Programiranje 1

Poglavje 1: Uvod

Poglavje 2: Osnovni pojmi

Luka Fürst

Predavanja

- vsak četrtek od 8:15 do 11:00

- Luka Fürst
- `luka.fuerst@fri.uni-lj.si`
- govorilne ure
  - po dogovoru

Vaje

- po urniku
- pričetek v ponedeljek, 7. oktobra

- Marko Poženel
- Jure Žabkar
- Manca Žerovnik
- Luka Fürst

Režim

- izpit
  - pisni del (možnih 100 točk)
  - ustni del
- pogoj za pozitivno oceno
  - ≥ 50 točk na pisnem delu
  - opravljen ustni del
- ocena
  - t: število točk na pisnem izpitu
  - $50 \leq t < 60 \mapsto 6$
  - $60 \leq t < 70 \mapsto 7$
  - $70 \leq t < 80 \mapsto 8$
  - $80 \leq t < 90 \mapsto 9$
  - $t \geq 90 \mapsto 10$

Sprotno delo

- domače naloge
  - objava: vsak petek, začenši z 11. oktobrom
  - oddaja: do izteka naslednje nedelje
- dodatne naloge
  - že na Učilnici
- izzivi
  - občasno
- neobvezno, a zelo priporočljivo

Literatura

- osnovna
  - Luka Fürst. *Java od začetka*. Založba FRI, 2023.
- Te prosojnice ne zadoščajo!
  - za določen (relativno majhen) delež snovi na predavanjih ne bo časa
  - v knjigi je več razlage, primerov, nalog ...

Literatura

- dodatna
  - Allen B. Downey, Chris Mayfield. *Think Java: How to Think Like a Computer Scientist*, 2nd Edition. O'Reilly, 2020.
  - Barry A. Burd. *Beginning Programming with Java For Dummies*, 6th Edition. For Dummies, 2021.
  - Viljan Mahnič, Luka Fürst, Igor Rožanc. *Java skozi primere*. Bi-Tim, 2008.
- zahtevnejša
  - Herbert Schildt. *Java: The Complete Reference*, 12th Edition. McGraw-Hill Education, 2021.
  - Joshua Bloch. *Effective Java*, 3rd Edition. Addison-Wesley, 2018.
  - Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft. *Java 8 in Action*. Manning, 2014.

Računalnik

- izjemno hiter
- izjemno natančen
- izjemno neumen
- . . . »pameten«, če ga tako sprogramiramo

Program, programiranje, programski jezik

- program
  - datoteka z navodili računalniku
- programiranje
  - pisanje programov
- programski jezik
  - skupek natančnih in nedvoumnih pravil, po katerih pišemo programe
- java
  - primer sodobnega programskega jezika

Zakaj java?

- industrijsko zanimiva
- močna podporna skupnost
- nenehen razvoj
- platformna neodvisnost
  - javanski navidezni stroj (JVM)
- stroga tipiziranost
- prevajanje + izvajanje
- časovna in prostorska učinkovitost
- bogata in preizkušena standardna knjižnica

Okvirni razpored

- osnovni pojmi
- krmilni konstrukti
- metode
- tabele
- razredi in objekti
- dedovanje
- generiki
- vmesniki
- vsebovalniki
- lambde

Namestitev jave

- Linux (Ubuntu)
  - `sudo apt install openjdk-`*`različica`*`-jdk`
- Windows
  - `java.oracle.com`
  - Java SE (zadnja različica)
  - JDK (ne JRE!)
  - dodajte absolutno pot do imenik z namestitvijo jave v spremenljivko `PATH`
  - če spremenljivka `CLASSPATH` obstaja, mora vsebovati piko (`.`)

Razvojno okolje

- programerski urejevalnik
  - gedit (Linux)
  - Notepad++ (Windows)
  - Sublime Text, Visual Studio Code, Vim ...
- terminal operacijskega sistema

Javanska konzola

- `jshell`
- vnesemo izraz ali stavek in takoj dobimo odgovor
- pravila so svobodnejša kot pri samostojnih programih

Števila

- računalniki računajo, zato skoraj vsak programski jezik omogoča delo s števili
- cela števila
  - `42`, `0`, `-17`, `1234567890` ...
- realna števila
  - vsebujejo decimalno piko
  - `3.14`, `-25.0`, `0.0036`, `2.7e+6` ...

Izraz in vrednost

- pravilno strukturirano zaporedje operandov in operatorjev
- vsak izraz ima vrednost (podatek določenega tipa)
- izraz `1 + 2` ima vrednost 3
- izraz `1 < 2` ima vrednost `true`

Aritmetični operatorji in izrazi

Operator	Pomen	Primer izraza	Vrednost
+	seštevanje	`4 + 3`	`7`
-	odštevanje	`2 - 6`	`-4`
*	množenje	`3 * 5`	`15`
/	deljenje	`7 / 2`	`3`
/	deljenje	`7.0 / 2`	`3.5`
%	ostanek pri deljenju	`7 % 2`	`1`

- operatorji *, / in % imajo prednost pred + in -
- operatorji na istem nivoju se računajo od leve proti desni
- vrstni red računanja lahko spremenimo z oklepaji
- če je vsaj en operand realno število, se izraz izračuna v domeni realnih števil

Spremenljivka

- poimenovan prostorček v pomnilniku, ki hrani vrednost določenega tipa
- npr. spremenljivka tipa `int` lahko hrani celo število z intervala $[-2^{31}, 2^{31} - 1]$
- deklaracija spremenljivke
  - `int a;`
  - uvedemo spremenljivko `a` tipa `int`
  - spremenljivka v tem trenutku še nima vrednosti (ni definirana)

Spremenljivka

- prireditev vrednosti spremenljivki
  - nastavitev vrednosti
  - `a = 42;`
  - spremenljivka a dobi vrednost 42
- prva prireditev = inicializacija
- deklaracijo in inicializacijo praviloma združimo
  - `int a = 42;`

Tip

- vsaka vrednost, vsaka spremenljivka in vsak izraz ima svoj tip
- tip podaja vrsto podatka in zalogo možnih vrednosti
- v javi obstaja veliko tipov
  - celoštevilski tipi
  - realnoštevilski tipi
  - logični tip
  - znak
  - niz (kos besedila)
  - ...

Celoštevilski podatkovni tipi

- razlikujejo se po obsegu

tip	obseg
`byte`	$[-2^7, 2^7 - 1]$
`short`	$[-2^{15}, 2^{15} - 1]$
`int`	$[-2^{31}, 2^{31} - 1]$
`long`	$[-2^{63}, 2^{63} - 1]$

- privzeti celoštevilski tip je `int`
- `42` je tipa `int`
- `42L` je tipa `long`
- `(byte) 42` je tipa `byte`

Realnoštevilski podatkovni tipi

- razlikujejo se po obsegu in natančnosti

tip	natančnost	obseg
`float`	7 mest	od ca. -10^{38} do ca. 10^{38}
`double`	15 mest	od ca. -10^{307} do ca. 10^{307}

- privzeti realnoštevilski tip je `double`
- `42.0` je tipa `double`
- `42.0f` je tipa `float`

Izrecna pretvorba tipa

- če sta oba operanda celi števili, se operacija izvede v domeni celih števil
- če je vsaj en operand realno število, se operacija izvede v domeni realnih števil
- podobno velja za tipa `int` in `long`
- z izrecno pretvorbo tipa lahko spremenimo tip vrednosti

  (*tip*) *izraz*

- pretvorba realnoštevilskega tipa v celoštevilski tip odreže decimalke

Izrecna pretvorba tipa

```
int a = 5;
int b = 3;
int c = a / b;                          // 1
double d = (double) (a / b);            // 1.0
double e = ((double) a) / b;            // 1.6666...
double f = a / (double) b;              // 1.6666...
double g = ((double) a) / (double) b;   // 1.6666...
int h = (int) e;                        // 1
```

Znak (tip char)

- znaki so predstavljeni s celimi števili
  - kode ASCII / Unicode
- tip `char` je dejansko celoštevilski tip
  - obseg: $[0, 2^{16} - 1]$

```
char znak = 'A';
int koda = (int) znak;     // 65
char znak2 = (char) 97;    // 'a'
```

- velike črke angleške abecede imajo zaporedne kode
- enako velja za male črke in števke

Niz (tip String)

- kos besedila znotraj para narekovajev (znakov ")
- "Dober dan!", "java", "42" ...
- za lepljenje nizov lahko uporabljamo operator +

```
String a = "Dober";
String b = "dan!";
System.out.println(a + " " + b);   // Dober dan!
```

- če zlepimo niz in število, se število samodejno pretvori v niz

```
int odgovor = 42;
System.out.println("Odgovor znaša " + odgovor);
```

Stavek

- element programa, ki ima nek učinek
- enostaven stavek
  - `int a;`
  - `double b = 3.14;`
  - `b = 3.0 / 4.0;`
  - `System.out.println("Pozdravljen, svet!");`
- kompleksen stavek
  - ```
    if (cena <= 100) {
        System.out.println("Kupim!");
    } else {
        System.out.println("Predrago ...");
    }
    ```
    ```

Prireditveni stavek

- oblika
  - *spremenljivka* = *izraz*;
- izvedba
  - izraz na desni strani se izračuna
  - rezultat se zapiše v spremenljivko na levi strani
- primer

```
int a = 3;
int b = 4;
a = b + 5;          // a: 9, b: 4
b = b - 3;          // a: 9, b: 1
a = 2 * a + b;      // a: 19, b: 1
```

Prvi program

- odpremo urejevalnik
- odtipkamo

```
public class Prvi {
    public static void main(String[] args) {
        System.out.println("Pozdravljen, svet!");
    }
}
```

- shranimo kot `Prvi.java` (velika začetnica!)

Prevajanje in zagon programa

- 1. faza: prevajanje
  - prevajalnik pretvori program v obliko, ki jo lahko izvede izvajalnik (JVM)
  - `javac Prvi.java`
  - nastane datoteka `Prvi.class`
- 2. faza: zagon
  - izvajalnik izvrši prevedeni program
  - `java Prvi`

Leksikalni elementi programa

- rezervirane besede
  - `public`, `static`, `class`, `void` ...
- imena
  - `Prvi`, `main`, `String`, `System`, `out`, `println` ...
- števila (cela in realna)
  - `42`, `-7`, `3.0`, `-2.5e+4` ...
- nizi
  - `"sreda"`, `"Pozdravljen, svet!"` ...
- ločila, oklepaji, operatorji ...
- komentarji
  - `// vrstični`
  - `/* bločni */`

Lepopisna pravila

- zamiki
  - če se prejšnja vrstica konča z {, povečamo zamik trenutne vrstice za 4 presledke
  - če se trenutna vrstica začne z }, zmanjšamo njen zamik za 4 presledke
- vsak enostaven stavek v svojo vrstico
- presledki
  - okrog operatorjev
  - pred {
  - ne pa pri klicu metode

Odpravljanje prevajalnikovih napak

- če namesto `System` odtipkamo `system` ...

```
Prvi.java:5: error: package system does not exist
        system.out.println("Pozdravljen, svet!");
               ^
1 error
```

- položaj napake
  - datoteka `Prvi.java`
  - vrstica 5

Napaka pri prevajanju in napaka pri izvajanju

- napaka pri prevajanju je ponavadi posledica sintaktičnih napak, nedefiniranih imen (spremenljivk, metod, razredov ...), neskladja tipov ...
- nekatere napake se pokažejo šele pri izvajanju
  - napaka pri izvajanju = izjema
- primer: deljenje z ničlo

```
public class Program {
    public static void main(String[] args) {
        int a = 1 / 0;
    }
}
```

- program se prevede, pri izvajanju pa se sproži izjema tipa `ArithmeticException`

Bločna zgradba programa

- blok = kos programa znotraj { in }
- vse spremenljivke obstajajo samo do konca bloka, v katerem so deklarirane

```
int a = 1;
{
    int b = 2;
    {
        int c = 3;
    }
    // c ne obstaja več
    int d = 4;
}
// b in d ne obstajata več
```

Vhod in izhod

- vhod = tisto, kar program dobi od okolja (npr. od uporabnika)
- izhod = tisto, kar program da okolju
- vsi naši programi bodo brali s standardnega vhoda in pisali na standardni izhod
- standardni vhod: privzeto tipkovnica, a ga lahko preusmerimo
- standardni izhod: privzeto zaslon, a ga lahko preusmerimo

Pisanje na standardni izhod

- `System.out.print(`*nekaj*`);`
  - izpiše *nekaj*
- `System.out.println(`*nekaj*`);`
  - izpiše *nekaj* in doda prelom vrstice

```
int a = 42;
System.out.println(a);     // 42
System.out.println("a");   // a
System.out.println("Vrednost znaša " + a);
                                  // Vrednost znaša 42
```

Branje s standardnega vhoda

- ogrodje

```
import java.util.Scanner;

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

- branje podatkov
  - `int a = sc.nextInt();`
  - `long b = sc.nextLong();`
  - `double d = sc.nextDouble();`

Primer

- program, ki prebere leto rojstva osebe in izpiše njeno starost v letu 2024

```
import java.util.Scanner;

public class Starost {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Vnesite leto rojstva: ");
        int letoRojstva = sc.nextInt();
        int starost = 2024 - letoRojstva;
        System.out.println("Letos praznujete " +
                starost + ". rojstni dan.");
    }
}
```

Primer

- program, ki prebere prevoženo pot v kilometrih in porabljen čas v minutah in izpiše povprečno hitrost v km/h

```
import java.util.Scanner;

public class PovprecnaHitrost {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Vnesite prevoženo pot v km: ");
        double pot = sc.nextDouble();
        System.out.print("Vnesite porabljen čas v min: ");
        double cas = sc.nextDouble();
        double hitrost = 60 * pot / cas;
        System.out.println("Povprečna hitrost znaša " +
                hitrost + " km/h.");
    }
}
```

Poganjanje programa

- vhod prek tipkovnice, izhod na zaslon
  - `java Program`
- vhod prek programa `echo`, izhod na zaslon
  - `echo` *podatki* `| java Program`
- vhod iz datoteke `test01.in`, izhod na zaslon
  - `java Program < test01.in`
  - `cat test01.in | java Program`
- vhod iz datoteke `test01.in`, izhod v datoteko `test01.res`
  - `java Program < test01.in > test01.res`
- primerjava dejanskega in pričakovanega izhoda (`test01.out`)
  - `diff test01.res test01.out`

Avtomatsko testiranje

- `tj.exe` *Program.java testi rezultati*
  - *testi*: imenik s testnimi primeri (datotekami `test*.in` in `test*.out`)
  - *rezultati*: imenik, v katerega se shranijo rezultati testiranja
  - *rezultati*`/prikaz.htm`: poročilo, ki ga lahko odpremo z brskalnikom

Preverjanje in dokazovanje

- če želimo biti prepričani, da program res deluje, ga moramo preveriti na vseh možnih vhodih
- v praksi je to skoraj vedno nemogoče
  - vsota dveh števil tipa `int`: 2^{64} možnosti!
- alternativa: matematični dokaz pravilnosti
- v praksi se ponavadi zadovoljimo z (dovolj temeljitim) naborom testnih primerov