

## Zadaci – OOP Java

### Zadatak 1.

U programskom jeziku Java implementirati:

Apstraktni tip **Proizvođač** koji sadrži:

- privatni string atribut **naziv**
- privatni string atribut **adresa**
- privatni niz stringova **proizvodi**
- privatni niz float vrednosti **cene**
- konstruktor koji prima naziv i setuje ga
- metodu **dodajProizvod** koja prima naziv i cenu proizvoda, pa ih dodaje u odgovarajuće nizove
- apstraktnu metodu **getCena** koja prima string i vraća float
- metodu **prebroj** koja vraća ceo broj koji predstavlja broj elemenata niza proizvodi
- metodu **izbaciProizvod** koja prima naziv proizvoda, nalazi ga u nizu i izbacuje ga (odgovarajuća cena takođe mora biti izbačena)
- string reprezentaciju koja sadrži naziv, adresu firme kao i spisak svih proizvoda sa cenama
- **gettere** i **settere** za sve promenljive osim **settera** za nizove

Tip **Poljoprivrednik** koji predstavlja proizvođača i ima:

- privatni atribut **brojHektara**
- konstruktor koji prima ime i broj hektara i setuje ih
- metodu **getCena** koja prima naziv proizvoda, traži ga u nizu proizvoda, nalazi odgovarajuću cenu i cenu umanjuje za po 2% sa svaki hektar preko prvog

Tip **Firma** koji predstavlja proizvođača i ima:

- privatni string atribut **PIB**
- konstruktor koji prima ime i PIB i setuje ih
- metodu **getCena** koja prima naziv proizvoda, traži ga u nizu proizvoda, nalazi odgovarajuću cenu i vrati je

Testna klasa koja:

- kreira jednog poljoprivrednika Marka Markovića sa brojem hektara 2 i jednu firmu Poljolink sa PIB-om 20133
- Marku dodaje na spisak proizvoda grašak i repu i setuje im cene na 50 i 100
- firmi Poljolink dodaje šećer i repu po cenama 400 i 100
- formira niz Proizvođača i smešta u njega Marka i Poljolink
- ispisuje string reprezentaciju svih elemenata niza
- ispisuje nazive i cene proizvoda koje Marko proizvodi

### Zadatak 2.

U programskom jeziku Java implementirati:

Tip **Brojač** ( simulacija **AutoNumber**) koji ima:

- privatni celobrojni podatak **vrd**
- metodu **getNext** koja uvećava **vrd** za 1 i vraća je
- metodu **reset** koja vraća **vrd** na početnu vrednost

Tip **Greška** koji predstavlja Exception i u konstruktoru prima string koji se postavlja kao vrednost message promenljive.

Apstraktan tip **Proizvod** ima:

- privatni string atribut **naziv**
- privatni celobrojni podatak **ID** (svaki proizvod ima svoj ID, ID se dodeljuje počevši od 1 do 50)
- privatnu promenljivu **b** tipa **Brojač** koju koristi za preuzimanje sledeće vrednosti za ID
- privatnu char promenljivu **tip**
- javni niz **listaProizvoda** koji pripada klasi i koji sadrži reference na sve registrovane proizvode
- javnu metodu klase koja vraća referencu na **listaProizvoda**
- konstruktor koji prima naziv i tip proizvoda i baca izuzetak tipa **Greška**. Konstruktor proverava da li već postoji proizvod pod datim imenom i ako postoji baca izuzetak tipa **Greška** sa odgovarajućim komentaram, a ako ne postoji korišćenjem **b** određuje vrednost za ID i postavlja je, zatim setuje naziv i tip.
- metoda **dodajUListuProizvoda** koja proizvod koji dobija u argumentu dodaje u **listaProizvoda**
- metoda **nađiProizvod** koja dobije string naziv i u **listaProizvoda** traži proizvod sa datim nazivom i vraća objekat tipa **Proizvod** ako ga je našao a null ako nije.
- **getter** i **setter** za naziv
- **getter-e** za ID i tip
- string reprezentaciju oblika ID naziv

Tip proizvod **Industrijski** koji ima:

- konstruktor koji u argumentu prima naziv i setuje ga, a pored toga setuje i tip na **I**

Tip proizvod **Poljoprivredni** koji ima:

- konstruktor koji u argumentu prima naziv i setuje ga, a pored toga setuje i tip na **P**

Interfejs **Proizvodi** koji ima:

- metodu **dajListuProizvoda** koja vraća niz objekata tipa **Proizvod**
- metodu **cena** koja prima string naziv i vraća float cenu
- metodu **cena** koja prima objekat tipa **proizvod** i vraća float cenu

Tip **Proizvođač** implementira interfejs proizvodi i koji ima:

- niz **N** objekata Proizvod
- niz float cene, čiji prvi element ima vrednost cene prvog proizvoda iz prethodnog niza, drugi drugog itd.
- string **ime**
- konstruktor koji prima ime
- metodu **dajListuProizvoda** koja vraća niz proizvode
- apstraktan metodu **dodajProizvod** koja prima objekat **Proizvod** i cenu
- metodu **nađiProizvod** koja prima string sa nazivom proizvoda i traži proizvod u nizu **N** sa takvim imenom, pa ako nađe vraća indeks nađenog objekta, inače vraća -1
- metodu **nađiProizvod** koja prima objekat Proizvod i traži ga u nizu **N**, pa ako nađe vraća indeks nađenog objekta, inače vraća -1
- metodu **cena** koja prima string sa nazivom proizvoda i vraća cenu tog proizvoda ako postoji u nizu **N** inače vraća -1
- metod **cena** koji prima objekat proizvod i vraća njegovu cenu ako postoji u nizu **N** inače vraća -1
- string reprezentaciju koja sadrži ime i opise svih proizvoda iz niza **N**

Klasa **Firma** koja predstavlja proizvođača i ima:

- string podatak **PIB**
- konstruktor koji prima ime i PIB i setuje ih
- metodu **dodajProizvod** koja dobije objekat Proizvod i cenu, ako nađe proizvod u nizu proizvode firme menja cenu, a ako ga ne nađe dodaje ga i postavlja cenu.

CENA se postavlja tako što se dobijena vrednost poveća za 40%.

Klasa **Poljoprivrednik** koja predstavlja proizvođača i ima:

- konstruktor koji prima ime setuje ga
- metodu **dodajProizvod** koja dobija objekat Proizvod i cenu. Na početku proverava da li je proizvod tipa P, pa ako nije diže izuzetak Greska sa odgovarajućim komentarom. Ako je proizvod odgovarajućeg tipa, on proverava da li proizvod postoji u nizu njegovih proizvoda, ako ga pronade, menja cenu, u suprotnom ga dodaje i postavlja cenu.

CENA se postavlja tako što se dobijena vrednost poveća za 20%.

Testna klasa treba da:

- kreira dva poljoprivredna (grašak i repa) i jedan industrijski proizvod (šećer)
- kreira jednog poljoprivrednika Marka Markovića i jednu firmu Poljolink
- Marku dodaje na spisak proizvode grašak i repu i setuje im cene na 50 i 100, a zatim ponovo dodaje grašak sa cenom 100
- firmi Poljolink dodaje šećer i repu po cenama 400 i 100
- formira niz Proizvođača i smešta u njega Marka i Poljolink
- ispisuje string reprezentaciju svih elemenata niza

### Zadatak 3.

U programskom jeziku Java implementirati:

Apstraktan tip **Vrednosno** koji sadrži:

- apstraktanu metodu **vrednost** koja vraća float promenljivu

Konkretna tip **JedinicaMere** koji:

- ima privatnu konstantnu **oznaka** tipa String koja se setuje pri instanciranju objekta tipa **JedinicaMere** i može imati jednu od četiri vrednosti "kom", "l", "m" i "kg", a sam konstruktor diže grešku ako se pokuša oznaci dodeliti neka vrednost koja nije jedna od 4 navedene
- string reprezentaciju koja sadrži oznaku

Apstraktan tip **Artikal** koji:

- ima privatni **naziv** tipa String i privatnu konstantnu jedinicu mere **jm**
- konstruktor koji prima string koji predstavlja naziv artikla i jedinicu mere i setuje ih
- string reprezentaciju koja sadrži naziv artikla
- **getter** i **setter** za jedinicu mere.

Konkretna tipove **Mleko** i **Šećer** koji predstavljaju Artikle koji se mere redom u litrima i kilogramima (vrednost atributa naziv sadrži ime proizvođača i naziv proizvoda, npr. u formi "Imlek: mleko"). Dakle, Mleko i Šećer u svojim definicijama jedino sadrže konstruktore koji primaju naziv i setuju ga.

Konkretna tip vrednosni **Zapis** koji ima:

- privatne podatke **artikal** tipa Artikal, **količina** tipa float i **jedCena** tipa float koja predstavlja cenu artikla po jedinici mere
- konstruktor koji prima vrednosti za sva tri atributa i setuje ih
- **getter-e** i **setter-e** za sve attribute
- vrednost artikla predstavlja ukupnu cenu, odnosno računa se kao proizvod količine i cene po jedinici mere
- string reprezentacija sadrži naziv artikla, količinu, jedinicu mere i vrednost artikla

Konkretni tip **Korpa** koja sadrži:

- privatni niz zapisa o artiklima u korpi **sadržaj**, privatni podatak **max** tipa float koji predstavlja maksimalnu vrednost potrošačke korpe
- konstruktor koji dobija maksimalnu vrednost korpe
- metodu **dodaj** koja prima zapis o artiklu i smešta ga u **sadržaj**, ako bi vrednost korpe prešla dozvoljenu baca grešku
- vrednost korpe predstavlja sumu vrednosti svih artikala u njoj
- string reprezentacija sadrži zapise o svim artiklima u njoj i to u formi [(naziv artikla, količina, jedinica mere i vrednost artikla),[(naziv artikla, količina, jedinica mere i vrednost artikla),...]

U testnoj klasi treba:

- napraviti niz vrednosnih objekata **red**
- napraviti 2 korpe, u prvu ubaciti dva pakovanja šećera, jedno od jednog kilograma, drugo od pola, u drugu ubaciti dva pakovanja mleka od po 1 litra
- u **red** ubaciti 2 korpe i jedno mleko od 0.7 l
- odštampati ukupnu vrednost svih artikala u tom redu

#### Zadatak 4.

Napisati program koji predstavlja Engine za simulaciju igre „Borba skakača”, koja se može opisati na sledeći način:

- Igra je uprošćena verzija šaha.
- U njoj učestvuju dva igrača.
- Na početku igre svaki igrač sa svoje strane u prvoj liniji dobija slučajno raspoređena tri skakača (čije je kretanje definisano na isti način kao kretanje skakača u šahu) i po jednog kralja (čije je kretanje potpuno isto kao i kretanje kralja u šahu).
- Cilj igre je „pojesti” protivničkog kralja. Pri povlačenju poteza program vodi računa o:
  - tome koji je tekući igrač, tako što uvek počinje beli i igrači naizmenično povlače po jedan potez,
  - tome da li je to figura koja pripada tekućem igraču,
  - tome da li ta vrsta figure prema sopstvenim pravilima kretanja može da pređe na novo polje,
  - tome da li je ciljano polje zauzeto figurom iste boje, u kom slučaju ne dozvoljava potez,
  - tome da li je ciljano polje zauzeto protivnikovom figurom, pa ako jeste istu uklanja i proverava da li je ostvaren uslov za kraj partije.

U programskom jeziku Java implementirati:

Apstraktna klasa **Figure** koja sadrži:

- string promenljivu **color** koja sadrži naziv boje figure (crna ili bela)
- javni konstruktor koji prima naziv boje i setuje ga
- javni **getter** za promenljivu **color**
- javnu apstraktnu metodu **move** koja prima prvu i drugu koordinatu polazne pozicije i prvu i drugu koordinatu ciljane pozicije (sve integer tipa) i vraća boolean.

Konkretni tipove **King** i **Knight** koji predstavljaju figure (i to kralja i skakača) i imaju:

- javne konstruktore koji primaju naziv i boju figure i setuju ih
- javnu metodu **toString** koja vraća string reprezentaciju figure u formi K[boja], odnosno S[boja]
- javnu metodu **move** koja na osnovu podataka o polaznoj i ciljnoj poziciji na tabli vraća true ili false u zavisnosti od toga da li je kralju, odnosno skakaču dozvoljeno takvo kretanje po tabli.

Konkretan tip **KnightFightEngine** koji predstavlja engine za igru i sadrži:

- privatni celobrojni podatak **moves** koji broji koliko poteza je od početka igre napravljeno, pri čemu se potez broji tek kada je povučen
- privatni string podatak **next** koji nosi informaciju o tome koja boja je na potezu
- privatni podatak **board** koji predstavlja matricu dimenzija 8X8 u kojoj je zabeležen sadržaj svakog polja na tabli, pri čemu sadržaj polja može biti neka od figura ili null
- javni **getter** za broj poteza i boju figura igrača na potezu
- javnu metodu **initialize** koja nema povratnu vrednost i poziva se na početku svake partije; pri pozivu ove metode kreiraju se po tri figure skakača i jedna figura kralja za svaku boju I slučajnim izborom se raspoređuju u prvoj, odnosno osmoj vrsti na tabli.
- javnu metodu **deployFigure** koja dobija instance **Figure** i indeks vrste u koju figura mora biti raspoređena, a zatim slučajnim izborom na slobodno polje u odgovarajućoj vrsti postavlja figure tako što je upisuje na odgovarajuću poziciju matrice **board**.
- javnu metodu **printBoard** koja na standardni izlaz ispisuje sadržaj table u sledećem obliku:

	A	B	C	D	E	F	G	H
0	K[Bela]	+	+	S[Bela]	S[Bela]	+	S[Bela]	+
1	+	+	+	+	+	+	+	+
2	+	+	+	+	+	+	+	+
3	+	+	+	+	+	+	+	+
4	+	+	+	+	+	+	+	+
5	+	+	+	+	+	+	+	+
6	+	+	+	+	+	+	+	+
7	+	S[Crna]	K[Crna]	+	+	S[Crna]	S[Crna]	+

- javnu metodu **move** koja dobija koordinate polazne i ciljne pozicije u potezu; prve koordinate pozicije se prosleđuju kao slova (A-H), a druge kao celi brojevi. Kada metoda dobija informacije o potezu, ona:
  - prevodi slovne koordinate u numeričke,
  - proverava da li na polaznoj poziciji postoji figura, koje boje i da li je ta boja na potezu
  - ako su prethodno navedeni uslovi zadovoljeni, onda proverava da li data figura može biti pomerena na ciljnu poziciju
  - ako figura prema sopstvenim pravilima može preći na ciljnu poziciju, proverava se da li je ciljna pozicija zauzeta
  - ako je ciljna pozicija zauzeta figurom iste boje, tekuća figura se ne pomera
  - ako je ciljna pozicija zauzeta figurom različite boje, onda se figura na ciljnoj pozicije izbacuje, a tekuća pomera na ciljnu poziciju; u tom slučaju se broj poteza povećava za jedan
  - ako na ciljnoj poziciji nema figura, onda se tekuća jednostavno pomera na ciljnu poziciju i broj poteza se uvećava za jedan.
- javnu metodu **isEnd** koja proverava da li na tabli nedostaje kralj neke boje i vraća true ili false
- javnu metodu **getWinner** koja vraća boju figura igrača koji je pobedio
- javnu metodu **getFigures** koja prima naziv boje i vraća niz figura odgovarajuće boje.
- javnu metodu **getPosition** koja prima figuru a vraća dvočlani niz celih brojeva koji predstavljaju koordinate date figure.

Testna klasa treba da:

- kreira jednu instacu igre i inicijalizuje je
- štampa sadržaj table
- smešta reference na figure belog igrača u jedan niz, a reference na figure crnog igrača u drugi niz.
- kupi pozicije belog kralja i pomera ga za dva polja napred.
- štampa sadržaj table.
- kupi pozicije crnog kralja i pomera ga za dva polja napred.
- štampa sadržaj table.
- poziva metodu **isEnd** i štampa odgovarajući komentar na standardnom izlazu.

## Zadatak 5.

U programskom jeziku Java implementirati:

Apstraktan tip **RačunarNaMreži** koji sadrži:

- privatnu promenljivu **adresa** tipa String
- javni konstruktor koji prihvata adresu računara i setuje je
- javni **getter** za atribut **adresa**
- metodu **toString** koja vraća vrednost atributa **adresa**
- javnu metodu **equals** koja prihvata objekat tipa String i poredi ga sa vrednošću atributa **adresa**
- javnu metodu **equals** koja prihvata objekat tipa **RačunarNaMreži** i poredi njegovu adresu sa atributom **adresa**

Interfejs **Klijent** sadrži sledeće funkcionalnosti:

- metodu **prijaviSe** koja prihvata objekat tipa String (predstavlja adresu servera) i nema povratnu vrednost
- metodu **odjaviSe** koja prihvata objekat tipa String (predstavlja adresu servera) i nema povratnu vrednost
- metodu **pošaljiZahtev** koja prihvata objekat tipa String (predstavlja adresu servera) i nema povratnu vrednost
- metodu **prihvatiOdgovor** koja prihvata objekat tipa String (predstavlja adresu servera) i nema povratnu vrednost
- metodu **getAdresa** koja ne prihvata argumente i vraća objekat tipa String

Tip **VebKlijent** koji predstavlja računar na mreži koji je **Klijent** i sadrži:

- privatni atribut **imeBrauzera** tipa String
- javni konstruktor koji prihvata adresu i imeBrauzera i setuje ih
- javnu metodu **prijaviSe** koja prihvata adresu servera (tipa String) i mreži prosleđuje zahtev sa porukom “**prijava**”
- javnu metodu **odjaviSe** koja prihvata adresu servera (tipa String) i mreži prosleđuje zahtev sa porukom “**odjava**”
- javnu metodu **pošaljiZahtev** koja prihvata adresu servera (tipa String) i mreži prosleđuje zahtev sa porukom “**ping**”
- javnu metodu **prihvatiOdgovor** koja prihvata poruku tipa String i štampa je u format “**stringReprezentacija:poruka**”
- prepisanu metodu **toString** koja vraća string reprezentaciju oblika “**adresa [imeBrauzera]**”

Tip **Server** koji predstavlja računar na mreži i sadrži:

- protected promenljivu **prijavljeniKlijenti** koja predstavlja listu objekata tipa **Klijent**
- protected promenljivu **brPrijavljenihKlijenata** tipa int koji predstavlja broj prijavljenih klijenata
- javni konstruktor koji prihvata adresu i setuje je
- javnu metodu **prihvatiZahtev** koja prihvata objekat tipa **Klijent** i objekat tipa string koji predstavlja poruku koju šalje klijent. U slučaju da je poruka:
  - **prijava** - klijent se dodaje u niz prijavljenih klijenata
  - **odjava** - klijent se uklanja iz niza prijavljenih klijenata
  - **ping** - ukoliko je klijent prijavljen, vrši obradu zahteva
- protected metodu **obradi** koja prihvata objekat tipa **Klijent** koji predstavlja klijenta čiji se zahtev obrađuje i preko mreže šalje odgovor na odgovarajuću adresu oblika „Server **adresa** šalje poruku.”

Tip **MultikastServer** koji je specijalna vrsta **Servera** koji sadrži:

- javni konstruktor koji prihvata adresu i setuje je
- protected metodu **obradi** koja prihvata objekat tipa **Klijent** koji predstavlja klijenta čiji se zahtev obrađuje i preko mreže na adresu svih prijavljenih klijenata šalje odgovor oblika „Server **adresa** svima šalje poruku, jer se javio **podaciKlijenta**.”

Tip **Mreza** koji sadrži:

- privatni statički atribut **dostupniUređaji** koji predstavlja listu objekata tipa **RačunarNaMreži**
- privatni statički atribut **brUređaja** tipa int koji predstavlja broj prijavljenih uređaja
- javnu statičku metodu **registrujUređaj** koji prihvata objekat tipa **RačunarNaMreži** i dodaje ga u niz dostupnih uređaja i nema povratnu vrednost

- privatnu statičku metodu **pronadi** koja prihvata argument tipa String, pronalazi **RačunarNaMreži** u nizu dostupnih uređaja sa datom adresom i vraća ga kao povratnu vrednost
- javnu statičku metodu **proslediZahtev** koja prihvata string adresu na koju se šalje zahtev, objekat tipa **Klijent** koji šalje zahtev i poruku tipa String, pronalazi server sa odgovarajućom adresom i ukoliko postoji prosleđuje zahtev tom serveru
- javnu statičku metodu **proslediOdgovor** koji prihvata string adresu klijenta kojem se šalje odgovor i string poruku koja se šalje, pronalazi klijenta sa odgovarajućom adresom i ukoliko postoji prosleđuje mu odgovor

Testna klasa treba da:

- kreira tri servera, dva obična (uns,ftn) i jedan multikast (twitter), dva veb klijenta (Ivica i Marica) i sve ih smešta u niz i registruje ih na mreži
- Ivicu prijavljuje na uns i twitter, a maricu na twitter
- Ivica salje zahteve sledećim redosledom:
  - uns
  - twitter
  - ftn
  - twitter
- Ivica i Marica se odjavljuju

## Zadatak 6.

U programskom jeziku Java implementirati:

Interfejs **jelzvođač** sadrži sledeće funkcionalnosti:

- metodu **prijavaFestival** koja prima jedan objekat klase **Festival** i nema povratnu vrednost
- metodu **odjavaFestival** koja prima jedan objekat klase **Festival** i nema povratnu vrednost
- metodu **nastup** koja prima jedan objekat klase **Festival** i vraća objekat tipa String

Konkretni tip **Student** ima:

- privatni String podatak **ime**
- privatni String podatak **prezime**
- javni konstruktor koji prima ime i prezime studenta i setuje ih
- javni **getter** za ime
- javni **getter** za prezime
- string reprezentaciju oblika ime prezime

Konkretni tip **StudentMuzičar**. **StudentMuzičar** je student koji je i izvođač. On ima:

- privatni String podatak vrstaMuzike
- javni konstruktor koji prima ime, prezime i vrstu muzike i setuje ih
- javni **getter** za vrstu muzike
- javnu metodu **prijavaFestival** koja prima referencu na objekat klase **Festival** kome se šalje poruka o prijavi sa informacijom o studentu muzičaru
- javnu metodu **odjavaFestival** koja prima referencu na objekat klase **Festival** kome se šalje poruka o odjavi sa informacijom o studentu muzičaru
- javnu metodu **nastup** koja prima referencu na objekat klase **Festival** a vraća string koji objekat festivala vraća na poziv njegove metode nastup, kojoj se prosleđuje informacija o student muzičaru za čiji nastup se traže podaci.

Konkretnu klasu **Bend** koja implementira interfejs **jelzvođač** i koja ima:

- privatni String podatak **naziv**
- javni konstruktor koji prima naziv i setuje ga
- javni **getter** za naziv

- javnu metodu **prijavaFestival** koja prima referencu na objekat klase **Festival** kome se šalje poruka o prijavi sa informacijom o bendu
- javnu metodu **odjavaFestival** koja prima referencu na objekat klase **Festival** kome se šalje poruka o odjavi sa informacijom o bendu
- javnu metodu **nastup** koja prima referencu na objekat klase **Festival** a vraća string koji objekat festivala vraća na poziv njegovog metoda **nastup**, kome se prosleđuje informacija o bendu za čiji nastup se traže podaci.

Konkretnu klasu **Par** koja ima:

- privatni **Object** podatak **key**
- privatni **Object** podatak **value**
- javni konstruktor koji prima **key** i **value** i setuje ih
- javni **getter** i **setter** za **value**
- javni **getter** za **key**
- javnu metodu **equals** koja prima objekat i poredi ga sa **key**, ako se ova dva podatka poklapaju vraća **true**, inače vraća **false**.

NAPOMENA. Izvođačem se smatra svaki objekat koji pripada tipu **jelzvođač**.

Konkretnu klasu **Festival** koja ima:

- privatni **String** podatak **naziv**
- privatni niz **raspored** koji pamti reference na objekte tipa **Par**
- privatni podatak **početak** tipa **Date** (biće pojašnjeno)
- privatni **int** **trajanje** koji predstavlja broj dana trajanja festivala
- privatni **int** **brBina** koji predstavlja broj bina
- privatni **int** **brUčesnika** koji predstavlja broj prijavljenih učesnika
- javni konstruktor koji prima naziv, trajanje, početak i broj bina i setuje ih
- javni konstruktor koji prima naziv, trajanje i početak i i setuje ih, a broj bina postavlja na 1
- metodu **prijava** koja prima objekat izvođača, određuje dan i binu za nastup izvođača, ukoliko je dan nakon isteka festivala štampa poruku da nema slobodnih termina, inače kreira instancu klase **Par** gde je **key** izvođač, a **value** **String** oblika "Dan " + danPoRedu + ", " + "bina " + bina I dodaje taj par u raspored
- metodu **odjava** koja prima objekat tipa **jelzvođač**, izbacuje odgovarajući par iz rasporeda i ažurira **value** za preostale parove iz rasporeda
- metodu **nastup** koja prima objekat tipa **jelzvođač**, pronalazi odgovarajući par u rasporedu i vraća opis nastupa datog izvođača kao **String** oblika naziv + "(" + početak + ")" - " + string reprezentacija izvođača + ": " + vrednost podatka o nastupu
- javnu metodu **ispišiRaspored** koja u jednoj liniji ispisuje **String** "Raspored", a za svakog izvođača prijavljenog na festival ispisuje vrednost koju vraća f-ja **nastup**

Testna klasa treba da:

- kreira kalendar sa trenutnim datumom 03.07.2012. kodom: **Calendar c = new GregorianCalendar(2012, 07, 03);**
- kreira festival "Arsenal fest" koji traje sedam dana, a početni datum tipa **Date** dobija iz kalendara kodom: **c.getTime()**
- kreira bendove Gobline i Električni orgazam
- kreira studenta muzičara Ivana Ivanović koja izvodi pop muziku
- ova tri izvođača pakuje u niz
- sve izvođače iz niza prijavljuje na festival
- ispisuje raspored festivala
- odjavljuje Gobline sa festivala
- ispisuje raspored festivala