3. laboratorijska vježba (11 bodova)

Važna napomena: *u svim zadacima potrebno je napisati Javadoc komentare za svaki razred te generirati dokumentaciju. Svi nazivi razreda, metoda i varijabli moraju biti na engleskom. Sav napisani programski kod mora biti napisan u skladu s konvencijama imenovanja varijabli, metoda i razreda (varijable i metode: malo početno slovo, camel-case; razredi i sučelja: veliko početno slovo, camel-case; konstante: uobičajeno sve veliko i razdvajanje podvlakom) te ostalim pozitivnim praksama (uključivo i korektno uvlačenje redaka; smisleno razdvajanje više različitih semantički grupiranih redaka praznim retcima, pravilnim razmještajem otvorene i zatvorene vitičaste zagrade i slično). Za više informacija pogledajte* ***http://www.oracle.com/technetwork/java/codeconventions-150003.pdf***

Zadatak 1.

Na stranici predmeta dostupna je biblioteka `picture.jar`. Biblioteka je opremljena i pripadnom dokumentacijom u obliku javadoc-a. Biblioteka sadrži implementaciju razreda `hr.fer.oop.lab3.pic.Picture` koji predstavlja crno-bijelu sliku te razreda `hr.fer.oop.lab3.pic.PictureDisplay` koji nudi mogućnost prikaza takve slike na standardnom izlazu u ASCII grafici ili pak na ekranu u zasebnom prozoru. Proučite pripadnu dokumentaciju.

Napišite razrede `EquilateralTriangle`, `Circle` i `Rectangle` koji modeliraju jednakostranični trokut, krug i pravokutnik. Pretpostavite da jednakostranični trokut ima bazu koja je usporedna s osi *x* te visinu koja raste u smjeru negativne osi *y* (prema gore na ekranu). Sve razrede smjestite u paket `hr.fer.oop.lab3.prob1`.
Opremite napisane razrede adekvatnim konstruktorima. Omogućite korisniku da svaki od likova stvara predavanjem svih potrebnih parametara kroz konstruktor ili pak predavanjem reference na neki drugi takav lik, u kojem slučaju treba preuzeti parametre iz tog lika (naravno, `Circle`-u se može predati samo drugi `Circle` i slično za ostale).

Svaki od razreda opremite metodom `drawOnPicture` koja prima referencu na sliku i na njoj paljenjem slikovnih elemenata crta taj lik (prikazuje se čitava ispuna lika). Iscrtavanje implementirajte na način da slijedno uzimate točku po točku pozadine. Ukoliko je ta točka unutar geometrijskog lika, slikovni element je potrebno upaliti, a inače ne. U slučaju da dio lika izlazi izvan područja slike, potrebno je nacrtati samo dio koji je vidljiv (ne smiju se događati pogreške odnosno iznimke).

Napišite glavni program `Demonstration` (u istom paketu) koji stvara jednu sliku dimenzija 100x50, stvara po dva primjerka svakog od zadanih razreda (likova), crta ih na slici te sliku na kraju prikazuje na standardnom izlazu (`System.out`). Poslužite se prikladnim razredima iz biblioteke `picture.jar`. Da biste to mogli napraviti, uključite `picture.jar` u projekt.
Na vježbi vas možemo pitati da izvorni kod prevedete i pokrenete direktno iz naredbenog retka.

Zadatak 2.

Napišite implementaciju parametriziranog razreda `SimpleHashtable<K,V>`. Razred predstavlja tablicu raspršenog adresiranja koja omogućava pohranu uređenih parova (ključ, vrijednost). Parametar `K` predstavlja tip ključa a parametar `V` tip vrijednosti. Postoje dva javna konstruktora: podrazumijevani (default) koji stvara tablicu veličine 16 slotova, te konstruktor koji prima jedan argument: broj koji predstavlja željeni početni kapacitet tablice i koji stvara tablicu veličine koja je potencija broja 2 koja je prva veća ili jednaka predanom broju (npr. ako zada 30, bira se 32).

Jedan slot tablice modelirajte statičkim ugniježđenim razredom `TableEntry<K,V>`. Ako još ne razumijete razliku između unutarnjih razreda i statičkih ugniježđenih razreda (nestatički vs. statički razred), razriješite sve nedoumice – pitat ćemo vas to. Primjerci razreda `TableEntry` imaju člansku varijablu `key` u kojoj pamte predani ključ, člansku varijablu `value` u kojoj pamte pridruženu vrijednost te člansku varijablu `next` koja pokazuje na sljedeći primjerak razreda `TableEntry` koji se nalazi u istom slotu tablice (izgradnjom ovakve liste rješavat ćete problem preljeva – situacije kada u isti slot treba upisati više uređenih parova). Svojstvo `key` u razredu `TableEntry` ne smije se moći mijenjati (mora biti *read-only*).

Ideju uporabe ovakve kolekcije ilustrira sljedeći kod.

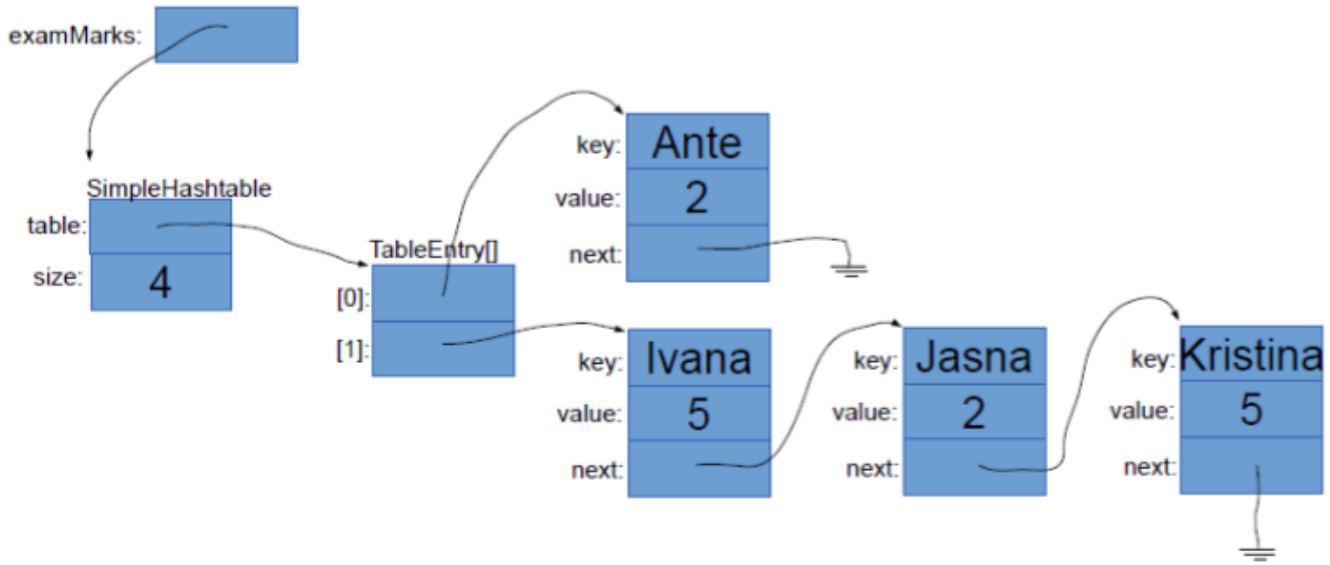
```
// create collection:
SimpleHashtable<String,Integer> examMarks = new SimpleHashtable<>(2);

// fill data:
examMarks.put("Ivana", Integer.valueOf(2));
examMarks.put("Ante", Integer.valueOf(2));
examMarks.put("Jasna", Integer.valueOf(2));
examMarks.put("Kristina", Integer.valueOf(5));
examMarks.put("Ivana", Integer.valueOf(5)); // overwrites old grade for Ivana

// query collection:
Integer kristinaGrade =  examMarks.get("Kristina");
System.out.println("Kristina's exam grade is: " + kristinaGrade); // writes: 5

// What is collection's size? Must be four!
System.out.println("Number of stored pairs: " + examMarks.size()); // writes: 4
```

Za potrebe izračuna slota u koji treba ubaciti uređeni par koristite metodu `hashCode()` ključa, pa modulo veličina tablice. Ključ uređenog para ne smije biti `null` dok vrijednost može biti `null`.

Razred `SimpleHashtable` treba imati sljedeće članske varijable:

- `TableEntry<K,V>[] table`: polje slotova tablice,
- `int size`: broj parova koji su pohranjeni u tablici.

Pojednostavljeni prikaz stanja u memoriji nakon izvođenja koda iz prethodnog primjera ilustriran je na sljedećoj slici.

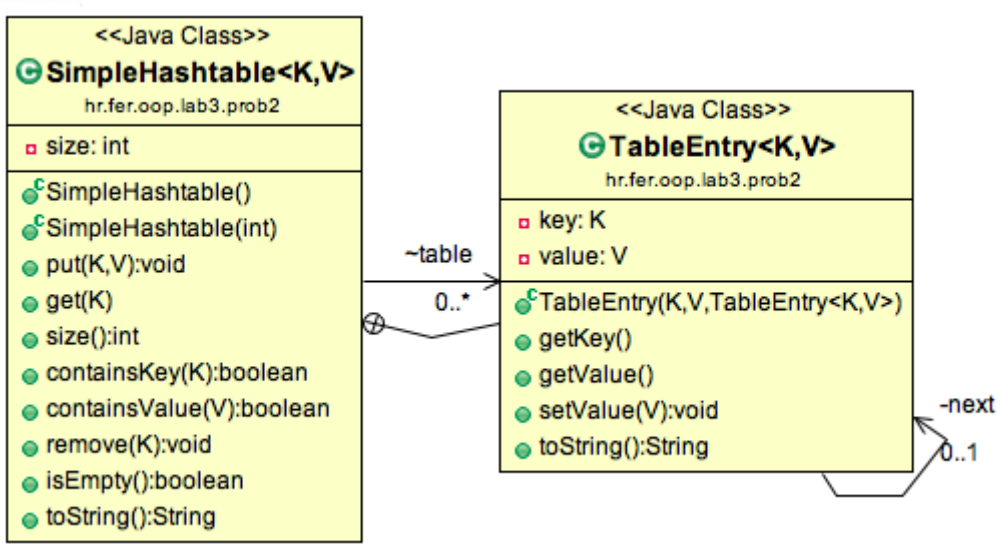


Pri tome na platformi Java 8 vrijedi:

objekt	**hashCode()**	**|hashCode()|**	**slot=|hashCode()| % 2**
"Ivana"	71029095	71029095	1
"Ante"	2045822	2045822	0
"Jasna"	71344303	71344303	1
"Kristina"	-1221180583	1221180583	1

Stoga će ključevi Ivana, Jasna i Kristina biti u slotu 1, a ključ Ante u slotu 0. Razmislite odgovara li prikazana slika stvarnom stanju u memoriji ili bismo za stvarno stanje dio slike trebali drugačije nacrtati?

Dijagram razreda koji prikazuje razrede ovog zadatka prikazan je u nastavku.



Metode i konstruktori koje razred `SimpleHashtable` mora ponuditi navedeni su u nastavku bez posebne dokumentacije (iz imena bi moralo biti jasno što se od metode očekuje).

```
public SimpleHashtable ();
public SimpleHashtable(int capacity);
public void put(K key, V value);
public V get(K key);
public int size();
public boolean containsKey(K key);
public boolean containsValue(V value);
public void remove(K key);
public boolean isEmpty();
public String toString();
```

Metoda `put` pozvana s ključem koji u tablici već postoji ažurira postojeći par novom vrijednošću; metoda ne dodaje još jedan par s istim ključem ali drugom vrijednosti.

Metoda `get` pozvana s ključem koji ne tablici ne postoji vraća `null`.

Što možete zaključiti o složenosti metode `containsKey` a što o složenosti metode `containsValue` u ovako implementiranoj kolekciji (uz pretpostavku da je broj parova dodanih u tablicu dosta manji od broja slotova tablice te da funkcija sažetka radi dobro raspršenje)?

Zadatak 3.

Ovaj zadatak rješavate kao nadogradnju prethodnog zadatka u istom projektu.
Modificirajte definiciju razreda `SimpleHashtable<K,V>` tako da definirate da razred implementira sučelje `Iterable<SimpleHashtable.TableEntry<K,V>>`, kako je prikazano u nastavku.

```
public class SimpleHashtable<K,V> implements Iterable<SimpleHashtable.TableEntry<K,V>> { … }
```

Uočite: objekti koje vraćaju stvoreni iteratori su elementi tipa `TableEntry`.
Kako biste riješili zadatak, u razred `SimpleHashtable` morat ćete dodati metodu tvornicu koja je definirana u sučelju `Iterable`, a koja će proizvoditi iteratore koji se mogu koristiti za obilazak po svim parovima koji su trenutno pohranjeni u tablici, i to redosljedom kojim se nalaze u tablici ako se tablica prolazi od slota 0.
Ideja je osigurati da možete napisati sljedeći isječak koda:

```
package hr.fer.oop.lab3;

public class Primjer {

	public static void main(String[] args) {
		// create collection:
		SimpleHashtable<String,Integer> examMarks = new SimpleHashtable<>(2);

		// fill data:
		examMarks.put("Ivana", Integer.valueOf(2));
		examMarks.put("Ante", Integer.valueOf(2));
		examMarks.put("Jasna", Integer.valueOf(2));
		examMarks.put("Kristina", Integer.valueOf(5));
		examMarks.put("Ivana", Integer.valueOf(5)); // overwrites old grade for Ivana

		System.out.println("Prvi obilazak kroz tablicu");
		for(SimpleHashtable.TableEntry<String,Integer> entry : examMarks) {
			System.out.printf("%s => %s%n", entry.getKey(), entry.getValue());
		}

		System.out.println("Drugi obilazak kroz tablicu");
		for(SimpleHashtable.TableEntry<String,Integer> entry : examMarks) {
			System.out.printf("%s => %s%n", entry.getKey(), entry.getValue());
		}


	}
}
```

Ovaj kod trebao bi rezultirati sljedećim ispisom:

```
Prvi obilazak kroz tablicu
Ante => 2
Ivana => 5
Jasna => 2
Kristina => 5
Drugi obilazak kroz tablicu
Ante => 2
Ivana => 5
Jasna => 2
Kristina => 5
```