A Zadatak

1. Uvod

Cilj vježbe je kreirati hijerarhiju klasa za rad sa bankovnim računima

2. Opis vježbe

Od studenta se zahtjeva da dizajnira i implementira klase za rad sa bankovnim računima.
Svaki bankovni račun treba imati slijedeća svojstva (stanja): broj računa, vlasnika, početno stanje, trenutno stanje, lista uplata i isplata sa računa.
Jednom postavljeni broj računa se ne može kasnije mijenjati, dok se početno stanje postavlja u 0 prilikom kreiranja računa.
Trenutno stanje se izračunava na temelju početnog stanja i liste uplata i isplata.
Od ponašanja treba se implementirati operacije uplata, isplata. Metode koje opisuju uplatu i isplatu trebaju vraćati boolean vrijednost koja definira da li je operacija obavljena. (npr. Ako nije moguće napraviti isplatu treba se vratiti vrijednost false). Obje metode primaju promet za koji se želi napraviti povećanje (uplata) ili smanjenje (isplata) stanja bankovnog računa.
Promet po računu sadržava iznos (posebno za uplatu i posebno za isplatu i samo jedna vrijednost može biti veća od 0) i valutu.
Vlasnik od podataka ima oib, ime, prezime i adresa. Oib se nakon postavljanja ne smije mijenjati.

3. Zadatak za studenta

Kreirati vlasnika v1 (oib=1, ime=Ivo, prezime=Ivić, adresa=Maksimir 22) i v2 (oib=2, ime=Mate, prezime=Matić, adresa=Maksimir 11).
Nakon svakog slijedećeg koraka potrebno je ispisati stanje pojedinog računa (korištenjem toString metode)

- Za svakog vlasnika definirati po jedan bankovni račun b1 (broj računa=1) i b2 (broj računa=2).
- Napraviti uplatu od 100 HRK na račun b1 i isplatu od 90 HRK sa računa b2
- Kreirati referencu b3 i pridodjeliti vrijednosti b1 (BankovniRacun b3=b1)
- Napraviti isplatu od 100 HRK sa računa b3 i uplatu od 200 HRK na račun b2
- Napraviti novi račun (vlasnik v1, broj računa 3) i pridodjeliti varijabli b3
- Napraviti uplatu od 250 HRK na račun b3

4. Dodatne upute

Potrebno je u Intellij okruženju napraviti novi Java project File->New->Project. Nakon toga potrebno je kreirati klase za bankovni račun, tekući račun i oročenje. Nove klase se kreiraju putem opcije File->New->Java Class.
Kao prvi iskaz u klasi je deklaracija paketa (package) u kojem će se nalaziti nova klasa. Naziv paketa se može sastojati od više dijelova odvojenih točkom i svaki dio naziva počinje sa malim slovom (npr. hr.vsite.java).
Kod pisanja implementacije klasa treba voditi računa od pravima pristupa svojstvima (varijablama) i ponašanju (metodama). Prava pristupa se definiraju modifikatora pristupa: private, protected, public i *bez deklaracije* (package private):

Modifikator	klasa	podklasa	package	svijet
private	X			
protected	X	X	X	
public	X	X	X	X
Bez modifikatora	X		X	

Modifikatori pristupa se navode prije deklaracije varijable ili metode:
private int iznos;
private boolean uplata(int iznos){ ...}

Za potrebe testiranja potrebno je u svakoj klasi implementirati i toString metodu:
public String toString()
Nakon što je implementirana klasa u Intellij se može pokrenuti generiranje toString metode pod Code->Generate pa izabrati

```
public String toString() {
        return "Promet [uplata=" + uplata + ", isplata=" + isplata
                        + ", valuta=" + valuta + "]";
}
```

Ako je implementirana toString onda se stanje objekta može jednostavno ispisati npr:

```
Promet p = new Promet(0, 100, "HRK");
System.out.println(p);
```

Ispis:

```
Promet [uplata=0, isplata=100, valuta=HRK]
```

B zadatak

1. Uvod

Cilj vježbe je proširiti hijerarhiju klasa za rad sa bankovnim računima

2. Opis vježbe

Od studenta se zahtjeva da se nadopuni hijerarhija klasa koja je implementirana u 3. vježbi. Treba dodati i implementirati za dva specijalna slučaja bankovnog računa: tekući račun i oročenje.

Tekući račun je specijalizirani bankovni račun koji ima dva nova svojstva: dozvoljeni minus i zatezna kamata (zatezna kamata je kamata koja se obračunava za negativni saldo). Kod tekućeg računa može se napraviti isplata do vrijednosti definirane svojstvom dozvoljeni minus.

Oročenje je posebna vrsta bankovnog računa koja se razlikuje od bankovnog računa po tome što nije dozvoljeno raditi isplate.

Oba specijalna slučaja imaju i dodatno svojstvo a to je kamata za pozitivno stanje računa (npr. vrijednost 15 za 15%) a od ponašanja obračun kamata. Metoda za obračun kamata

treba povećati iznos stanja bankovnog računa (ako je stanje pozitivno) i vratiti vrijednost obračunate kamate:
Stanje = stanje*(1+kamata/100);
Kod tekućeg računa obračun kamate se radi i za pozitivnu vrijednost (kamata) i za negativnu vrijednost (zatezna kamata koja smanjuje stanje računa).

Kod vlasnika računa moramo razlikovati da li se radi o fizičkoj osobi ili pravnoj osobi. Vlasnik objekt opisan u 3. vježbi je fizička osoba tako da se treba definirati klasu za pravnu osobu. Pravna osoba ima slijedeća svojstva OIB, naziv, adresu i vlasnika.

3. Zadatak za studenta

- U sve metode za provođenje transakcija i obračun kamata dodati ispis o kojoj se klasi i metodi radi
- Kreirati dva vlasnika fizičku osobu i pravnu osobu
- Definirati dvije varijable bankovni račun b1 i b2
- Kreirati novi tekući račun (dozvoljeni minus 100, kamata 4, zatezna kamata 14) sa vlasnikom fizička osoba i dodijeliti varijabli b1
- Kreirati novi oročeni račun (kamata 5) sa vlasnikom pravna osoba i dodijeliti varijabli b2
- Ispisati stanja računa b1 i b2 (sa nazivima vlasnika)
- Napraviti uplatu na račun b1 u iznosu 100 HRK
- Napraviti uplatu na račun b2 u iznosu 200 HRK
- Ispisati stanja računa b1 i b2 (sa nazivima vlasnika)
- Napraviti obračun kamata na računima b1 i b2
- Ispisati stanja računa b1 i b2 (sa nazivima vlasnika)
- Napraviti isplatu sa računa b1 u iznosu 200 HRK
- Napraviti isplatu sa računa b2 u iznosu 10 HRK
- Ispisati stanja računa b1 i b2 (sa nazivima vlasnika)
- Napraviti obračun kamata na računima b1 i b2
- Ispisati stanja računa b1 i b2 (sa nazivima vlasnika)

4. Dodatne upute

Nasljeđivanje je jedno od osnovnih svojstva objektno orijentiranog programiranja. Kod nasljeđivanja poklasa će imati sva svojstva i ponašanja koja su definirana u nadklasi (osim onih koja su definirana kao private):

```
public class Vozilo {
	protected int brzina;
	public int brojKotaca = 4;
	private int brojOkretaja;
	//definicija konstruktora koja inicijalizira
	//objekt prilikom kreiranja (postavlja se trenutna brzina)
	public Vozilo(int a){
		brzina=a;
	}
```

```
	public void promijeniBrzinu(int novaBrzina){
		brzina = novaBrzina;
	}

	public int povecajBrojOkretaja(){
		brojOkretaja = brojOkretaja+100;
		return brojOkretaja;
	}
}
public class TeretnoVozilo extends Vozilo {
	@Override
	public void promijeniBrzinu(int novaBrzina) {
		if (Math.abs(novaBrzina - getBrzina()) > 2)
			return;
		brzina = novaBrzina;
	}
}
```

Klasa TerentoVozilo nasljeđuje svojstva i ponašanje od klase Vozilo (definirano sa `extends Vozilo`) tako da i TeretnoVozilo ima svojstava brzina i brojKotaca (zbog private nema svojstvo brojKotaca) te ponašanja promijeniBrzinu i povecajBrojOkretaja. Izvedene klase mogu promijeniti implementaciju naslijeđenih ponašanja bazne klase. To se ostvaruje tako da se u izvedenoj klasi definira i implementira metoda koja ima isti naziv. Tako u primjeru TeretnoVozilo implementira svoje specifično ponašanje promjeniBrzinu.

Kod kreiranja novog računa (tekućeg računa ili oročenja) trebamo kreirati novi objekt odgovarajuće klase. Pošto i jedna i druga klasa nasljeđuju klasu BankovniRacun to nam omogućuje da novo kreirane objekte pridružimo istoj varijabli tipa BankovniRacun. Općenito se objekt izvedene klase može pridružiti varijabli tipa bazne klase:
BankovniRacun br = new TekuciRacun(...);
Jedini je problem što se mogu pozvati samo one metode na objektu koje su definirane u klasi tipa varijable. U konkretnom slučaju nad objektom tekući račun mogu se pozvati samo metode koje su definirane u klasi BankovniRacun.
Svojstvo objektno orijentiranih jezika koje se zove polimorfizam nam omogućuje da se pozove prava implementacija metode u ovisnosti o tipu objekta a ne o tipu varijable.

```
public class Prva {
	public void pozivMetode(){
		System.out.println("Ispis iz klase Prva");
	}
}
public class Druga extends Prva {
	public void pozivMetode(){
		System.out.println("Ispis iz klase Druga");
	}
}
public class TestPolimorfizam {
	public static void main(String[] args) {
		Prva a = new Prva();
		Druga b = new Druga();
		Prva c = a;
		c.pozivMetode();
		c = b;
		c.pozivMetode();
	}
}
Ispis:
Ispis iz klase Prva
Ispis iz klase Druga
```