Verónica Gaspes
ITE
2 november 2023

Inlämningsuppgifter för vecka 2

Anmärkningar

- Hur man lämnar in och vad som ska lämnas in förklaras i Blackboard och i *README* filen som följer med zip-paketet. Denna dokument beskriver uppgifterna. Varje uppgift finns i en .java fil. I varje .java fil finns en kommentar som förklarar uppgiften på engelska.
- När du löser veckans uppgifter behöver du bara använda det vi har presenterat under första och andra veckan:
  - Många operationer och några metoder för att bygga upp uttryck i olika typer,
  - if- och switch kommandon,
  - for- och while kommandon, och
  - tilldelning och System.out.println
- Inför tentan måste du kunna
  - att det finns både kommandon och uttryck
  - att det finns olika typer
  - förklara vad de kommandon vi har studerat gör
  - förklara vilka typer och värden några uttryck har
  - använda uttryck, tilldelning, System.out.println, if, switch, for och while i enkla program samt förklara vad program som är uppbyggda med dessa gör när de körs.

  Efter inlämningsuppgifterna finns några uppgifter från gamla tentor som handlar om detta: de är här bara för att du ska kunna bekanta dig med uppgifter av tenta typ, de ska INTE lämnas in! Har du frågor om de kan du ta upp det på Drop-in passet!

Inlämningsuppgifter - del 1

1. I filen *E1.java* finns följande kod, inklusive en kommentar som förklarar vad som behöver göras på engelska:

```
/*

  This program does not compile!
  Correct it so that it compiles and does
  what you think the programmer tried to achieve.

*/
public class E1{

  public static void main(String[] args) {
    int x = Intger.parseInt(args[0]);
    int y = Integer.parseInt(args[1]);

    if (x = y){
      System.out.println("x is equal to y");
    }

  }
}
```

   Det du behöver göra är:

   (a) Spara filen *E1.java* i mappen där du ska arbeta.

   (b) Kompilera programmet med kommandot

   ```
   javac E1.java
   ```

   Om allt *går bra* kommer det att gå fel! Den text som finns i filen är inte ett korrekt java program: kompilatorn kommer att skriva ett felmeddelande. Du måste läsa meddelandet, förstå det och hitta felet i filen.

   (c) Ändra i programmet så att felet rättas till. För att göra det måste du försöka förstå vad programmeraren försökte skriva och vad som blev fel.

   (d) Spara och kompilera igen. Gör om tills du inte får ett kompileringsfel.

   (e) Kör programmet med

   ```
   java E1 1 2
   ```

   Du borde se ingen utskrift.

   (f) Kör programmet med

   ```
   java E1 1 1
   ```

   Du borde se utskriften

   ```
   x is equal to y
   ```

2. I filen *E2.java* finns ett Java program. Det du behöver göra är:

   (a) Spara filen *E2.java* i mappen där du ska arbeta.

   (b) Kompilera programmet med kommandot

```
javac E2.java
```

(c) Köra programmet med kommandot

```
java E2 9
```

Du borde se utskriften

```
I don't know
```

Du kan köra programmet med andra heltal som kommandoradsargument, du kommer att få samma utskrift.

(d) Ändra i filen så att

i. programmet genererar ett slump heltal mellan 1 och 10 (inklusive 1 och 10),
ii. jämför slumptalet med talet från kommandoraden och
iii. tilldelar en text till variabeln `message` så att utskriften blir `Well done!` om användaren gissade rätt och `No luck, play once more!` om användaren gissade fel (användaren kan då välja köra programmet en gång till).

Tanken är att det som användaren anger i kommandoraden är en gissning av vad programmet kommer att generera!

(e) Spara, kompilera och kör några gånger.

(f) För att övertyga dig själv om att programmet fungerar kan du göra följande:

i. Ändra i programmet så att det även skriver ut både slumptalet och användarens gissning.
ii. Ändra i programmet så att slumptalet som genereras är bara mellan 1 och 2.
iii. Kör flera gånger och kolla att om bägge talen är lika så är utskriften `Well done!` och annars `No luck, play once more!`.

OBS! För att lämna in måste du återställa programmet så att det fungerar för slumptal mellan 1 och 10 och det inte skriver ut talen, bara meddelanden!

3. I filen *E3.java* finns ett Java program. Det du behöver göra är:

(a) Spara filen *E3.java* i mappen där du ska arbeta.

(b) Kompilera programmet med kommandot

```
javac E3.java
```

(c) Köra programmet med kommandot

```
java E3 Rock
```

Du borde se utskriften

```
You say Rock
I say Rock
I don't know
```

Kör programmet även med

```
java E3 Scissors
You say Scissors
I say Rock
I don't know
```

```
java E3 Bag
```

```
You say Bag
I say Rock
I don't know
```

Tanken är att programmet ska spela Sten-Sax-Påse med användaren. Programmet är nästan klar: det saknas bara den delen där programmet beräknar det som ska tilldelas variabeln `programSays` (ett slumptal 0, 1 eller 3) samt det som ska tilldelas variabeln `message` som ska vara enligt:

- `"You win!"` om användaren vinner,
- `"I win!"` om programmet vinner eller
- `"We are even!"` annars.

(d) Ändra i filen där det står `// Your code here` för att kunna tilldela värdena till `programSays` och till `message` enligt förklaringen ovan.

(e) Spara filen innan du kompilerar och kör på samma sätt som ovan. Kontrollera att vem som vinner stämmer med det som användaren och datorn säger.

4. I filen *E4.java* finns ett Java program. Det du behöver göra är att spara och kompilera som i tidigare uppgifter för att sedan

(a) Köra programmet med

```
java E4
```

Du borde se utskriften

```
And that was all!
```

(b) Ändra i filen så att den skriver ut alla värden som variablerna `m` och `n` tar när man kör programmet. När programmet skriver ut start värdena för variablerna

```
The initial value of n is:
The initial value of m is:
```

När programmet skriver ut sista värdena som variablerna får måste det också visas i början av respektive rad:

```
The final value of n is:
The final value of m is:
```

För alla andra värden som variablerna tar måste det också visas i början av respektive rad:

```
n:
m:
```

(c) Spara filen innan du kompilerar och kör på samma sätt som innan. Utskriften ska då ha följande form (utan ⋮ och med rätt värden. Det blir flera rader!):

```
The initial value of n is:  det första värdet för n
The initial value of m is:  det första värdet för m
n:  nästa värdet för n
m:  nästa värdet för m
⋮
n:  nästa värdet för n
m:  nästa värdet för m
The final value of n is:  sista värdet för n
The final value of m is:  sista värdet för m
```

```
And that was all!
```

Kan du beskriva i ord vad slutvärdena är i förhållanden till startvärdena?

5. I filen *E5.java* finns ett Java program. Spara filen i rätt mapp och kompilera. Kör med ett komandoradsargument som måste vara ett positivt heltal. Om kommandoradsargumentet är 10 bör du se utskriften:

   ```
   The positive odd numbers smaller or equal than 10 are:
   ```

   Ändra i filen så att utskriften visar alla positiva udda tal som är mindre eller lika med kommandoradsargumentet. Här är några exempel på hur utskriften bör se ut när man kör programmet med olika indata:

   ```
   java E5 1
   The positive odd numbers smaller or equal than 1 are:
   1

   java E5 10
   The positive odd numbers smaller or equal than 10 are:
   1
   3
   5
   7
   9

   java E5 9
   The positive odd numbers smaller or equal than 9 are:
   1
   3
   5
   7
   9
   ```

6. I filen *E6.java* finns ett Java program. Spara filen i rätt mapp, kompilera och kör med ett komandoradsargument som måste vara ett positivt heltal. Om kommandoradsargumentet är 10 bör du se utskriften:

   ```
   The first 10 odd numbers are:
   ```

   Ändra i filen så att utskriften visar de första positiva udda tal, så många som kommandoradsargumentet säger. Här är några exempel på hur utskriften bör se ut när man kör programmet med olika indata:

   ```
   java E6 1
   The first 1 odd numbers are:
   1

   java E6 10
   The first 10 odd numbers are:
   1
   3
   5
   7
   ```

```
9
11
13
15
17
19

java E6 9
The first 9 odd numbers are:
1
3
5
7
9
11
13
15
17
```

7. I filen *E7.java* finns ett Java program. Spara filen i rätt mapp, kompilera och kör med ett komandoradsargument som måste vara ett positivt heltal. Om kommandoradsargumentet är 10 bör du se utskriften:

   ```
   java E7 10
   And that was all!
   ```

   Ändra i filen så att programmet skriver ut alla kombinationer av 3 heltal (x, y, z) med $1 \leq x \leq y \leq z \leq n$ där $x^2 + y^2 = z^2$. Värdet för n anges som kommandoradsargument. Här är några exempel på hur utskriften bör se ut när man kör programmet med olika indata:

   ```
   java E7 1
   And that was all!

   java E7 10
   (3, 4, 5)
   (6, 8, 10)
   And that was all!

   java E7 20
   (3, 4, 5)
   (5, 12, 13)
   (6, 8, 10)
   (8, 15, 17)
   (9, 12, 15)
   (12, 16, 20)
   And that was all!
   ```

8. I filen *E8.java* finns ett Java program. Spara filen i rätt mapp, kompilera och kör med tre kommandoradsargument som måste vara flyttal. Programmet löser andragradsekvationen $ax^2 + bx + c = 0$. Här ser du två exempel på användning med utskrift:

   ```
   java E8 1 0 -4
   ```

```
2.0
-2.0

java E8 1 0 4
NaN
NaN
```

Det första indata motsvarar att programmet löser $x^2 - 4 = 0$ och det andra motsvarar att programmet löser $x^2 + 4 = 0$. Den andra ekvationen har ingen lösning i de reella talen eftersom det måste beräkna kvadratroten av ett negativt tal.

Lägg till den kod som behövs för att förbättra programmet så att det inte beräknar kvadratroten av negativa tal. Om indata är sådant att det blir en negativ diskriminant bör programmet avslutas med utskriften `There are no real solutions.`.

Här ser du samma exempel som ovan med det förbättrade programmet:

```
java E8 1 0 -4
2.0
-2.0

java E8 1 0 4
There are no real solutions.
```

9. I filen *E9.java* finns ett Java program. Spara filen i rätt mapp, kompilera och kör med ett kommandoradsargument som måste vara ett positivt heltal. Programmet skriver ut en stjärna per äkta delare som kommandoradsargumentet har. En äkta delare till ett tal är en delare som inte är 1 eller talet själv. Här ser du några exempel:

```
java E9 24
24: ******

java E9 10
10: **

java E9 23
23:
```

Det du ser är att talet 23 har inga äkta delare (talet är ett primtal!), att 10 har två äkta delare (de är 2 och 5) och 24 har sex äkta delare (de är 2, 3, 4, 6, 8 och 12).

(a) Testa programmet med 10, 120 och 124.

(b) Eftersom det är så olika med antalet äkta delare vill vi kunna se dessa stjärnor för flera tal och inte bara för ett. Du ska förbättra programmet så att kommandoradsargumentet står för ett gränsvärde och programmet skriver ut stjärnorna för alla tal som är mindre eller lika med kommandoradsargumentet. Här ser du ett exempel av att köra det förbättrade programmet:

```
java E9 10
1:
2:
3:
```

```
4: *
5:
6: **
7:
8: **
9: *
10: **
```

Inlämningsuppgifter - del 2

I filen *Bonus.java* finns ett Java program. Spara i rätt mapp och kompilera. Uppgiften kommer från kursboken: se uppgift 46 under rubriken **Web Exercises** i introcs.cs.princeton.edu/java/13flow/.

Komplettera koden för att beräkna kontrollsiffran och skriva ut den fullständiga 12 siffriga UPC koden. Testa med exemplet i boken.

Uppgifter från gamla tentor som kan lösas efter vecka 2.

Den blåa texten är ett lösningsförslag.

- Betrakta följande kodfragment:

```
for(int i = 0; i < 5 ; i++) {
   for(int j = i; j < 5; j++) {
      System.out.print("(" + i + ", " + j + ") ");
   }
   System.out.println();
}
```

  1. Vilka är kontrollvariablerna i for-looparna och vilka värden tar de när kodfragmentet körs?
     Kontrollvariablerna är i och j. När kodfragmentet körs tar i värdena 0, 1, 2, 3 och 4 i den ordning. j tar följande värden::
     när i tar värde 0 tar j värdena 0, 1, 2, 3 och 4,
     när i tar värde 1 tar j värdena 1, 2, 3 och 4,
     när i tar värde 2 tar j värdena 2, 3 och 4,
     när i tar värde 3 tar j värdena 3 och 4,
     när i tar värde 4 tar j värde 4.
  2. Ange hur många gånger kommandot `System.out.print("(" + i + ", " + j + ") ");` utförs.
     Kommandot utförs $5 + 4 + 3 + 2 + 1$ gånger (15 gånger)
  3. Ange hur utskriften ser ut.
     (0, 0) (0, 1) (0, 2) (0, 3) (0, 4)
     (1, 1) (1, 2) (1, 3) (1, 4)
     (2, 2) (2, 3) (2, 4)
     (3, 3) (3, 4)
     (4, 4)

- Betrakta följande program:

```
public class One {

    public static void main(String[] args){

        int ex1 = Integer.parseInt(args[0]);
        int ex2 = Integer.parseInt(args[1]);
        int ex3 = Integer.parseInt(args[2]);
        int ex4 = Integer.parseInt(args[3]);
        int ex5 = Integer.parseInt(args[4]);
        int ex6 = Integer.parseInt(args[5]);

        boolean firstFive = ex1 > 0 && ex2 > 0 && ex3 > 0 && ex4 > 0 && ex5 > 0;
        boolean passed1   =  firstFive && ex6 > 0;
        boolean passed2   =  firstFive && (ex1+ex2+ex3+ex4+ex5) > 5;

        if (passed1 || passed2){
            System.out.println("Passed part 1 (at least grade 3!)");
        }
        else{
            System.out.println("Not passed!");
        }

    }
}
```

1. Ange typen och värdet för uttrycket
   `firstFive && (ex1+ex2+ex3+ex4+ex5) > 5`
   när man använder programmet med kommandoraden
   `java One 1 1 2 1 1 0`
   Uttrycket har typ `boolean` och värde `true`.
   Uttrycket har typ `boolean`: det är uppbyggt med operatorn `&&` mellan två booleska uttryck. Det första byggt med `&&` mellan tal jämförelser och det andra är en jämförelse. När man använder programmet med de givna kommandoradsargumenten får `ex1, ex2, ex4, ex5` värde 1 och `ex3` får värde 2. Med detta får `firstFive` värde `true` och `ex1 + ex2 + ex3 + ex4 + ex5` får värde 6 och därför har uttrycket värde `true`.
2. Ange vad programmet skriver ut när man använder programmet med kommandoraden
   `java One 1 1 1 1 1 1`
   Programmet skriver ut

   ```
   Passed part 1 (at least grade 3!)
   ```

   därför att villkoret i `if`-kommandot har värde `true` eftersom `passed1` har värde `true`.

- Betrakta följande kodfragment:

  ```
  int f = 0;
  int g = 1;
  for (int i = 0; i <= 15; i++) {
     System.out.println(f);
     f = f + g;
     g = f - g;
  }
  ```

  Vad skriver kodfragmentet ut?
  Programmet skriver ut:

  ```
  0
  1
  1
  2
  3
  5
  8
  13
  21
  34
  55
  89
  144
  233
  377
  610
  ```

  eftersom under exekvering sker följande (först variablerna får ett start värde och sedan görs `for`-kommandot 16 gånger):

f 0
g 1

Utskrift f: 0
f 1
g 0

Utskrift f: 1
f 1
g 1

Utskrift f: 1
f 2
g 1

Utskrift f: 2
f 3
g 2

Utskrift f: 3
f 5
g 3

Utskrift f: 5
f 8
g 5

Utskrift f: 8
f 13
g 8

Utskrift f: 13
f 21
g 13

Utskrift f: 21
f 34
g 21

Utskrift f: 34
f 55
g 34

Utskrift f: 55
f 89
g 55

Utskrift f: 89
f 144
g 89

Utskrift f: 144
f 233
g 144

Utskrift f: 233
f 377
g 233

Utskrift f: 377
f 610
g 377

Utskrift f: 610
f 987
g 610