIGHT JOIN - odabire sve podatke iz druge tablice i iste podatke iz obje tablice  OUTER JOIN – odabire sve podatke iz obje tablice |
| **Vrste relacija između tablica** | |
| 1 – 1 | Jedna instanca objekta određenog tipa u vezi je s drugom instancom entiteta iste vrste. |
| 1 – više | Primarni ključ entiteta sa strane veze JEDAN doda se kao ključ na strani VIŠE. |
| više - više | Dodaje se novi entitet koji sadrži primarne ključeve obaju ključnih elemenata te ti atributi zajedno tvore složeni primarni ključ novonastalog entiteta. |
| **SVN** | |
| **Općenito** | |
| Check-out | Provjera radne verzije iz repozitorija. |
| Commit | Pošalji promjene u repozitorij. |
| trunk | „Main line of development“. |
| branch | Side line of development. |
| **UNIX** | |
| **Osnovne naredbe** | |
| Ls | Izlistava sadržaje direktorija. |
| Cd | Promjena direktorija. |
| Mkdir | Napravi direktorij. |
| Grep | Koristi se za pretraživanje teksta u dokumentu. |
| Cp | Copy. |
| Rm | Delete. |
| Mv | Koristi se za premještanje datoteka ili direktorija. |
| Rmdir | Brisanje praznih direktorija. |
| Touch | Kreiranje direktorija. |
| Chmod | Promjena prava. |
| Chown | Promjena administratora neke datoteke ili direktorija. |
| Man | Upute za korištenje. |
| Ps | Prikaz informacija o aktivnih procesima. |
| Kill | Prekidanje određenog procesa. Kao identifikator se koristi PID(Proces ID) |
| Kako izlistati stanje servisa | service --status-all |
| Gdje se definiraju mount pointi da budu perzistentni | /etc/fstab |
| Vrste distribucija | Linux, openserver, Minix, Mac itd. |
| Koja je zadnja verzija referentnog kernela | 5.16.7 |
| **Specijalni direktoriji** | |
| / | Root direktorij |
| . | Trenutni direktorij |
| .. | Direktorij iznad |
| ~ | Označuje $HOME sistemsku varijablu |
| **Putanje** | |
| Apsolutna putanja | Apsolutna putanja se odnosi na potpuni izraz potreban da dođemo do neke mape ili datoteke. |
| Relativna putanja | Relativna putanja se odnosi na izraz potreban da dođemo do neke mape ili datoteke iz naše trenutne mape ili datoteke. |
| **Osnovni preglednici** | |
| Vrste osnovnih preglednika |  |
| Naredbe u „less“ pregledniku |  |
| **vi editor** | |
| Uređivanje teksta | Neke od naredbi: r – zamjeni jedan char  R – zamjeni vise charova  Cw- zamjeni trenutnu riječ sa novim tekstom |
| Snimanje promjena | :w<Return> Zapisati trenutni sadržaj u datoteku nazvanu u izvornom vi pozivu |
| Brisanje retka | Neke od naredbi: x – obriši jedan char  Nx – obriši N charova  D- obriši ostatak linije |
| Pretraga teksta i navigacija (next/previous match) | /string Traži naprijed u tekstu za string  ?string Traži unazad u tekstu za string  n Navigacija naprijed u tekstu do pojave stringa  N Navigacija unazad u tekstu do pojave stringa |
| **General know-how** | |
| SOAP | Protokol za komunikaciju između aplikacija pomoću HTTP-a. |
| RegExp | Regular expression , koristi se za pretraživanje stringova. |
| Node-red | Node-red, alat za povezivanje API-a, hardvera i online servisa. |
| GMT / UTC / CEST | Vremenske zone |
| RabbitMQ | „Open source message broker“. |
| MosqitoMQ | Isto kao rabbit, mosqito je open source message broker, ali sa većim fokusom na funkcioniranje u okolini gdje je signal ograničen te na uređajima niske potrošnje. |
| Cassandra | NoSQL baza podataka |
| NoSQL / NewSQL | Baze podataka koje podatke spremaju različito od relacijskih baza kako bi lakše i brže radili sa velikim količinama informacija i prilikom većeg opterećenja. |
| ETSI | Neovisna, neprofitna organizacija za standardizaciju u području informacija i komunikacija. |
| 3GPP | 3GPP je zajednički naziv za više raznih organizacija koje istražuju i razvijaju protokole za telekomunikacije mreže. |
| LTE | Bežična telekomunikacijska tehnologija. |
| 5G | Peta generacija bežične telekomunikacijske tehnologije. |
| Što je podatak, a što informacija | Podatak je jednostavna činjenica koja ima neko značenje te je sama po sebi apstraktna.  Informacija je rezultat analize i organizacije podataka. |
| Što je REGEXP, napisati regexp za IPv4 adresu i objasniti ga | ^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$ Jednostavi ipv4 regexp  Provjerava ima li brojeva [0-9]{1,3} tri puta {3} i odvojenih točkama \. i završava s drugim brojem. |
| Što je RFC | Dokument kojeg izdaje ERTF. Opisuje metode, ponašanja, istraživanja i/ili inovacije primjenjive na Internet. |
| Kojim RFCom je opisan HTTP protokol (v 1.1), kojim HTTP2? | HTTP v 1.1 🡪 RFC 2616  HTTP v 2 🡪RFC 7540 |
| JSON | Tip datoteke najčešće korišten za razmjenu podataka između web aplikacija. Prednost mu je također što je jako jednostavno raščlaniti pojedine objekte. |
| REST | API koji nam omogućuje lakše slanje raznih poziva na server. |
| WS | WebSocket predstavlja objekt koji nam olakšava povezivanje na server te slanje i primanje podataka. |
| FYI | For your information |
| IIRC, IMHO | If I recal correctly, In my humble 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.