**KRIMINALISTIČKO**

**POLICIJSKI**

**UNIVERZITET**

**Osnovne akademske studije Informatika i računarstvo**

**Predmet: Objektno-orijentisani programski jezici**

Školska godina: 2019/2020.

Predmetni profesor: prof. dr Kristijan Kuk, prof. dr Milan Gocić

**NAPOMENE:**

1. Naziv projekta OBAVEZNO mora biti u formatu ImePrezime\_BrojIndeksa.

Primer: UrosDinic\_1B1-0003-15)

**DOMAĆI ZADATAK 5**

1. Napravite javne klase **ProizvodException***,* **Proizvod**, **PrehambeniProizvod**i **WriteProizvod**sa sledećim uslovima:
2. Klasa **ProizvodException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor

nadklase prosleđujući mu parametar.

1. Klasa **Proizvod** ima:

* Atribut naziv.
* Atribut cena (npr. 34.5 dinara).
* Konstruktor bez ulaznih argumenata koji postavlja vrednosti atributa naziv i cena na “nepoznat”.
* Konstruktor koji kao ulazne argumente prima vrednosti za naziv i cenu i dodeljuje ih atributima samo ako uneti naziv nije null i ako je uneta cena veća od nule. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase ProizvodException sa odgovarajućom porukom, a atributima dodeliti vrednosti “nepoznat” i 0.0 respektivno.
* Odgovarajuće get i set metode za ova dva atributa. Nedozvoljene vrednosti za naziv su svi null Stringovi, a za cenu sve cene koje su manje ili jednake nuli. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase ProizvodException sa odgovarajućom porukom.

1. Klasa **PrehambeniProizvod** nasleđuje klasu Proizvod i ima:

* Atribut rokTrajanja koji predstavlja datum do kojeg je proizvod upotrebljiv. Ovaj datum je dat kao String.
* Konstruktor bez ulaznih argumenata koji postavlja vrednosti atributa naziv, cena i rokTrajanja na “nepoznat”, 0.0 i “12.12.2007.” respektivno.
* Konstruktor koji kao ulazne argumente prima vrednosti za naziv, cenu i rokTrajanja i dodeljuje ih atributima samo ako uneti naziv nije null, ako je uneta cena veća od nule i ako rok trajanja nije null. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase ProizvodException sa odgovarajućom porukom, a atributima dodeliti vrednosti “nepoznat”, 0.0 i “12.12.2007.” respektivno.
* Get i set metodu za atribut rokTrajanja. Nedozvoljene vrednosti za ovaj atribut su null stringovi. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase ProizvodException sa odgovarajućom porukom.

1. Glavna klasa **WriteProizvod** koja kreira dva objekta klase PrehrambeniProizvod. Prvi objekat ima naziv “hleb”, cenu 30.0 dinara i rok trajanja do “15.12.2018.”. Drugi objekat ima naziv “senf”, cenu 120.5 dinara i rok trajanja “10.10.2019.”. Prvo objekat kreirati korišćenjem konstruktora koji ima ulazne argumente, a drugi korišćenjem konstruktora koji nema ulazne argumente. Koristeći klasu **PrintWriter** upisati podatke o oba proizvoda u tekstualni fajl “proizvodi.txt”.
2. Napraviti javne klase **OsobaException**, **Osoba**, **Djak**, **Penzioner** i **WriteOsoba**sa sledećim uslovima:

1. Klasa **OsobaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **Osoba** ima:

* Atribut ime. Početna vrednost ovog atributa je “N”.
* Atribut prezime. Početna vrednost ovog atributa je “N”.
* Atribut jmbg (String) koji predstavlja matični broj.
* Metodu getIme koja vraća trenutnu vrednost atributa ime.
* Metodu setIme koja kao ulazni argument dobija novu vrednost za atribut ime i dodeljuje mu je. Dodeljivanje se vrši samo ako je uneta vrednost različita od null, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa odgovarajućom porukom.
* Metodu getPrezime koja vraća trenutnu vrednost atributa prezime.
* Metodu setPrezime koja kao ulazni argument dobija novu vrednost za atribut prezime i dodeljuje mu je. Dodeljivanje se vrši samo ako je uneta vrednost različita od null, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa adekvatnom porukom.
* Metodu getJmbg koja vraća trenutnu vrednost atributa maticnibroj.
* Metodu setJmbg koja kao ulazni argument dobija novu vrednost za atribut jmbg i dodeljuje mu je. Dodeljivanje se vrši samo ako je uneta vrednost različita od null, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa odgovarajućom porukom.

1. Klasa **Djak** nasleđuje klasu Osoba i ima:

* Atribut prosecnaOcena koji predstavlja prosečnu ocenu u školi (npr. 4,66).
* Metodu getProsecnaOcena koja vraća trenutnu vrednost atributa prosecnaOcena.
* Metodu setProsecnaOcena koja kao ulazni argument dobija novu vrednost za atribut prosecnaOcena i dodeljuje mu je. Dodeljivanje se vrši samo ako je uneta vrednost u rasponu 1.0 do 5.0, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa odgovarajućom porukom.

1. Klasa **Penzioner** koja nasleđuje klasu Osoba i ima:

* Atribut penzija koji predstavlja iznos koji penzioner prima kao penziju (npr. 14200,5 dinara).
* Metodu getPenzija koja vraća trenutnu vrednost atributa penzija.
* Metodu setPenzija koja kao ulazni argument dobija novu vrednost za atribut penzija i dodeljuje mu je. Dodeljivanje se vrši samo ako je uneta vrednost veća od nule, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa odgovarajućom porukom.

1. Glavna klasa **WritetOsoba** koja kreira po jedan objekat klase Osoba, Djak i Penzioner. Dodeliti osobi ime “Pera Peric” i matični broj “1212007710567”. Dodeliti đaku ime “Mika Lazic”, matični broj “1010000715076” i prosečnu ocenu 5.0. Dodeliti penzioneru ime “Zika Zikic”, maticni broj “0909944710078” i penziju 23400.0 dinara. Koristeći klasu **PrintWriter** upisati podatke o sva tri proizvoda u tekstualni fajl “osobe.txt”.
2. Napraviti javne klase **KucniAparatException**, **KucniAparat**, **Televizor**, **Radio**i **WriteKucniAparat**sa sledećim uslovima:
3. Klasa **KucniAparatException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **KucniAparat** ima:

* Atribut marka koji predstavlja ime proizvođača aparata.
* Atribut model koji predstavlja naziv modela aparata.
* Atribut ukljucen koji ima vrednost TRUE ako je aparat uključen, a u suprotnom FALSE.
* Konstruktor koji kao ulazne argumente prima vrednosti za marku, model i ukljucen (true ili false) i dodeljuje ih atributima. Dodeljivanje se vrši isključivo ako su unete vrednosti za marku i model različiti od null, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KucniAparatException sa odgovarajućom porukom.
* Odgovarajuće get i set metode za atribute marka i model. Nedozvoljene vrednosti za atribute marka i model su Stringovi koji imaju vrednost null. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KucniAparatException sa odgovarajućom porukom.
* Metodu ukljuci koja ukljucuje aparat tj. postavlja vrednost atributa ukljucen na TRUE.
* Metodu iskljuci koja iskljucuje aparat tj. postavlja vrednost atributa ukljucen na FALSE.
* Metodu ispisi koja primenom klase **PrintWriter** upisuje podatke o marki i modelu aparata kao i to da li je aparat uključen u tekstualni fajl “marka\_model.txt”.

1. Klasa **Televizor** koja nasleđuje klasu KucniAparat i ima:

* Atribut program koji predstavlja broj programa koji se trenutno prikazuje. Početna vrednost za atribut program je 1.
* Konstruktor koji kao ulazne argumente prima vrednosti za marku, model, ukljucen (true ili false) i program i dodeljuje ih atributima. Dodeljivanje se vrši isključivo ako su unete vrednosti za marku i model različiti od null i ako je broj programa između 1 i 40, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KucniAparatException sa odgovarajućom porukom.
* Metodu promeniProgramNavise koja menja program na sledeći. Ako se na televizoru već pušta 40. program, program dobija vrednost 1.
* Metodu promeniProgramNanize koja menja program na prethodni. Ako se na televizoru već pušta 1. program, program dobija vrednost 40.
* Metodu ispisi koja, pored poruke o marki, modelu i tome da li je uključen, primenom klase **PrintWriter** upisuje podatke u tekstualni fajl “televizor.txt” o tome koji program se trenutno prikazuje.

1. Klasa **Radio** koja nasleđuje klasu KucniAparat i ima:

* Atribut frekvencija koji predstavlja trenutnu frekvenciju koju pušta radio aparat (npr. 87,5). Početna vrednost za ovaj atribut je 87.5.
* Konstruktor koji kao ulazne argumente prima vrednosti za marku, model, ukljucen (true ili false) i frekvenciju i dodeljuje ih atributima. Dodeljivanje se vrši isključivo ako su unete vrednosti za marku i model različiti od null i ako je frekvencija između 87.5 i 108.0, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KucniAparatException sa odgovarajućom porukom.
* Odgovarajuće get i set metode za atribut frekvencija. Pri unosu nove vrednosti za atribut frekvencija proveriti da li je nova vrednost u opsegu 87.5 – 108.0. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KucniAparatException sa odgovarajućom porukom.
* Redefinisati metodu ispisi klase KucniAparat tako da, ako je radio uključen, primenom klase **PrintWriter** upisuje se podatak u tekstualni fajl “radio.txt” o frekvenciji koju radio pušta. Ako je radio isključen, primenom klase **PrintWriter** upisuje se podatak u tekstualni fajl “radio.txt” o tome da je isključen. Ni u jednom ni u drugom slučaju ne treba upisivati podatake o marki i modelu radija.

1. Glavna klasa **WriteKucniAparat** koja kreira po jedan objekat klase KucniAparat, Televizor i Radio. Kucni aparat je marke Philips, model SQ2 i uključen je. Televizor je LG, model LT33, isključen je i pušta 2. program. Radio je marke Sony, model SR, isključen je i pušta radio stanicu na frekvenciji 88.9. Isključiti kućni aparat, uključiti televizor i podesiti ga na 3. program, i uključiti radio i podesiti ga na frekvenciju 105.6. Koristeći klasu **PrintWriter** upisati podatke o sva tri uređaja u tekstualni fajl “kucniAparati.txt”.
2. Napravite javne klase **KnjigaException**, **Knjiga**i **WriteKnjiga**sa sledećim uslovima:
3. Klasa **KnjigaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **Knjiga** ima:

* Atribut naslov.
* Atribut autor koji sadrži ime i prezime autora.
* Atribut ISBN koji predstavlja jedinstvenu šifru knjige (String).
* Odgovarajuće get i set metode za sve atribute. Nedozvoljene vrednosti za sva tri atributa su null Stringovi, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KnjigaException sa odgovarajućom porukom.
* Redefinisanu metodu toString klase Object koja vraća jedan String. Ovaj String sadrži podatke o knjizi: naslov, autor i ISBN uz odgovarajući tekst.
* Redefinisanu metodu equals klase Object koja kao ulazni argument prima objekat klase Object, ali se smatra da će se zaista unositi objekti klase Knjiga. Ova metoda vraća true ako je vrednost atributa ISBN jednaka ISBN knjige koja je uneta kao ulazni argument. U suprotnom, metoda vraća false.

1. Glavna klasa **WriteKnjiga** koja kreira dva objekta klase Knjiga. Prva knjiga je “Istorija lepote”, njen autor je Umberto Eko, a ISBN je 0-234-567. Druga knjiga je “Istorija ruznoce”, njen autor je takođe Umberto Eko, a ISBN je 0-567-890. Proveriti da li je u pitanju ista knjiga, a nakon toga, koristeći klasu **PrintWriter** upisati podatke o sve tri knjige i podatak o tome da li su u pitanju iste knjige u tekstualni fajl “knjige.txt”.
2. Napravite javne klase **OsobaException**, **Osoba**i **WriteOsoba**sa sledećim uslovima:
3. Klasa **OsobaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **Osoba** ima:

* Atribut ime. Početna vrednost ovog atributa je “N”.
* Atribut prezime. Početna vrednost ovog atributa je “N”.
* Atribut JMBG koji predstavlja matični broj predstavljen kao String..
* Odgovarajuće get i set metode za ove atribute. Nedozvoljene vrednosti za ime i prezime i JMBG su null. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa odgovarajućom porukom.
* Redefinisanu metodu toString klase Object. Ova metoda bi trebalo da vraća String koji sadrži vrednosti imena i prezimena osobe, kao i odgovarajući tekst uz to.
* Redefinisanu metodu equals klase Object koja kao ulazni argument prima objekat klase Object ali se smatra da će se zaista unositi objekti klase Osoba. Prvo, potrebno je proveriti da li je zaista unet objekat klase Osoba, pa ako nije baciti izuzetak klase OsobaException sa odgovarajućom porukom i vratiti false. Ako jeste unet objekat klase Osoba, ova metoda vraća true ako su vrednosti atributa ime, prezime i JMBG jednaka imenu, prezimenu i JMBG unete osobe, a u suprotnom false.

1. Glavna klasa **WriteOsoba** koja kreira dva objekta klase Osoba. Dodeliti i jednoj i drugoj osobi ime “Pera Perić” i JMBG “1010980710034”pa proveriti da li je u pitanju ista osoba. Nakon toga, koristeći klasu **PrintWriter** upisati podatke o prvoj osobi i podatak o tome da li su u pitanju iste osobe u tekstualni fajl “osobe.txt”.
2. Napravite javne klase **MonitorException**, **Monitor**i **WriteMonitor**sa sledećim uslovima:
3. Klasa **MonitorException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **Monitor** ima:

* Atribut marka
* Atribut model
* Atribut velicinaEkrana koji predstavlja dužinu dijagonale ekrana u inčima (npr. 17”).
* odgovarajuće get i set metode za ova tri atributa. Nedozvoljene vrednosti za marku i model su null Stringovi a za dijagonalu svi brojevi manji od 12. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase MonitorException sa odgovarajućom porukom.
* Redefinisanu metodu toString klase Object koja vraća jedan String. Ovaj String sadrži sve podatke o monitoru: marka, model i dijagonala uz odgovarajući tekst.
* Redefinisanu metodu equals klase Object koja kao ulazni argument prima objekat klase Object, ali se smatra da će se zaista unositi objekti klase Monitor. Prvo, potrebno je proveriti da li je zaista unet objekat klase Monitor, pa ako nije baciti izuzetak klase MonitorException sa odgovarajućom porukom i vratiti false. Ako jeste unet objekat klase Monitor, ova metoda vraća true ako su vrednosti atributa marka, model i dijagonala jednaki marki modelu i dijagonali monitora koji je unet kao ulazni argument. U suprotnom, metoda vraća false.

1. Glavna klasa **WriteMonitor** koja kreira dva objekta klase Monitor. Prvi je Samsung model SyncMaster 151n sa dijagonalom 15”, a drugi je Sony model WS sa dijagonalom 17”. Proveriti da li je u pitanju isti monitor. Nakon toga, koristeći klasu **PrintWriter** upisati podatke o oba monitora i podatak o tome da li su u pitanju iste monitori u tekstualni fajl “monitori.txt”.
2. Napravite klase javne klase **ZaposleniException**, **Zaposleni**, **KancelarijskiRadnik**, **Menadzer**i **WriteZaposleni**sa sledećim uslovima:
3. Klasa **ZaposleniException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Apstraktna klasa **Zaposleni** ima:

* Atribut plata koji je realan broj.
* Metodu getPlata koja vraća trenutnu vrednost atributa plata.
* Apstraktnu metodu izracunajPlatu koja ne vraća ništa, ali ima ulazni argument koji predstavlja broj sati koji je zaposleni radio.

1. Klasa **KancelarijskiRadnik** koja nasleđuje klasu Zaposleni i:

* Realizuje metodu izracunajPlatu. Ova metoda postavlja vrednost atributa plata prema formuli PLATA = 100 \* BROJ\_SATI.

1. Klasa **Menadzer** koja nasleđuje klasu Zaposleni i:

* Realizuje metodu izracunajPlatu. Ova metoda postavlja vrednost atributa plata prema formuli PLATA = 1000 \* BROJ\_SATI.

1. Glavna klasa **WriteZaposleni** koja kreira po jedan objekat klase KancelarijskiRadnik i Menadzer. Izracunati i koristeci klasu **PrintWriter** upisati plate obojice u tekstualni fajl “zaposleni.txt” ako se zna da je Menadzer radio 250 sati, a kancelarijski radnik 200 sati.
2. Napravite javne klase **VoziloException**, **Vozilo**, **Automobil**, **Avion**i **WriteVozila**sa sledećim uslovima:
3. Klasa **VoziloException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Apstraktna klasa **Vozilo** ima:

* Atribut marka koji predstavlja marku vozila.
* Atribut model koji predstavlja model vozila.
* Odgovarajuće get i set metode za ova dva polja. Nedozvoljene vrednosti za ova dva polja su null Stringovi. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase VoziloException sa odgovarajućom porukom.
* Apstraktnu metodu daLiIdePoKopnu koja vraća boolean vrednost koja označava da li vozilo može da se kreće po kopnu.
* Apstraktnu metodu daLiIdePoVodi koja vraća boolean vrednost koja označava da li vozilo može da se kreće po vodi.
* Apstraktnu metodu daLiIdePoVazduhu koja vraća boolean vrednost koja označava da li vozilo može da leti.
* Metodu ispisi koja na ekranu ispisuje podatke o vozilu – marku, model i da li ide po kopnu, vodi ili vazduhu.

1. Klasa **Automobil** koja nasleđuje klasu Vozilo i:

* Implementiranu metodu daLiIdePoKopnu koja uvek vraća TRUE jer automobil može da se kreće po kopnu.
* Implementiranu metodu daLiIdePoVodi koja uvek vraća FALSE jer automobil ne može da se kreće po vodi.
* Implementiranu metodu daLiIdePoVazduhu koja uvek vraća FALSE jer automobil ne može da leti.

1. Klasa **Avion** koja nasleđuje klasu Vozilo i:

* Implementiranu metodu daLiIdePoKopnu koja uvek vraća FALSE jer avion ne može da se kreće po kopnu.
* Implementiranu metodu daLiIdePoVodi koja uvek vraća FALSE jer avion ne može da se kreće po vodi.
* Implementiranu metodu daLiIdePoVazduhu koja uvek vraća TRUE jer avion može da leti.
* Metodu upisi koja koristeci klasu **PrintWriter** upisuje u tekstualni fajl “avioni.txt” poruku o tome da je u pitanju avion i podatke o marki i modelu aviona.

1. Glavna klasa **WriteVozila** koja kreira po jedan objekat klase Automobil i Avion. Automobil je marke Mercedes, model E220, dok je avion Boeing model 747. Koristeći klasu **PrintWriter** upisati podatke o oba vozila u tekstualni fajl “vozila.txt”.
2. Napravite javne klase **GeometrijskoTeloException**, **GeometrijskoTelo**, **Kocka**, **Kvadar**i **WriteGeometrijskaTela**sa sledećim uslovima:
3. Klasa **GeometrijskoTeloException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Apstraktna klasa **GeometrijskoTelo** ima:

* Atribut povrsina koji predstavlja površinu geometrijskog tela.
* Atribut zapremina koji predstavlja zapreminu geometrijskog tela.
* Apstraktnu metodu izracunajPovrsinu koja nema ulazne argumente, niti povratnu vrednost.
* Apstraktnu metodu izracunajZapreminu koja nema ulazne argumente, niti povratnu vrednost.
* Metodu getPovrsina koja vraća trenutnu vrednost površine geometrijskog tela. Pre nego što može da vrati tu vrednost, ova metoda mora da pozove metodu izracunajPovrsinu da izracuna povrsinu.
* Metodu getZapremina koja vraća trenutnu vrednost zapremine geometrijskog tela. Pre nego što može da vrati tu vrednost, ova metoda mora da pozove metodu izracunajZapreminu da izracuna zapreminu.

1. Klasa **Kocka** koja nasleđuje klasu GeometrijskoTelo i:

* Atribut duzinaStranice koja predstavlja dužinu stranice kocke.
* Metode getDuzinaStranice i setDuzinaStranice. Duzina stranice mora da bude broj veći od nule. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase GeometrijskoTeloException sa odgovarajućom porukom.
* Realizovanu metodu izracunajPovrsinu koja na osnovu dužine stranice (atribut) postavlja vrednost atributa povrsina po formuli povrsina=duzinaStranice\*duzinaStranice\*6.
* Realizovanu metodu izracunajZapreminu koja na osnovu dužine stranice (atribut) postavlja vrednost atributa zapremina po formuli zapremina=duzinaStranice\*duzinaStranice\*duzinaStranice.

1. Klasa **Kvadar** koja nasleđuje klasu GeometrijskoTelo i:

* Atribut duzina koja predstavlja dužinu kvadra.
* Atribut sirina koji predstavlja širinu kvadra.
* Atribut visina koji predstavlja visinu kvadra.
* Odgovarajuće get i set metode za sva tri atributa. Dužina, visina i širina moraju da budu brojevi veći od nule. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase GeometrijskoTeloException sa odgovarajućom porukom.
* Realizovanu metodu izracunajPovrsinu koja na osnovu dužine, visine i širine (atributi) postavlja vrednost atributa povrsina po formuli:

povrsina = 2\*(duzina\*sirina)+2\*(duzina\*visina)+2\*(sirina\*visina).

* Realizovanu metodu izracunajZapreminu koja na osnovu dužine, visine i širine (atributi) postavlja vrednost atributa zapremina po formuli zapremina = duzina\*sirina\*visina.

1. Glavna klasa **WriteGeometrijskaTela** koja kreira jedan objekat klase Kocka i jedan objekat klase Kvadar. Postaviti dužinu stranice kocke na 25.5cm. Postaviti dužinu, širinu i visinu kvadra na vrednosti 20 10 i 30 respektivno. Koristeći klasu **PrintWriter** upisati podatke o površini i zapremine kocke i kvadra u tekstualni fajl “geometrijskaTela.txt”.
2. Napravite klase **KonvertorValueException**, **KonvertorValute** i **WriteKonvertorValute**sa sledećim uslovima:
3. Klasa **KonvertorValueException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Javna klasa **KonvertorValute** u paketu valute. Ova klasa bi trebalo da ima:

* Privatni atribut kursEvra koji je realan broj i označava vrednost jednog evra u dinarima.
* Privatni atribut kursDolara koji je realan broj i označava vrednost jednog dolara u dinarima.
* Javne set metode za ova dva atributa. Kursevi moraju da budu već od nule, a u slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase KonvertorException sa odgovarajućom porukom.
* Javnu metodu konvertujDinareUEvre koja, na osnovu unetog dinarskog iznosa izračunava i vraća iznos u evrima po trenutnom kursu.
* Javnu metodu konvertujDinareUDolare koja, na osnovu unetog dinarskog iznosa izračunava i vraća iznos u dolarima po trenutnom kursu.
* Metodu ispisiKursnuListu koja ima podrazumevani pristup i primenom klase **PrintWriter** u tekstualni fajl “kursna\_lista.txt” upisati podatke o kursevima za dolar i evro.

1. Glavna klasa **WriteKonvertorValute** u okviru paketa valute koja kreira jedan objekat klase Konvertor valute i postavlja trenutni kurs dolara na 60.34 dinara i evra na 76.89 dinara. Iskoristiti taj objekat za izračunavanje: dolarske protiv vrednosti 600 dinara i protiv vrednosti 4500 dinara u evrima. Primenom klase **PrintWriter** rezultate računanja upisati u tekstualni fajl “konverotor\_valuta.txt”, pa zatim pozvati metodu klase KonvertorValuta za formiranje tekstuelnog fajla sa podacima o kursu.
2. Napravite klase **OsobaException**, **Osoba**, **Zaposleni**i **WriteOsoba**sa sledećim uslovima:
3. Klasa **OsobaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Javna klasa **Osoba** u okviru paketa ljudi koja ima:

* Privatni atribut ime. Početna vrednost ovog atributa je “N”.
* Privatni atribut prezime. Početna vrednost ovog atributa je “N”.
* Privatni atribut JMBG koji predstavlja matični broj datu u vidu String-a.
* Privatnu metodu proveriJMBG koja kao ulazni argument dobija String (koji predstavlja JMBG) i proverava da li taj String ima tačno 13 znakova (matični broj uvek ima tačno 13 cifara). Ako je String različit od null i ima tačno 13 znakova, metoda vraća TRUE a u suprotnom FALSE.
* Javne get i set metode za sva polja. Nedozvoljene vrednosti za ime i prezime su null String-ovi. Metoda setJMBG koristi metodu proveriJMBG da proveri da li je uneti JMBG u redu. U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase OsobaException sa odgovarajućom porukom.
* Zaštićenu (protected) metodu upisi koja primenom klase **PrintWriter** upisuje podatke o osobi u tekstualni fajl “osobe.txt”.

1. Javna klasa **Zaposleni** u okviru paketa zaposleni koja nasleđuje klasu Osoba i ima:

* Privatni atribut plata (realan broj).
* Javne get i set metode za ovaj atribut. Plata mora da bude veća od nule.
* Javnu metodu upisiZaposlenog koja primenom klase **PrintWriter** upisuje sve podatke o zaposlenom u tekstualni fajl “zaposleni.txt”. Metoda poziva metodu upisi klase Osoba.

1. Glavna klasa **WriteOsoba** u paketu provera koja kreira objekat klase Osoba i objekat klase Zaposleni. Dodeliti osobi ime “Pera Perić” i JMBG “3112979710089”, a zaposlenom “Mika Lazic”, “3110970710111” i platu od 40000 dinara. Pozivanjem odgovarajucih metoda, formirati tekstualne fajlove “osobe.txt” I “zaposleni.txt”.
2. Napravite javne klase **MuzickiUredjajException**, **MuzickiUredjaj***,* **RadioKomponenta***,* **CDKomponenta***,* **MuzickiStub**i **WriteMuzickiStub**sa sledećim uslovima:
3. Klasa **MuzickiUredjajException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Apstraktna klasa **MuzickiUredjaj** koja ima:

* Atribut ukljucen koji ima vrednost TRUE ako je uređaj ukljucen, a FALSE ako nije.
* Atribut jacinaTona koji predstavlja jačinu tona na uređaju. Raspon vrednosti za ovaj atribut je od 0 do 40. Početna vrednost je 0.
* Metodu ukljuci koja uključuje uređaj.
* Metodu iskljuci koja isključuje uređaj.
* Metodu pojacajTon koja pojačava ton za 1 (ako je ton bio 5, posle poziva metode ima vrednost 6). Pojačavanje tona se vrši samo ako je ton manji od 40.
* Metodu smanjiTon koja smanjuje ton za 1 (ako je ton bio 5, posle poziva metode ima vrednost 4). Smanjivanje tona se vrši samo ako je ton veći od 0.
* Apstraktnu metodu ispisi koja ništa ne vraća a nema ni ulaznih argumenata.

1. Interfejs **RadioKomponenta** koji ima:

* Metodu promeniStanicu koja ne vraća ništa ali ako ulazni argument prima novu frekvenciju radio stanice.

1. Interfejs **CDKomponenta** koji ima:

* Metodu pustiPesmu koja ne vraća ništa ali kao ulazni argument prima broj pesme na CD-u koji je potrebno pustiti.

1. Klasa **MuzickiStub** koja nasleđuje klasu MuzickiUredjaj, implementira interfejse RadioKomponenta i CDKomponenta i ima:

* Atribut frekvencija koji predstavlja trenutnu frekvenciju radio stanice. Početna vrednost za ovaj atribut je 87.5.
* Atribut brojPesme koji predstavlja broj pesme sa CD-a koju muzički stub trenutno pušta.
* Atribut ukljucenRadio koji ima vrednost TRUE ako muzički stub trenutno pušta muziku sa radija, a inače FALSE.
* Atribut ukljucenCD koji ima vrednost TRUE ako muzički stub trenutno pušta muziku sa CD-a, a inače FALSE.
* Implementiranu metodu promeniStanicu koja uključuje radio komponentu muzičkog stuba (postavlja vrednost atributa ukljucenRadio na TRUE), isključuje CD komponentu (postavlja vrednost atributa ukljucenCD na FALSE), i postavlja vrednost atributa frekvencija na unetu vrednost. Dodeljivanje nove vrednosti za frekvenciju se vrši samo ako je uneta frekvencija u rasponu 87.5-108.0. U suprotnom, potrebno je dodeliti frekvenciju 87.5.
* Implementiranu metodu pustiPesmu koja uključuje CD komponentu muzičkog stuba (postavlja vrednost atributa ukljucenCD na TRUE), isključuje radio komponentu (postavlja vrednost atributa ukljucenRadio na FALSE), i postavlja vrednost atributa brojPesme na unetu vrednost. Dodeljivanje nove vrednosti za broj pesme se vrši samo ako je uneti broj u rasponu 1 do 21. U suprotnom, potrebno je pustiti prvu pesmu sa CD-a.
* Implementiranu metodu upisi koja primenom klase **PrintWriter** u tekstualni fajl “muzickiStub.txt” upisuje poruku o tome da li je muzički stub uključen. Ako jeste, metoda upisuje i poruku o tome koja je trenutna jačina tona. Pored toga, ova metoda bi trebalo da, ako je uključena radio komponenta muzičkog stuba, upise frekvenciju radio stanice, a ako je uključen CD broj pesme koja se trenutno pušta. Ako stub nije uključen ne upisuje se ništa drugo osim poruke da je isključen.

1. Glavna klasa **WriteMuzickiStub** koja kreira jedan objekat klase MuzickiStub, uključuje ga, pojačava ton za 2, podešava frekvenciju radija na 88.9 i kreira tekstualni fajl “muzickiStub.txt” sa odgovarajucim sadrzajem.
2. Napravite javne klase **ProdavnicaException**, **SluzbaNabavke**, **SluzbaProdaje**, **Prodavnica**, **Skladiste**, **Dobavljac**i **WriteProdavnica**sa sledećim uslovima:
3. Klasa **ProdavnicaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Interfejs **SluzbaNabavke** koja ima:

* Metodu dodajRobu koja ne vraća ništa, a kao ulazni argument prima broj komada određenog proizvoda koji je potrebno dodati.

1. Interfejs **SluzbaProdaje**koji ima:

* Metodu prodajRobu koja ne vraća ništa, a kao ulazni argument prima broj komada određenog proizvoda koji je potrebno prodati.

1. Klasa **Prodavnica** koja implementira interfejse SluzbaNabavke i SluzbaProdaje i ima:

* Atribut brojProizvoda koji predstavlja koliko ima komada određenog proizvoda u prodavnici.Početna vrednost za ovaj atribut je 0.
* Implementiranu metodu dodajRobu koja kao ulazni argument prima broj komada određenog proizvoda koji je potrebno dodati i dodaje ga na brojProizvoda. Dodavanje se vrši samo ako je vrednost ulaznog argumenta veća od nule, a u suprotnom baciti izuzetak klase ProdavnicaException sa odgovarajućom porukom .
* Implementiranu metodu prodajRobu koja kao ulazni argument prima broj komada određenog proizvoda koji je potrebno prodati i oduzima ga od atributa brojProizvoda. Prodaja se vrši samo ako je vrednost ulaznog argumenta veća od nule i ako ima dovoljno proizvoda u prodavnici, a u suprotnom baciti izuzetak klase ProdavnicaException sa odgovarajućom porukom.
* Metodu upisi koja primenom klase **PrintWriter** u tekstualni fajl “proizvodi.txt” upisuje koliko ima proizvoda u prodavnici.

1. Klasa **Skladiste** koja implementira interfejs SluzbaNabavke i ima:

* Atribut stanjeNaSkladistu koji predstavlja koliko ima komada određenog proizvoda u prodavnici.Početna vrednost za ovaj atribut je 1000.
* Implementiranu metodu dodajRobu koja kao ulazni argument prima broj komada određenog proizvoda koji je potrebno dodati i dodaje ga na stanjeNaSkladistu. Dodavanje se vrši samo ako je vrednost ulaznog argumenta veća od nule, a u suprotnom baciti izuzetak klase ProdavnicaException sa odgovarajućom porukom.
* Metodu skiniSaStanja koja kao ulazni argument prima broj komada određenog proizvoda koji je potrebno skinuti sa stanja i oduzima ga od atributa stanjeNaSkladistu. Skidanje sa stanja se vrši samo ako je vrednost ulaznog argumenta veća od nule i ako ima dovoljno proizvoda u skladištu, a u suprotnom baciti izuzetak klase ProdavnicaException sa odgovarajućom porukom.

1. Klasa **Dobavljac**koja ima:

* Metodu izvrsiIsporuku koja kao ulazni argument dobija interfejs SluzbaNabavke i broj proizvoda koji je potrebno isporučiti. Isporuka robe se vrši pozivom metode dodajRobu interfejsa SluzbaNabavke sa brojem proizvoda kao ulaznim argumentom.

1. Glavna klasa **WriteProdavnica** koja kreira po jedan objekat klase Prodavnica, Skladiste i Dobavljac. Potrebno je dodati robu (500 komada) u skladište korišćenjem metode izvrsiIsporuku klase Dobavljac. Pozivanjem odgovarajuce metode kreirati tekstualni fajl “proizvodi.txt” sa podacima o broju proizvoda u prodavnici.
2. Napravite klase javne klase **AutomatNovcaException**, **AutomatNovcaInterfejs**, **AutomatNovca**, **NapredniAutomatNovca**i **WriteAutomatNovca**sa sledećim uslovima:
3. Klasa **AutomatNovcaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Interfejs **AutomatNovcaInterfejs** koji ima:

* Metodu uloziNovac koja ne vraća ništa, a ima kao ulazni argument iznos koji se želi uneti (npr. 123,45 din).
* Metodu podigniNovac koja ne vraća ništa, a ima kao ulazni argument iznos koji se želi podići (npr. 123,45 din).

1. Klasa **AutomatNovca** koja implementira interfejs AutomatNovcaInterfejs i ima:

* Atribut stanje koji predstavlja količinu novca koja se nalazi u automatu. Početno stanje je 5000.0 din.
* Implementiranu metodu uloziNovac. Ova metoda vrši ulaganje tj. dodavanje unetog iznosa na trenutno stanje samo ako je uneti iznos veći od nule, a u suprotnom baciti izuzetak klase AutomatNovcaException sa odgovarajućom porukom.
* Implementiranu metodu podigniNovac. Ova metoda vrši podizanje željenog iznosa tj, smanjivanje stanja samo ako je uneti iznos veći od nule i ako u automatu ima dovoljno novca, a u suprotnom baciti izuzetak klase AutomatNovcaException sa odgovarajućom porukom.
* Metodu upisi koja primenom klase **PrintWriter** u tekstualni fajl “stanje.txt” upisuje koliko još novca ima u automatu.

1. Klasa **NapredniAutomatNovca** koja implementira interfejs AutomatNovcaInterfejs i ima:

* Atribut stanje koji predstavlja količinu novca koja se nalazi u automatu. Početno stanje je 20000.0 din.
* Implementiranu metodu uloziNovac. Ova metoda vrši ulaganje tj. dodavanje unetog iznosa na trenutno stanje. Maksimalni iznos koji se može uplatiti je 20.000 dinara. Ako se ovaj iznos prekorači, potrebno je baciti izuzetak klase AutomatNovcaException sa odgovarajućom porukom o tome da je maksimalni iznos koji se može uplatiti 20.000 dinara. Ako je uneta vrednost za iznos manja od nule baciti izuzetak klase AutomatNovcaException sa odgovarajućom porukom.
* Implementiranu metodu podigniNovac. Ova metoda vrši podizanje željenog iznosa tj, smanjivanje stanja. Maksimalni iznos koji se može podići je 10.000 dinara. Ako se ovaj iznos prekorači, potrebno je baciti izuzetak klase AutomatNovcaException sa odgovarajućom porukom o tome da je maksimalni iznos koji se može podići 10.000 dinara. Ako je uneta vrednost za iznos manja od nule baciti izuzetak klase AutomatNovcaException sa odgovarajućom porukom.

1. Glavna klasa **WriteAutomatNovca** koja pravi po jedan objekat klase AutomatNovca i NapredniAutomatNovca i ulaže u prvi 3000 din, a iz drugog podiže 9.999 dinara. Pozivanjem odgovarajuce metode kreirati tekstualni fajl “stanje.txt” sa podacima o stanju na oba automata.
2. Napravite javne klase **AutobusException**, **Autobus**i **WriteAutobus**sa sledećim uslovima:
3. Klasa **AutobusException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **Autobus** koja ima:

* Atribut koji predstavlja niz sedišta u autobusu. Svako sedište može da bude slobodno ili zauzeto. Ako je slobodno, vrednost odgovarajućeg elementa niza je TRUE, a ako je zauzeto, onda je FALSE. Autobus ima tačno 50 sedišta.
* Konstruktor koji postavlja vrednost svih sedišta iz niza na slobodna (TRUE).
* Metodu za uvođenje putnika u autobus. Ova metoda prima kao ulazni argument broj sedišta na koje bi trebalo uvesti putnika (brojevi sedišta su od 0 do 49). Ako je sedište slobodno (TRUE), sedište postaje zauzeto (FALSE) a ako je već bilo zauzeto, baciti izuzetak klase AutobusException sa odgovarajućom porukom.
* Metodu koja proverava da li ima slobodnih mesta u autobusu. Metoda vraća TRUE ako ima slobodnih mesta, u suprotnom FALSE.
* Metodu koja vraća broj slobodnih mesta u autobusu.
* Metodu koja vraća broja zauzetih mesta u autobusu.
* Metodu koja ispisuje status svakog sedišta iz autobusa u obliku “Sedište broj ## je slobodno” ili “Sedište broj ## je zauzeto”.

1. Glavna klasa **WriteAutobus** koja kreira jedan objekat klase Autobus i uvodi u njega tri putnika: na prvo, dvadeseto i poslednje mesto u autobusu. Posle toga, koristeći klasu **PrintWriter** upisati podatke o statusu svih mesta u autobusu u tekstualni fajl “autobus.txt” u fomatu “Sedište broj ## je slobodno” ili “Sedište broj ## je zauzeto”.
2. Napravite javne klase **DNKLanacException**, **DNKLanac**i **WriteDNKLanac**sa sledećim uslovima:
3. Klasa **DNKLanacException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **DNKLanac** koja predstavlja deo DNKLanca čoveka i ima:

* Atribut koji predstavlja niz karika DNK lanca. Svaka karika DNK lanca može da ima samo jednu od vrednosti: 'A', 'C', 'G' ili 'T'.
* Atribut koji predstavlja trenutni broj karika u lancu.
* Konstruktor u kome se lanac kreira tako da mu maksimalni broj karika bude jednak vrednosti koja se prosleđuje konstruktoru u obliku ulaznog argumenta. Ako se desi da je ulazni argument manji ili jednak nula, maksimalni kapacitet treba podesiti na 256 karika.
* Metodu za dodavanje karika u DNK lanac. Nova karika se daje u vidu ulaznog argumenta. Dodavanje se vrši samo ako nova karika ima vrednost 'A', 'C', 'G' ili 'T' i ako u lancu ima mesta (broj karika je manji od maksimalnog kapaciteta). U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase DNKLanacException sa odgovarajućom porukom.
* Metodu koja prebrojava i vraća koliko ima 'A' karika u lancu.
* Metodu koja vraća trenutnu dužinu lanca (broj karika).
* Metodu koja vraća maksimalni kapacitet lanca.
* Metodu koja vraća broj nepopunjenih mesta u lancu.
* Metodu koja ispisuje karike DNK lanca u jednom redu.
* Metodu koja ispisuje karike DNK lanca u jednom redu ali u obrnutom redosledu.

1. Glavna klasa **WriteDNKLanac** koja kreira jedan DNK lanac maksimalne dužine 8 karika i unosi u njega elemente ACCGTTTT. Potrebno je ispisati ovaj DNK lanac u regularnom i obrnutom redosledu. Koristeći klasu **PrintWriter** upisati podatke DNK lancu u regularnom I obrnutom redosledu u tekstualni fajl “DNK.txt”.
2. Napravite javne klase **NizCenaException**, **NizCena**i **WriteNizCena**sa sledećim uslovima:
3. Klasa **NizCenaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **NizCena** koja predstavlja niz cena raznih proizvoda i ima:

* Atribut koji predstavlja niz cena proizvoda (primer cene: 100.5 dinara). Maksimalni kapacitet niza je uvek 100 elemenata.
* Atribut koji predstavlja brojač elemenata niza. Brojač na početku ima vrednost nula jer je niz prazan.
* Metodu za dodavanje nove cene u niz. Dodavanje se vrši samo ako je nova cena veća od nule i ako u nizu ima mesta (brojač je manji od maksimalnog kapaciteta). U slučaju unosa nedozvoljenih vrednosti, baciti izuzetak klase NizCenaException sa odgovarajućom porukom. Ako se ubacivanje izvrši, potrebno je brojač uvećati za jedan.
* Metodu koja izračunava i vraća prosečnu cenu proizvoda. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
* Metodu koja vraća najnižu cenu proizvoda. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
* Metodu koja vraća najvišu cenu proizvoda. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
* Metodu koja vraća razliku između najniže i najviše cene. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
* Metodu koja ispisuje samo one cene proizvoda koje su veće od neke zadate vrednosti. Ta vrednost se unosi u metodu kao ulazni argument. Ukoliko je niz prazan, ispisuje se poruka o tome.

1. Glavna klasa **WriteNizCena** koja kreira jedan objekat klase NizCena. U ovaj niz cena je potrebno uneti sledeće cene: 123.0 , 234.55, 301.0 i 5000.0. Kada se to uradi,potrebno je na ekranu ispisati one cene koje su veće od 300.0 dinara i razliku između najniže i najviše cene. Koristeći klasu **PrintWriter** u tekstualni fajl “cene.txt” upisati one cene koje su veće od 300.0 dinara i razliku između najniže i najviše cene.
2. Napravite javne klase **ParkngMestoException**, **ParkingMesto***,* **Parking**i **TestParking**sa sledećim uslovima:
3. Klasa **ParkngMestoException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Javna klasa **ParkingMesto** koja ima:

* Privatni atribut slobodno koji predstavlja indikator zauzetosti parking mesta. Ovaj indikator ima vrednost TRUE ako je mesto slobodno a FALSE ako nije.
* Privatni atribut registarskiBroj koji predstavlja registarski broj vozila koje se nalazi na tom parking mestu (String).
* Odgovarajuće javne get i set metode za ova dva atributa.

1. Javna klasa **Parking** koja ima:

* Privatni atribut mesta koji predstavlja niz objekata klase ParkingMesto.
* Javni konstruktor koji kao ulazni argument prima broj koji predstavlja kapacitet parkinga tj. ukupan broj parking mesta. Ako je uneti broj veći od nule, potrebno je inicijalizovati atribut mesta na taj kapacitet. Ako je uneti broj nula ili manji od nule, kapacitet parking mesta se postavlja na 40 i baca se izuzetak klase ParkingMestoException sa odgovarajućom porukom. U svakom slučaju je potrebno inicijalizovati svako parking mesto i postaviti ga da bude slobodno.
* Javnu metodu koja vraca registarski broj kola koja se nalaze na prvom parking mestu. Ako je parking mesto slobodno, vraca se poruka o tome.
* Javnu metodu koja vraca registarski broj kola koja se nalaze na poslednjem parking mestu. Ako je parking mesto slobodno, vraca se poruka o tome.
* Javnu metodu koja proverava da li na parkingu ima slobodnih mesta i vraća TRUE ako ima, a FALSE ako nema.
* Javnu metodu koja vraća broj slobodnih parking mesta.
* Javnu metodu koja kao ulazni argument prima registarski broj vozila i proverava da li se to vozilo nalazi na parkingu. Ako se nalazi, metoda vraća TRUE a suprotnom FALSE.
* Javnu metodu za “uvođenje” vozila na parking. Ova metoda kao ulazni argument dobija registarski broj vozila. Prvo je potrebno proveriti da li na parkingu ima slobodnih mesta. Ako ima, potrebno je uvesti vozilo na prvo slobodno mesto. Ako slobodnih mesta nema, vratiti poruku o tome.
* Javnu metodu za “izvođenje” vozila sa parkinga. Ova metoda kao ulazni argument dobija registarski broj vozila. Prvo je potrebno proveriti da li se vozilo sa tim registarskim brojem nalazi na parkingu. Ako se nalazi, potrebno ga je izvesti, tako da parking mesto ponovo postane slobodno. Izvodjenje vozila podrazumeva da se mesto na kome je bilo označi kao slobodno i da se podatak o njegovom registarskom broju ukloni.
* Javnu metodu koja na vraca registarske brojeve svih vozila koja se nalaze na parkingu i broj parking mesta na kome se nalaze. Naravno, vracaju se samo ona parking mesta koja su zauzeta.
* Javnu metodu koja vraca registarske brojeve svih vozila koja se nalaze na parkingu a imaju beogradske tablice.

1. Glavna klasa **WriteParking** koja kreira jedan objekat klase Parking kapaciteta 50 mesta i u njega unosi kola sa tablicama “BG 123-456” i “NS 234-56”. Koristeći klasu **PrintWriter** u tekstualni fajl “parking.txt” upisati reigstarske tablice svih vozila koja su na parkingu, a onda I registarske tablica vozila iz Beograda.
2. Napravite javne klase **AvionException**, **MestoUAvionu***,* **Avion**i **WriteAvion**sa sledećim uslovima:
3. Klasa **AvionException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Javna klasa **MestoUAvionu** koja ima:

* Privatni atribut slobodno koji predstavlja indikator zauzetosti sedišta (mesta). Ovaj indikator ima vrednost TRUE ako je mesto slobodno a FALSE ako nije. Početna vrednost za ovaj atribut je TRUE.
* Privatni atribut imePrezime koji predstavlja ime i prezime putnika koji sedi na tom mestu dato u formatu ”IME PREZIME”. Početna vrednost za ovaj atribut je null.
* Privatni atribut starost koji predstavlja broj godina putnika koji sedi na tom mestu. Početna vrednost za ovaj atribut je 0.
* Odgovarajuće javne get i set metode za ova tri atributa.
* Redefinisanu metodu toString klase Object koja vraća String sa svim podacima o mestu u avionu. Ako je mesto slobodno, vraća se String koji sadrži reč “Slobodno”, a ako nije vraća se String sa imenom i prezimenom i godinama starosti putnika koji se nalazi na tom mestu.

1. Javna klasa **Avion** koja ima:

* Privatni atribut mesta koji predstavlja niz objekata klase MestoUAvionu.
* Javni konstruktor koji inicijalizuje atribut mesta i kreira svako pojedinačno mesto u avionu. Avion ima tačno 120 mesta.
* Javnu metodu uvediPutnika koja kao ulazni argument dobija redni broj mesta, String sa imenom i prezimenom putnika i godine starosti putnika. Ako je mesto sa tim brojem već zauzeto ili je uneti broj mesta van opsega (0-119), potrebno je baciti izuzetak klase AvionException sa odgovarajućom porukom, a u suprotnom “uvesti” putnika na dato mesto i postaviti da mesto nije slobodno.
* Javnu metodu izvediPutnika koja kao ulazni argument dobija redni broj mesta. Ako je mesto sa tim brojem već slobodno ili ako je broj mesta van opsega (0-119), potrebno baciti izuzetak klase AvionException sa odgovarajućom porukom, a u suprotnom “izvesti” putnika sa datog mesta i postaviti mesto da bude slobodno.
* Javnu metodu daLiJeUAvionu koja kao ulazni argument dobija String sa imenom i prezimenom putnika i njegove godine starosti. Metoda vraća TRUE ako se putnik nalazi u avionu, a u suprotnom FALSE.
* Javnu metodu koja izračunava i vraća procentualnu zauzetost sedišta u avionu. Ako je avion popunjen, metoda vraća 100.0%, a ako je prazan 0.0%.
* Javnu metodu koja proverava da li u avionu ima slobodnih sedišta. Ako ima, metoda vraća TRUE, a u suprotnom FALSE.
* Javnu metodu koja izračunava i vraća prosečnu starost putnika kao ceo broj.
* Javnu metodu koja vraća broj godina najstarijeg putnika.
* Javnu metodu koja vraća broj godina najmlađeg putnika.
* Javnu metodu koja na vraca redni broj i podatke o svakom mestu u avionu.
* Javnu metodu koja na vraca podatke o svim putnicima u avionu koji se prezivaju “Jovanovic”.

1. Glavna klasa **writeAvion** koja kreira jedan objekat klase Avion. Potrebno je u avion uvesti putnike: “Jovan Jovanovic” (53 godine), “Milos Milosevic” (23 godine) i “Ana Jovanovic” (22 godine) na 21, 31 i 41 mesto. Koristeći klasu **PrintWriter** u tekstualni fajl “putnici.txt” upisati sve podatke o putnicima, samo podatke o putnicima koji se prezivaju Jovanovic I prosečnu starost putnika.
2. Napravite javne klase **NizCelihBrojevaException**, **NizCelihBrojeva**i **WriteNizCelihBrojeva**sa sledećim uslovima:
3. Klasa **NizCelihBrojevaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **NizCelihBrojeva** koja ima:

* Atribut koji predstavlja niz celih brojeva.
* Atribut koji predstavlja brojač elemenata niza. Brojač na početku ima vrednost nula jer je niz prazan.
* Konstruktor u kome se niz kreira tako da mu maksimalni kapacitet bude jednak vrednosti koja se prosleđuje konstruktoru u obliku ulaznog argumenta.
* Metodu za dodavanje elemenata u niz. Ova metoda prima kao ulazni argument broj koji je potrebno dodati u niz. Pre nego što se izvrši dodavanje, proverava se da li je kapacitet prekoračen (da li je brojač dostigao maksimalni kapacitet niza). Ako je kapacitet prekoračen, baciti izuzetak klase NizCelihBrojevaException sa odgovarajućom porukom. Ako nije, broj se dodaje na prvo slobodno mesto u nizu. Ako je element dodat, potrebno je u okviru metode uvećati vrednost brojača za jedan.
* Metodu koja sabira vrednosti prvog i poslednjeg elementa niza i vraća rezultat. Ako je niz prazan, vraca se poruka o tome.
* Metodu koja sabira samo pozitivne elemente niza i vraća njihovu vrednost.
* Metodu koja množi samo negativne elemente niza i vraća njihov proizvod.
* Metodu koja vraća broj ponavljanja nekog broja u nizu. Broj se prosleđuje metodi u vidu ulaznog argumenta.
* Metoda koja vraca članove niza koji su parni brojevi.

1. Glavna klasa **WriteNizCelihBrojeva** koja kreira tri objekta klase NizCelihBrojeva: prvi kapaciteta 3 člana, drugi kapaciteta 5 članova, a treći kapaciteta 10 članova. U prvi niz je potrebno ubaciti elemente 1, -1 i 3 u drugi elemente 34 i 45 a u treći elemente 56, 67 i – 89. Koristeći klasu **PrintWriter** u tekstualni fajl “nizCelihBrojeva.txt” upisati zbir prvog i poslednjeg elementa prvog niza, a drugi i treci niz upisati u obrnutom redosledu.
2. Napravite klase **TemperaturaMoraException**, **TemperatureMora**i **WriteTemperatureMora**sa sledećim uslovima:
3. Klasa **TemperaturaMoraException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **TemperatureMora** koja ima:

* Atribut koji predstavlja niz dnevnih merenja temperature za jedno more u toku godinu dana (prvi element niza je temperatura za 1. januar, sledeći element za 2. januar itd. a poslednji element za 31.decembar). Vrednost se izražava u stepenima (npr. 12,3).Inicijalizovati niz na kapacitet 365.
* Statičku konstantu KRITICNA\_TEMPERATURA koja ima vrednost 45oC.
* Metodu koja unosi novu temperaturu u niz. Metoda kao parametre dobija temperaturu i redni broj dana (u toku godine) na koji se ta temperatura odnosi. Redni broj se unosi u rasponu od 1 do 365. Ako je temperatura manja od nule ili redni broj van raspona, baciti izuzetak klase TemperaturaMoraException sa odgovarajućom porukom.
* Metodu koja na ekranu ispisuje prosečnu mesečnu temperaturu za jun. Prosečna mesečna temperatura se dobija izračunavanjem proseka dnevnih temperatura za taj mesec.
* Metodu koja kao ulaz dobija niz sa 365 temperatura i kopira te vrednosti u niz temperatura mora ali u obrnutom redosledu.
* Metodu koja proverava da li more može da ima živ svet u sebi ili ne i vraća TRUE ako može, a FALSE inače. Ako je temperatura mora u bilo kojem trenutku u toku godine bila pet dana uzastopce iznad konstante KRITICNA\_TEMPERATURA, to znači da je more previše toplo i da u njemu ne može da ima života. U suprotnom, more može da ima živ svet u sebi.

1. Glavna klasa **WriteTemperatureMora** koja pravi dva objekta klase TemperatureMora. U prvi objekat uneti temperaturu 41 C za 1. februar, a u drugo more temperaturu 23,5 C za 25. januar. Koristeći klasu **PrintWriter** u tekstualni fajl “temperaturaMora.txt” upisati da li oba mora mogu da imaju živ svet u sebi ili ne.
2. Napravite klase **KoncertnaDvoranaException**, **KoncertnaDvorana**i **WriteKoncertnaDvorana**sa sledećim uslovima:
3. Klasa **KoncertnaDvoranaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **KoncertnaDvorana** koja ima:

* Statičku konstantu KAPACITET koja ima vrednost 300.
* Atribut koji predstavlja niz sedišta u dvorani. Svako sedište može da bude slobodno ili zauzeto. Akoje slobodno, vrednost odgovarajućeg elementa niza je TRUE, a ako je zauzeto, onda je FALSE. Koncertna dvorana ima tačno onoliko sedišta kolika je vrednost konstante KAPACITET. Odmah inicijalizovati niz.
* Konstruktor koji postavlja vrednost svih sedišta iz niza na slobodna.
* Metodu koja uvodi gledaoca na sedište. Broj sedišta je dat kao ulazni argument metode (unose se brojevi od 1 do 300) . Prvo se proverava da li je broj sedišta u granicama i da li je sedište već zauzeto pa, ako je sve u redu, sedište postaje zauzeto. Ako je sedište već bilo zauzeto ili broj sedišta nije u granicama, baciti izuzetak klase KoncertnaDvoranaException sa odgovarajućom porukom.
* Metodu koja proverava da li je dvorana dovoljno popunjena da bi se koncert isplatio i vraća TRUE ako jeste, a u suprotnom FALSE. Dvorana je dovoljno popunjena ako su ispunjena tri uslova: procenat zauzetosti sedišta iz prve polovine dvorane (prvih 150 sedišta) je veći od 50%, procenat zauzetosti sedišta iz druge polovine dvorane (ostalih 150 sedišta) je veći od 40% i procenat zauzetosti sedišta iz cele dvorane je veći od 55%.
* Metodu koja vraća koliko ima slobodnih sedišta u svakom redu dvorane. Zna se da su svi redovi iste veličine i da sadrže tačno po 10 sedišta. Prema tome, prvih 10 sedišta iz niza su zapravo u prvom redu, sledećih 10 sedišta u drugom redu itd. Ispis bi trebalo da bude u formatu: „U redu br. \_\_ ima \_\_ slobodnih sedišta“.

1. Glavna klasa **WriteKoncertnaDvorana** koja kreira jedan objekat klase KoncetnaDvorana i uvodi gledaoce na sedišta broj 21, 23 i 24. Koristeći klasu **PrintWriter** u tekstualni fajl “koncertnaDvorana.txt” upisati poruku o tome da li je dvorana dovoljno popunjena da bi se koncert isplatio.
2. Napravite klase **ZaradaDvoraneException**, **ZaradaDvorane***,* **KoncertnaDvorana**i **WriteKoncertnaDvorana**sa sledećim uslovima:
3. Klasa **ZaradaDvoraneException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **ZaradaDvorane** koja ima:

* Atribut zarada koji predstavlja zaradu ostvarenu na koncertu (npr. 20000 dinara).

1. Klasa **KoncertnaDvorana** koja ima:

* Atribut koji predstavlja niz sedišta u dvorani. Svako sedište može da bude slobodno ili zauzeto. Ako je slobodno, vrednost odgovarajućeg elementa niza je TRUE, a ako je zauzeto, onda je FALSE.
* Atribut zaradaDvorane koji predstavlja objekat klase ZaradaDvorane. Odmah inicijalizovati atribut (objekat).
* Konstruktor koji kao ulazni argument prima kapacitet sale i inicijalizuje niz sedišta na taj kapacitet. Ako je uneti kapacitet sale jednak ili manji od nule potrebno je kapacitet postaviti na 120 mesta baciti izuzetak klase ZaradaDvoraneException sa odgovarajućom porukom. Postaviti vrednosti svih sedišta iz niza na slobodna.
* Metodu koja uvodi gledaoca na sedište. Broj sedišta je dat kao ulazni argument metode. Prvo se proverava da li je broj sedišta u granicama (brojevi se poklapaju sa indeksima niza) i da li je sedište već zauzeto, pa ako je sve u redu sedište postaje zauzeto. Ako je sedište već bilo zauzeto ili broj sedišta nije u granicama, ne dešava se ništa.
* Metodu koja izračunava zaradu koncertne dvorane ako se zna da karte za sedišta iz prve polovine sale koštaju 500 dinara, a da karte za sedišta iz druge polovine sale koštaju 250 dinara. Zarada se računa samo za ona sedišta koja su zauzeta. Kada metoda izračuna zaradu, ovu vrednost unosi u atribut zarada klase zaradaDvorane.
* Metodu koja proverava da li se u dvorani mogu naći tri susedna sedišta koja su slobodna i nalaze se u istom redu i vraća TRUE ako može, a FALSE ako ne može. Zna se da su svi redovi iste veličine I da sadrže tačno po 10 sedišta. Prema tome, prvih 10 sedišta iz niza su zapravo u prvom redu, sledećih 10 sedišta u drugom redu itd.

1. Glavna klasa **WriteKoncertnaDvorana** koja kreira jedan objekat klase KoncertnaDvorana kapaciteta 40 mesta i uvodi gledaoce na sedišta broj 0, 20 i 40. Koristeći klasu **PrintWriter** u tekstualni fajl “koncertnaDvorana.txt” upisati zaradu koncertne dvorane.
2. Napravite klase **TezinaClanovaEkipeException**, **TezinaClanovaEkipe** i **WriteTezinaClanovaEkipe**sa sledećim uslovima:
3. Klasa **TezinaClanovaEkipeException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **TezinaClanovaEkipe** koja ima:

* Atribut koji predstavlja niz težina članova bokserske ekipe u kilogramima (npr. 89,5 kg). Ekipa može da ima maksimalno 20 boksera. Niz inicijalizovati odmah.
  + Atribut koji predstavlja broj članova ekipe. Postaviti da početna vrednost bude nula.
  + Statičku konstantu MIN\_TEZINA koja predstavlja minimalnu težinu boksera i iznosi 40kg.
  + Konstruktor koji postavlja vrednosti svih 20 elemenata niza na -1.
* Metodu koja dodaje težinu novog člana ekipe u niz. Težina se dobija u vidu ulaznog argumenta I dodaje se u niz samo ako niz nije popunjen do maksimuma i ako je uneta težina veća od vrednosti konstante MIN\_TEZINA. Ako je niz popunjen ili ako težina nije u granicama, baciti izuzetak klase TezinaClanovaEkipeException sa odgovarajućom porukom.
* Metodu koja vraca koji procenat boksera iz ekipe pripada lakoj, koji srednjoj i koji teškoj kategoriji. Ako je težina boksera manja od 65kg on pripada lakoj kategoriji. Ako je težina 65kg ili više, ali manja od 85kg on pripada srednjoj kategoriji. Ako je težina 85kg ili više, on pripada teškoj kategoriji. Ispis bi trebalo da bude u formatu „\_\_\_ kategoriji pripada \_\_\_% boksera“.
* Statičku metodu koja kao ulazni parametar dobija niz realnih brojeva koji predstavlja niz težina tri bokserske ekipe. Ovaj niz je popunjen do maksimalnog kapaciteta. Niz težina prve ekipe počinje od prvog elementa a završava se elementom koji ima vrednost -1 i koji ne predstavlja konkretnu težinu već samo graničnik. Niz težina druge ekipe počinje posle ovog graničnika, a završava se drugim graničnikom koji takođe ima vrednost -1. Ostatak niza čine težine članova treće ekipe. Metoda bi trebalo da sastavlja i vraća (kao povratnu vrednost) novi niz koji se sastoji samo od težina onih boksera koji pripadaju drugoj ekipi.

1. Glavna klasa **WriteTezinaClanovaEkipe** koja kreira dva objekta klase TezineClanovaEkipe. U prvu ekipu dodati težine 100.0, 55.5 i 44.5 a u drugu ekipu težine 66.6, 44.3 i 150.0 kg. Ispisati na ekranu procenat boksera lake, srednje i teške kategorije za prvu ekipu. Koristeći klasu **PrintWriter** u tekstualni fajl “tezinaClanovaEkipe.txt” upisati procenat boksera lake, srednje i teške kategorije za prvu ekipu.
2. Napravite javne klase **TipEkipeException**, **TipEkipe**, **TezineClanovaEkipe** i **WriteTezinaClanovaEkipe**sa sledećim uslovima:
3. Klasa **TipEkipeException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **TipEkipe** koja ima:

* Atribut oznaka koji predstavlja oznaku težinske kategorije (npr. 'L' za laku kategoriju).

1. Klasa **TezineClanovaEkipe** koja ima:

* Atribut koji predstavlja niz težina članova bokserske ekipe u kilogramima (npr. 89,5 kg).
* Atribut koji predstavlja broj članova ekipe. Postaviti da početna vrednost bude nula.
* Atribut kategorija koji predstavlja objekat klase TipEkipe.
* Konstruktor koji kao ulazni argument prima maksimalni broj članova ekipe i oznaku težinske kategorije. Konstruktor bi trebalo da prvo inicijalizuje atribut kategorija i u njega unese oznaku težinske kategorije. Konstruktor onda inicijalizuje niz na uneti kapacitet samo ako je uneti broj članova ekipe veći od nule. Ako to nije slučaj, niz se inicijalizuje na kapacitet 20.
* Metodu koja dodaje težinu novog člana ekipe u niz. Težina se dobija u vidu ulaznog argumenta I dodaje se u niz samo ako niz nije popunjen do maksimuma i ako je uneta težina u granicama. Težinske granice za laku kategoriju su od 45 do 65kg (ne obuhvatajući 65kg), za srednju kategoriju od 65 do 85kg (ne obuhvatajući 85kg), a za tešku kategoriju 85kg i više. Ako je oznaka kategorije (atribut oznaka atributa kategorija) jednaka 'L', primenjivati granice za laku kategoriju, ako je 'S' za srednju, a ako je 'T' za tešku kategoriju.
* Statičku metodu koja kao ulazne parametre dobija dva niza realnih brojeva koji predstavljaju težine dve bokserske ekipe. Oba niza su popunjena do maksimalnog kapaciteta a ne moraju da budu iste dužine. Metoda bi trebalo da vraća TRUE ako postoje bar dva boksera koji su iste težine pri čemu jedan pripada prvoj, a drugi drugoj ekipi, a FALSE ako to nije slučaj.

1. Glavna klasa **WriteTezineClanovaEkipe** koja kreira tri objekta klase TezineClanovaEkipe sa kapacitetom od 20, 30 i 40 članova. Prvu ekipu čine bokseri lake kategorije, a drugu i treću bokseri srednje kategorije`-. Uneti u drugu ekipu težine 66, 69 i 84kg. Koristeći klasu **PrintWriter** u tekstualni fajl “tezinaClanovaEkipe.txt” upisati tezine clanova druge ekipe.
2. Napravite javne klase **NizOcenaException**, **NizOcena** i **WriteNizOcena** sa sledećim uslovima:
3. Klasa **NizOcenaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **NizOcena** koja ima:

* Atribut koji predstavlja niz ocena učenika u odeljenju. Svaki učenik ima po jednu ocenu, ocena se predstavlja jednim slovom, pri čemu slovo 'A' znači odličan, slovo 'B' vrlo dobar, slovo 'C' dobar, slovo 'D' dovoljan i slovo 'F' znači nedovoljan. Odeljenje ima maksimalno 30 đaka. Niz inicijalizovati odmah.
* Atribut koji predstavlja broj ocena (đaka). Početna vrednost je nula.
* Statičku konstantu OCEKIVANA\_OCENA koja predstavlja ocenu za koju se očekuje da će je imati najveći broj đaka - ocena 'B'.
* Konstruktor koji postavlja vrednosti svih 30 elemenata niza na 'N'.
* Metodu za unošenje nove ocene. Ocena je ulazni argument metode, a unošenje se vrši samo ako niz nije pun i ako ocena ima vrednost 'A' , 'B', 'C', 'D' ili 'F'. Ako je bilo koji od ovih uslova prekoračen baciti izuzetak klase NizOcenaException sa odgovarajućom porukom.
* Metodu koja proverava da li je postignuta očekivana ocena u odeljenju ili ne i vraća TRUE ako jeste a FALSE ako nije. Očekivana ocena je postignuta ako je broj đaka koji imaju ocenu koja je jednaka konstanti OCEKIVANA\_OCENA istovremeno veći od: broja đaka sa ocenom 'A', broja đaka sa ocenom 'C', broja đaka sa ocenom 'D' i broja đaka sa ocenom 'F'.
* Statičku metodu koja kao ulazni argument dobija niz znakova koji predstavljaju ocene nekoliko odeljenja. Niz je popunjen do maksimalnog kapaciteta. Svako odeljenje ima tačno po 30 đaka. Prema tome, prvih 30 elemenata niza predstavljaju ocene đaka iz prvog odeljenja, sledećih 30 elemenata ocene đaka iz drugog odeljenja itd. Metoda vraca koliko đaka u svakom odeljenju ima ocenu 'A' i to u formatu „U odeljenju br. \_\_ ima \_\_ đaka sa ocenom A“.

1. Glavna klasa **WriteNizOcena** koja kreira objekat klase NizOcena i unosi u njega ocene: 'A', 'A', 'B', 'F', 'F' i

'D'. Koristeći klasu **PrintWriter** u tekstualni fajl “nizOcena.txt” upisati da li je postignuta očekivana ocena.

1. Napravite javne klase **StatistikaException**, **Statistika**, **NizOcena** i **WriteNizOcena** sa sledećim uslovima:
2. Klasa **StatistikaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasu **Statistika** koja ima:

* Atribut procenatProlaznosti koji predstavlja procenat đaka koji imaju prelazne ocene (npr. 45.3%).

1. Klasu **NizOcena** koja ima:

* Atribut koji predstavlja niz ocena učenika u odeljenju. Svaki učenik ima po jednu ocenu, i ocena se predstavlja jednim slovom, pri čemu slovo 'A' znači odličan, slovo 'B' vrlo dobar, slovo 'C' dobar, slovo 'D' dovoljan i slovo 'N' znači nedovoljan.
* Atribut koji predstavlja broj ocena (đaka). Početna vrednost je nula.
* Atribut statistikaOdeljenja koja predstavlja objekat klase Statistika. Odmah inicijalizovati ovaj atribut (objekat).
* Konstruktor koji kao ulazni argument prima maksimalni broj ocena (đaka) i inicijalizuje niz na taj kapacitet. Ako je uneti broj đaka jednak ili manji od nule, baciti izuzetak klase StatistikaException sa odgovarajućom porukom i postaviti maksimalni kapacitet na 25 đaka.
* Metodu za unošenje nove ocene. Ocena je ulazni argument metode, a unošenje se vrši samo ako niz nije pun i ako ocena ima vrednost 'A' , 'B', 'C', 'D' ili 'N'. Ako je bilo koji od ovih uslova prekoračen, baciti izuzetak klase StatistikaException sa odgovarajućom porukom.
* Metodu koja vraca koliko đaka ima koju ocenu.
* Metodu koja izračunava procenat prolaznosti odeljenja (procenat đaka koji imaju prelazne ocene tj. ocene različite od 'N') i unosi ovaj procenat u atribut procenatProlaznosti atributa (objekta) statistikaOdeljenja.
* Statičku metodu koja kao ulazni argument dobija niz znakova koji predstavljaju ocene nekoliko odeljenja. Niz je popunjen do maksimalnog kapaciteta. Svako odeljenje ima tačno po 25 đaka. Prema tome, prvih 25 elemenata niza predstavljaju ocene đaka iz prvog odeljenja, sledećih 25 elemenata ocene đaka iz drugog odeljenja itd. Metoda bi trebalo da proverava da li su u nekom odeljenju svi đaci pali na popravni (svih 25 đaka ima ocenu 'N'). Ako postoji bar jedno takvo odeljenje metoda vraća TRUE, a inače FALSE.

1. Glavna klasa **WriteNizOcena** koja kreira kreira objekat klase NizOcena maksimalnog kapaciteta 10 đaka i unosi u njega ocene: 'A', 'B', 'C', 'N', 'N' i 'D'. Koristeći klasu **PrintWriter** u tekstualni fajl “nizOcena.txt” upisati koliko đaka ima koju ocenu.
2. Napravite klase **DnevneTemperatureException**, **DnevneTemperature** i **WriteDnevneTemperature** sa sledećim uslovima:
3. Klasa **DnevneTemperatureException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **DnevneTemperature** koja ima:

* Atribut koji predstavlja niz dnevnih temperatura za jedan grad (npr. 12 C) i to za svaki dan u toku godine (prvi element niza je temperatura za 1. januar, sledeći element za 2. januar itd... a poslednji element za 31. decembar). Inicijalizovati niz na kapacitet 365.
* Statičku konstantu MIN\_TEMP koja iznosi -50 C.
* Metodu koja unosi novu temperaturu u niz. Metoda kao parametre dobija temperaturu i redni broj dana (u toku godine) na koji se ta temperatura odnosi. Redni broj se unosi u rasponu od 1 do 365. Ako je uneta temperatura manja od MIN\_TEMP ili ako je redni broj dana van traženog raspona, baciti izuzetak klase DnevnaTemperaturaException sa odgovarajućom porukom.
* Metodu koja vraca zbir dnevnih temperatura za mesec februar. Godina nije prestupna.
* Metodu koja vraca dnevne temperature za dane vikenda (subota i nedelja) i to za celu godinu. Zna se da je godina počela ponedeljkom tj. 1. januar je ponedeljak.
* Metodu koja na osnovu niza dnevnih temperatura sastavlja i vraća niz prosečnih sedmičnih temperatura. Računati samo prva 364 dana u godini tj. samo 52 cele nedelje.

1. Glavna klasa **WriteDnevneTemperature** koja kreira jedan objekat klase DnevneTemperature. U ovaj objekat uneti temperature 12 C, 34 C i 8 C i to za 30. , 150. i 365. dan u godini. Koristeći klasu **PrintWriter** u tekstualni fajl “koncertnaDvorana.txt” upisati temperaturu za mesec februar.
2. Napravite klase **AvionException**, **Avion** i **WriteAvion** sa sledećim uslovima:
3. Klasa **AvionException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasu **Avion** koja ima:

* Atribut koji predstavlja niz sedišta ekonomske klase u avionu. Sedište je slobodno ako je vrednost elementa na tom mestu TRUE a zauzeto ako je vrednost elementa na tom mestu FALSE.
* Atribut koji predstavlja niz sedišta poslovne klase u avionu. Sedište je slobodno ako je vrednost elementa na tom mestu TRUE a zauzeto ako je vrednost elementa na tom mestu FALSE.
* Konstruktor koji kao ulazne argumente prima ukupan broj sedišta ekonomske i ukupan broj sedišta poslovne klase i inicijalizuje oba niza na odgovarajuće kapacitete. Ako je neki od unetih brojeva sedišta manji od nule (bilo poslovnih bilo ekonomskih), postaviti da avion ima 120 mesta ekonomske klase i 60 mesta poslovne klase i baciti izuzetak klase AvionException sa odgovarajućom porukom. U svakom slučaju, postaviti da sva sedišta u avionu budu slobodna.
* Metodu koja uvodi putnika na sedište. Metoda kao parametre prima boolean vrednost (koja ima vrednost TRUE ako putnika treba smestiti u poslovnu klasu, a FALSE ako ga teba smestiti u ekonomsku klasu), kao i broj sedišta (brojevi kreću od nule). Metoda smešta putnika na zadato mesto iz odgovarajuće klase. Ako to mesto nije slobodno, baciti izuzetak klase AvionException sa odgovarajućom porukom.
* Metodu koja izračunava i vraca izveštaj o letu. Vraca se procentualna zauzetost sedišta poslovne klase, procentualna zauzetost sedišta ekonomske klase kao i prihod od svih prodatih karata. Karte za sedišta iz poslovne klase koštaju 500 evra, a karte za sedišta iz ekonomske klase koštaju 250 evra. Zarada se računa samo za ona sedišta koja su zauzeta.
* Metodu koja proverava da li se u ekonomskoj klasi mogu naći tri susedna sedišta koja su slobodna i nalaze se u istom redu i vraća TRUE ako može, a FALSE ako ne može. Zna se da su svi redovi iste veličine i da sadrže tačno po 6 sedišta. Prema tome, prvih 6 sedišta iz niza su zapravo u prvom redu, sledećih 6 sedišta u drugom redu itd.

1. Glavna klasa **WriteAvion** koja koja kreira tri objekta klase Avion. Prvi avion ima 120 sedišta ekonomske klase i 60 sedišta poslovne klase, drugi ima po 50 sedišta obe klase a treći ima samo 30 sedišta poslovne klase. Uneti putnike u treći avion i to na sedišta broj 1, 5 i 15 iz poslovne klase. Koristeći klasu **PrintWriter** u tekstualni fajl “avion.txt” upisati poruku o procentualnoj zauzetosti sedista poslovne i ekonomske klase, kao I podatak o tome da li se u ekonomskoj klasi mogu naci tri susedna sedista.
2. Napravite klase **TrajektException**, **Trajekt** i **WriteTrajekt** sa sledećim uslovima:
3. Klasa **TrajektException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasu **Trajekt** koja ima:

* Atribut koji istovremeno predstavlja niz sedišta za putnike i mesta za automobile u trajektu. Sedište tj. mesto je slobodno ako je vrednost elementa niza TRUE a zauzeto ako je vrednost elementa niza FALSE. Sedišta za putnike se nalaze na početku niza, a njihov broj je dat u atributu ukupanBrojSedišta. Nakon poslednjeg sedišta, ostatak niza čine mesta za automobile.
* Atribut ukupanBrojSedišta koji predstavlja ukupan broj sedišta u trajektu.
* Konstruktor koji kao ulazne argumente prima ukupan broj sedišta i ukupan broj mesta za automobile. Konstruktor inicijalizuje niz na kapacitet koji je jednak zbiru ova dva broja i postavlja vrednost atributa ukupanBrojSedišta na uneti broj sedišta. Ako je neki od unetih brojeva manji od nule, postaviti da trajekt ima 120 mesta za automobile, 0 sedišta i baciti izuzetak klase TrajektException sa odgovarajućom porukom. U svakom slučaju, postaviti da sva sedišta i mesta u trajektu budu slobodna.
* Metodu koja uvodi auto na mesto u trajektu. Metoda kao parametar prima redni broj mesta (uneti brojevi se kreću od 0 do ukupnog broja mesta za automobile). Obratiti pažnju na to da elementi niza koji predstavljaju mesta za automobile počinju tek posle svih sedišta za putnike. Ako traženo mesto nije slobodno, baciti izuzetak klase TrajektException sa odgovarajućom porukom.
* Metodu koja vraca izveštaj o trajektu. Vraca se broj slobodnih i zauzetih sedišta, broj slobodnih i zauzetih mesta za automobile, kao i to da li se pokretanje tako napunjenog trajekta isplati ili ne. Da bi se put isplatio, trajekt mora da ima bar 66% zauzetih sedišta, ali i bar 45% zauzetih mesta za automobile.
* Metodu koja proverava da li postoji dovoljno mesta na trajektu da se preveze autobus i vraća TRUE ako može a FALSE inače. Autobus može da se preveze ako negde na trajektu postoji pet susednih mesta za automobile koja su sva slobodna.

1. Glavna klasa **WriteTrajekt** koja kreira tri objekta klase Trajekt. Prvi ima 120 sedišta i 20 mesta za automobile, drugi ima 50 sedišta i 50 mesta, a treći ima samo 30 sedišta. Proveriti da li se u prvi trajekt može smestiti autobus tj. da li na njemu ima dovoljno mesta za to. Koristeći klasu **PrintWriter** u tekstualni fajl “trajekt.txt” upisati poruku o dobijenom rezultatu.
2. Napravite klase **BankaException**, **PolitikaBanke**, **Banka** i **WriteBanka** sa sledećim uslovima:
3. Klasa **BankaException** predstavlja neproveravani izuzetak (nasleđuje klasu **RuntimeException**) i ima:

* Javni konstruktor koji kao parametar prima poruku greške i poziva odgovarajući konstruktor nadklase prosleđujući mu parametar.

1. Klasa **PolitikaBanke** koja ima:

* Statičku konstantu KAMATNA\_STOPA koja iznosi 5,8 (%).

1. Klasa **Banka** koja ima:

* Atribut krediti koji predstavlja niz čiji su elementi iznosi kredita. Dakle, svaki element niza je iznos za jedan kredit koji je odobrila banka (npr. 15500 eur)
* Atribut brojKredita koji predstavlja broj trenutno odobrenih kredita u banci. Početna vrednost je 0.
* Konstruktor koji kao ulazni argument prima maksimalan broj kredita i inicijalizuje niz na taj kapacitet. Takođe, potrebno je postaviti da vrednosti svih elemenata niza budu 0.
* Metodu za unos novog kredita u niz i to na prvo slobodno mesto. Kredit se unosi kao parametar metode. Ako je uneta vrednost nula, manja od nule ili ako je niz pun, baciti izuzetak klase BankaException sa odgovarajućom porukom.
* Metodu koja računa i vraća kolika je prosečna mesečna rata za one kredite čiji je iznos veći od 10.000 eur. Svi takvi krediti su odobreni na period od godinu dana, pa se mesečna rata za jedan kredit računa po formuli: RATA = (IZNOS \* (100 + KAMATNA\_STOPA)) / (100\*12). Uzeti kamatnu stopu iz klase PolitikaBanke i obratiti pažnju na prazna mesta u nizu.

1. Glavna klasa **WriteBanka** koja kreira jedan objekat klase Banka kapaciteta 40 kredita i uneti u njega kredite iznosa 3.000, 2.000 i 5.000e. Koristeći klasu **PrintWriter** u tekstualni fajl “banka.txt” upisati prosečnu mesečnu ratu za kredite koji iznose više od 10.000e.