2 Upis i Ispis

Ispis na standardni izlaz (Konzola)

Čistim baratanjem podataka, premeštanjem iz jedne promenjive u drugu, vršenje kompleksnih operacija su stvari koje same za sebe ne nose nikakav značaj; ne ukoliko te podatke ne možemo nikako da obradimo, nikako da ih pregledamo.

Najednostavniji način da pregledamo naše podatke je da ih ispišemo na standardni izlaz, odnosno konzolu. Skoro svaki programski jezik ima sposobnost da ispiše svoje podatke u konzolu. Ovo potiče iz dana kada su kompjuteri bili mašine dizajnirane za mnogo uži spektar primene. Tada, jedini ljudi koji su koristili računare su bili baš oni koji su pisali programe na njima, i zato nije bilo potrebe da se ti podaci prikažu na ekranima, za specijalnim efektima itd. Često je postojao jedan izlazni terminal (uglavnom neki LCD ekran) na kome su se prikazivali podaci. Kompjuteri su vremenom evoluirali sa proizvodnjom ličnih računara, pa se razvila i potreba za boljim, jasnijim i sveobuhvatnijim prikazivanjem podataka. Doduše, za potrebe programera, jednostavni tekstualni ispis na prostu koznolu (danas su to uglavnom terminali) je ostao u upotrebi. Ovakv ispis vidi ulogu u debugovanju koda, i često je namenjen samo za programere tokom ciklusa kreiranja proizvoda. Za naše potrebe ovog kursa, takav ispis će biti i više nego dovoljan, te ubuduće, ukoliko nažnačim da je upitanju samo 'ispis', misliću na standardni konzolni ispis.

Da bi se neki tekst ispisao na standardnom izlazu, potrebno je pozvati tok `System.out.` . Tada koristimo dve funkcije:

- `System.out.print(<neki Stringovni literal>);` <- naredba koja ispisuje jedan red teksta u konzolu
- `System.out.println(<neki Stringovni literal>);` <- naredba koja ispisuje jedan red teksta u konzolu i šalje karakter za novi red ('\n').
  U praksi, to bi izgledalo nekako ovako:

```
1  package neki.paket;
2
3  public class NekaKlasa {
4
5      public static void main(String[] args) {
6          System.out.print("Zdravo ");
7          System.out.print(" svete!");
8      }
9  }
```

Gornji kod će ispisati nisku "Zdravo svete!" u konozli. Ukoliko bi vezali još jednu takvu komandu ona bi se nadovezala odmah nakon znaka '!'. Ukoliko želimo da posle našeg ispisa, sedeći ispis bude u novom, zasebnom redu, onda koristimo `System.out.println()` komandu:

```
1  package neki.paket;
2
3  public class NekaKlasa {
4
5      public static void main(String[] args) {
6          System.out.println("Zdravo svete!");
7      }
8  }
```

Naravno, Java je dovoljno inteligentna da može da razazna i sama konvertuje neke ostale osnovne tipove u stringovni literal. Tako na primer, naredni kodovi će raditi bez ikakvog problema:

```
1  package neki.paket;
2
3  public class NekaKlasa {
4
5      public static void main(String[] args) {
6          int x = 5;
7          int y = 7;
8          System.out.println(x);
9          System.out.println(y);
10         System.out.println(2.1*(x + y));
11     }
```

pa će gornji kod ispisati:

```
1  5
2  7
3  25.2
```

Sabiranje (pa i oduzimanje, mnozenje, deljenje ...) su matematičke operacije. Matematička operacije je funkcija (u daljim lekcijama ćemo se detaljnije pozabaviti šta su to funkcije tačno) koje uzimaju elemente iz jednog istog skupa i dodeljuju vrednost iz tog istog skupa. Primeri:

Sabiranje prirodnih brojeva: $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$
npr: $2 + 3 = 5$
Množenje realnih brojeva: $\cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
npr: $33.3 \cdot 3 = 99.9$

Primetimo da su ove operacije vrlo apstraktne. "Sabiranje" je ništa drugo do jedne operacije koja je asocijativna i ima element koji

kada se sabere sa bilo kojim drugim elementom daje vrednost baš tog drugog elementa (to je 0). Dokle god se mi dogovorimo kako to "sabiranje" funkcioniše, možemo da sabiramo bilo šta, babe sa babama, žabe sa žabama ... (ali nikako babe i žabe! Napomenuli smo da moraju biti isti skupovi tj objekti u pitanju!!!).
To nas dovodi do sledeće važne teme, **sabiranje niski**

$$(x + y) + z = x + (y + z) \qquad (\text{Asocijativnost})$$
$$x + 0 = 0 + x = x \qquad (\text{Neutral})$$
$$- \; - -$$
$$x + y = y + x \qquad (\text{Komutativnost})$$

Sabiranje niski

Kao što smo napomenuli, želimo da generališemo sabiranje tako da obuhvatimo i sabiranje niski, odnosno rečenica (tj. čitavog teksta). Na prvi pogled nešto poput:
`"Prva rečenica." + "Druga rečenica".`
Nema nikakvog smisla samo za sebe, ali ako možemo nekako da se dogovorimo kako se sabiraju rečenice, pazeći da tako definisano sabiranje bude asocijativno i da ima neutral, onda ćemo lako moći da saberemo bilo koje dve rečenice.

Najprirodniji način da to uradimo je da ta dva Stringovna literala koja želimo da saberemo nadovežemo: na kraj prvog nadovežemo drugi. Tako, za prethodni primer bi smo dobili:
`"Prva rečenica." + "Druga rečenica." = "Prva rečenica.Druga rečenica.`
Ovakvo sabiranje se zove **nadovezivanje** odnosno **konkatenacija**.
Uverimo se da tako definisano sabiranje jeste asocijativno:
(Primetimo da ako želimo razmake izmedju interpunkcijskih znakova moramo dodati je dan razmak, tj karakter ' ' na kraj prve

rečenice ili na početak druge!)
Pazimo na redosled operacija unutrar i izvan zagrada!

$$\begin{aligned}&\text{"Prva rečenica. "} + (\text{"Druga rečenica."} + \text{" Treća rečenica."})\\ &= \text{"Prva rečenica. "} + \text{Druga rečenica. Treća rečenica."}\\ &= \text{"Prva rečenica. Druga rečenica. Treća rečenica."}\end{aligned}$$

$$\begin{aligned}&(\text{"Prva rečenica ."} + \text{"Druga rečenica."}) + \text{" Treća rečenica."}\\ &= \text{"Prva rečenica. Druga rečenica."} + \text{" Treća rečenica."}\\ &= \text{"Prva rečenica. Druga rečenica. Treća rečenica."}\end{aligned}$$

... što vidimo da jeste slučaj.
Ostalo je jedino da definišemo šta je to neutral za sabiranje niski. Kakva je to niska koja kada se sabere sa bilo kojom drugom niskom, bilo da je ona prva niska ili druga niska koja se sabira, ne menja njen sadržaj? Prosto, to je **prazna niska**, odnosno `""`, niska bez ijednog karaktera.

$$\begin{aligned}&\text{"Neka rečenica."} + \text{""} = \text{"Neka rečenica."}\\ &\text{""} + \text{"Neki tekst. Od više rečenica.} = \text{"Neki tekst. Od više rečenica."}\end{aligned}$$

Primetmo takodje, da prazna niska nije isto što i niska sa samo praznim karakterima (space, enter, tab)!!!:

$$\begin{aligned}(\text{"Jedan"} + \text{""}) + \text{"Dva"} &= \text{"Jedan"} + \text{"Dva"} = \text{"JedanDva"}\\ &\neq\\ (\text{"Jedan"} + \text{" "}) + \text{"Dva"} &= \text{"Jedan "} + \text{"Dva"} = \text{"Jedan Dva"}\end{aligned}$$

Sa ovim smo pokazali da ovako definisano sabiranje niski, kao njihova konkatenacija, zadovoljava sve uslove "sabiranja", te je tako nešto dozvoljeno i u Javi:

```
1  package neki.paket;
2
3  public class NekaKlasa {
4
```

```
5       public static void main(String[] args) {
6           String s1 = "Zdravo ";
7           String s2 = "Svete!";
8           String s3 = s1 + s2;
9           System.out.println(s3);
10      }
11  }
12
```

```
1  Output: "Zdravo Svete!"
```

Važno je još napomenuti da su asocijativnost i neutral samo potrebni uslovi da bi se nešto računalo kao sabiranje. Specijalni slučajevi sabiranja mogu imati svoje specijalne (dodatne) uslove. Tako naprim sabiranje brojeva je i komutativno, dok sabiranje niski nije!

$$2 + 3 = 5 = 3 + 2$$

$$\text{"Jedan"} + \text{"Dva"} = \text{"JedanDva"} \neq \text{"Dva"} + \text{"Jedan"} = \text{"DvaJedan"}$$

Upis sa standardnog inputa

Do sada smo videli kako da dolelimo promenjivima njihove vrednosti i kako da ih ispišemo na standardni izlaz. Sa dosadašnjim znanjem, ukoliko bi smo želeli da ispišemo poruku dobrodošlice korisniku našeg programa, npr "Zdravo <ime i prezime>", morali bi smo pre svakog pokretanja programa da ručno promenimo vrednosti imena i prezimena za svakog novog korisnika. U praksi ovako nešto ne bi bilo nimalo praktično i često programi koje koristimo traže od nas dinamički da unosimo razne podatke (naše ime, godište, ime našeg karaktera u igrici, da biramo pozadinu ekrana ...). U realnosti, ovi odabiri se uglavnom vrše preko neke vrste korisničkog interfejca (Graphical User Interface),

ali za potrebe našeg kursa, jedini način na koji ćemo mi slati programu informacije dinamički je u konzolu, putem tastature.

Da bi smo mogli da u Javi šaljemo podatke sa tastature, moramo da kreiramo objekat klase `Scanner` (za sada nećemo se dugo zadržavati na činjenici da kreiramo objekat jedne klase, već cemo tu promenjivu prihvatiti takvu kakva je). Klasa `Scanner` nije deo srži Jave i kao takva ona se mora importovati. Proces importovanja se vrši izmedju naredbi za lokaciju paketa i početka klase pomoću ključne reči `import` koju prati putanja do zadate klase koju želimo da importujemo. Klasa `Scanner` je do Java utility paketa te njeno importovanje i kreiranje promenjive (objekta) te klase se vrši na sledeći način:

```
1  package neki.paket;
2
3  import java.util.Scanner; //Importujemo Scanner iz paketa
   java iz podpaketa util
4
5  public class NekaKlasa {
6
7      public static void main(String[] args) {
8          Scanner sc = new Scanner(System.in);
9          /*
10         Kreirali smo objekat klase Scanner pod nazivom sc
   (mogli smo bilo koju neključnu reč da iskoristimo)
11         i zadali smo joj da prikuplja podatke iz
   System.in toka.
12     }
13 }
```

Kako je promenjiva *sc* jedan objekat, sve operacije koje radimo nad njome se pišu tako što navedemo njeno ime, prateći znakom `.` pa zatim funkcionalnost koja nam treba.

Iako ovako definisan skener ima niz interesantnih funkcionalnosti, nama će trebati samo jedan manji deo, koji se tiče prikupljanja podataka. Podatke šaljemo skeneru preko konzole. Pripremanje podataka se vrši tako što se željeni podaći napišu sa jednim razmakom izmedju svaka dva podatka, a slanje na obradu se izvršava pritiskom `Enter` tastera na tastaturi. Alternativno i češće, svaki podatak se piše u zasebnom redu, tj. svaki pripremljen podatak se odmah šalje na obradu.
npr: `2 3.14 c 5.55 true` (Pripremljeni su `int`, `double`, `char`, `double`, `boolean` podatak i za njihovu obradu potrebno je kliknuti `Enter` taster

Neke od najznačajnijih komandi za klasu `Scanner` su:

- `sc.nextInt();` - skener uzima naredni pripremljen podatak i tretira ga kao ceo broj
- `sc.nextDouble();` - skener uzima naredni pripremljen podatak i tretira ga kao realan broj
- `sc.nextBoolean();` - skener uzima naredni pripremljen podatak i tretira ga kao tip logički tip
- `sc.next();` - skener uzima narednu reč i tretira je kao nisku
- `sc.nextLine();` - skener uzima sve podatke do kraja (sve do, ali ne i `Enter` tastera) i tretira ih kao jednu veliku nisku
  Pored ovih, postoje i slične komande za ostale, redje korišćene podatke.

Primetimo da nema nema izdvojene komande za prikupljanje karaktera. Za potrebe prikupljanja karaktera koristićemo komande za prikupljanje niski, sa malom modifikacijom koju za sada moramo napamet da naučimo (ali biće objašnjena kasnije):

>

```
char c = sc.next().charAt(0); //ili alternativno, sc.nextLine().charAt(0);
```

Pogledajmo sada kako bi izlgedao program koji zahteva od korisnika da unese svoje ime, prezime i broj godina i ispisuje poruku dobrodošlice:

```
package neki.paket;

import java.util.Scanner;

public class Dobrodosli {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Unesite vase ime, prezime i broj godina u zasebnim redovima:");
        String ime = sc.nextLine();
        String prezime = sc.nextLine();
        int brojGodina = sc.nextInt();
        String poruka = ime + " " prezime + " (" + brojGodina + ")";
        System.out.println("Dobrodosli " + poruka);
    }
}
```

```
Input:
Nikola
Tesla
33
Output: Dobrodosli Nikola Tesla (33)
```