HÖGSKOLAN
I HALMSTAD

Verónica Gaspes
ITE
18 november 2022

Inlämningsuppgifter för vecka 3

Anmärkningar

- Hur man lämnar in och vad som ska lämnas in förklaras i Blackboard. 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, andra och tredje veckan:
  - Många operationer och några metoder för att bygga upp uttryck i olika typer,
  - if- och switch kommandon,
  - for- och while kommandon,
  - tilldelning och System.out.println,
  - fält
  - standard input
  - omdirigering av standard input och standard output
- 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.
  - förklara hur man deklarerar, skapar, initierar och går igenom fält,
  - förklara varför programmet måste gå igenom fält för att skriva ut eller jämföra fältets element,
  - använda fält i enkla program och förklara hur programmet fungerar
  - använda standard input och omdirigering av standard input och standard output

  Efter inlämningsuppgifterna finns några uppgifter från gamla tentor som handlar om detta: de finns 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 ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
   Programmet ska användas för att simulera ett stort antal tärningskast med flera tärningar. Programmets användare ska ange hur många gånger tärningarna ska kastas samt hur många tärningar som ska kastas varje gång. Ett par exempel av användning av programmet följer.

   ```
   java E1 5 3
   5
   3
   2 5 3
   4 6 1
   3 2 5
   3 3 5
   2 4 1
   ```

   Programmet användes för att simulera att man fem gånger kastar tre tärningar. Man ser att i utskriften ingår antal försök, antal tärningar och sedan resultaten av alla kast, en per rad.

   ```
   java E1 3 5
   3
   5
   5 1 4 4 2
   6 6 6 4 4
   2 5 5 1 1
   ```

   Programmet användes för att simulera att man tre gånger kastar fem tärningar. Man ser att i utskriften ingår antal försök, antal tärningar och sedan resultaten av alla kast, en per rad.

   Det du behöver göra är:

   (a) Spara filen *E1.java* i mappen där du ska arbeta. Kompilera programmet med kommandot

   `javac E1.java`

   Kör programmet med

   `java E1 0 0`

   Du borde inte se någon utskrift.

   (b) Lägg till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplen ovan.

   (c) Använd programmet för att simulera 1000 kast med 3 tärningar. För att inte behöva se alla 1002 rader på terminalen dirigera om utdatan till en fil med namn *three_dices* genom att köra programmet med kommandot

   `java E1 1000 3 > three_dices`

   istället för att bara köra med `java E1 1000 3`. Då skapas en fil med namn *three_dices* och standard output som är terminalen dirigeras om till den filen. Du kan öppna filen med en text redigerare för att se innehållet (1002 rader där första raden är 1000, andra raden är 3 och de följande 1000 rader innehåller tre heltal mellan 1 och 6).

2. I filen *E2.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
   Programmet ska kunna användas med ett kommandoradsargument som ska vara ett heltal. Programmet ska sedan räkna ut hur många gånger det talet förekommer bland några tal som användaren matar in eftersom. Användaren bestämmer när inmatningen är över genom att använda Ctrl-D (i ett unix system) eller Ctrl-Z (i ett windows system). Här ser du några exempel av användning av programmet.

```
java E2 3
1 2 3 4 5 6
7 6
5
4
3
2
Number 3 occurred 2 times.
```

   Programmet startade efter `java E2 3`. Sedan läste programmet de tal som användaren matar in från tangentbordet: `1 2 3 4 5 6 7 6 5 4 3 2`. Det är operativsystemet som ekar det man skriver i tangentbordet i terminalfönstret. Efter att användaren matar in Ctrl-D eller Ctrl-Z avslutas programmet med att den skriver ut hur många gånger talet 3 förekom bland alla tal som matades in. Programmets utskrift är då bara `Number 3 occurred 2 times.`

```
java E2 9
1
2
3
4 5 6
Number 9 occurred 0 times.
```

   Programmet startade efter `java E2 9`. Utskriften är `Number 9 occurred 0 times.`
   Det du behöver göra är:

   (a) Spara filen *E2.java* i mappen där du ska arbeta. Kompilera programmet med kommandot

   `javac E2.java`

   Kör programmet med kommandot

   `java E2 9`

   Du borde inte se någon utskrift.

   (b) Lägg till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplen ovan.

   (c) Spara, kompilera och testa med samma exempel som gavs ovan.

   (d) Använd programmet för att undersöka hur många gånger en viss tärningsvärde förekom i de 1000 kast med 3 tärningar du simulerade i uppgift 1. För att testa hur många gånger 6 förekom ska du använda

   `java E2 6 < three_dices`

   Då läser programmet in alla tal från filen *three_dices* istället för att läsa från terminalen.

3. I filen *E3.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
   Programmet ska användas utan kommandoradsargument. Användaren ska mata in några tal och programmet ska sedan räkna ut hur många gånger första talet förekommer totalt. Användaren bestämmer när inmatningen är över genom att använda Ctrl-D (i ett unix system) eller Ctrl-Z (i ett windows system). Här ser du några exempel av användning av programmet.

   ```
   java E3
   1 2 3 4 3 2 1 2 3 4
   Number 1 occurred 2 times.
   ```

   Programmet startas med `java E3`. Sedan matar användaren in några tal. Första talet är 1. Programmet skriver ut hur många gånger talet 1 förekommer. Programmets utskrift är då bara `Number 1 occurred 2 times.` (resten av det som ses i terminalen är bara ett eko av det användaren skriver som indata till programmet).

   ```
   java E3
   1
   Number 1 occurred 1 times.
   ```

   Programmet startas med `java E3`. Användaren matade in bara ett tal. Programmet skriver ut att det talet förekom bara en gång.
   Det du behöver göra är:

   (a) Spar filen *E3.java* i mappen där du ska arbeta. Kompilera programmet med kommandot

   `javac E3.java`

   Kör programmet med kommandot

   `java E3`

   Du borde inte se någon utskrift.

   (b) Lägg till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplet ovan.

   (c) Spara, kompilera och testa med samma exempel som gavs ovan.

   (d) (Inte obligatorisk) Om användaren trycker på Ctrl-D (eller Ctrl-Z) utan att ange några siffror har programmet inget att göra: det ska inte finnas någon utskrift och programmet borde avsluta tyst. Se till att ditt program har detta beteende.

4. I filen *E4.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
   Programmet används utan kommandoradsargument. Programmet läser in tal som användaren matar in eftersom och skriver ut de lästa talen utom de som upprepas i en följd. Användaren bestämmer när inmatningen är över genom att använda Ctrl-D (i ett unix system) eller Ctrl-Z (i ett windows system). Här ser du några exempel av användning av programmet.

   ```
   java E4
   1 2 2 1 5 1 1 7 7 7 7 1 1
   1 2 1 5 1 7 1 ^D
   ```

Programmet läser in talen från standard input ett i taget men bara efter att man trycker på Enter. Efter att ha läst varje tal avgör det on det ska skriva ut det lästa talet eller inte: om talet är samma som föregående skrivs det inte ut. Programmet avslutas när användaren gör Ctrl-D (eller Ctrl-Z).

```
java E4
1 2 2 1 5 1 1 7
1 2 1 5 1 7

1 2 2 1 5 1 1 7 7 7 7 1
1 2 1 5 1 7 1

2 2 1 5 1 1 7 7 7 7 1
2 1 5 1 7 1
^D
```

Det du behöver göra är:

(a) Spar filen *E4.java* i mappen där du ska arbeta. Kompilera programmet med kommandot

```
javac E4.java
```

Kör programmet med kommandot

```
java E4
```

Du borde inte se någon utskrift.

(b) Lägg till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplen ovan.

(c) Spara, kompilera och testa med samma exempel som gavs ovan.

(d) Om du tycker att det är svårt att se vad som händer då indata blandas med utdata i terminalen kan du dirigera om standard input till en fil och standard output till en annan. Till exempel kan du använda filen *dups* som finns med i zip-filen du laddade ner. Filen innehåller en rad med talen `1 2 2 3 4 5 4 4 4 3 2 1 1 1 0`. Om du kör ditt program enligt

```
java E4 < dups > dedups
```

skapas filen *dedups* med innehållet `1 2 3 4 5 4 3 2 1 0`.

5. I filen *E5.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
Programmet ska användas med ett kommandoradsargument som ska vara ett heltal. Detta talet anger hur många heltal som ska läsas in från standard input därefter. Programmet ska då beräkna medelvärdet av dessa tal samt hur många är större än medelvärdet. Här ser du några exempel av användning av programmet.

```
java E5 10
1 2 3 4 5 6 7 8 9 10
5 of 10 values were above the average 5.5
```

Kommandoradsargumentet är 10. Programmet läser in 10 heltal och skriver ut att 5 av dessa tal är större än medelvärdet 5.5 som den har beräknat. De stora talen är 6,7,8,9 och 10 som är större än medelvärdet.
Tänk på att de tio talen syns i terminalen men det är inget som programmet skriver ut, det är input från tangentbordet som ekas i terminalen! Det enda som programmet skriver ut är raden
`5 of 10 values were above the average 5.5`.

```
java E5 10
34 2 54 4 65 1 2 7 1 1
3 of 10 values were above the average 17.1
```

Här är de tre stora talen 34, 54 och 65 som är större än medelvärdet. Alla andra är mindre än 17.1.
Det du behöver göra är:

(a) Spara filen *E5.java* i mappen där du ska arbeta. Kompilera programmet med kommandot `javac E5.java` och kör med `java E5 10`. Du borde inte se någon utskrift.

(b) Lägg till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplen ovan.
Tänk på att medelvärdet inte ska vara ett heltal utan en `double`.
Formeln för medelvärdet av n värden $x_0, x_1, \ldots, x_{n-1}$ är $\frac{x_0+x_1+\ldots+x_{n-1}}{n}$.
Tänk på att använda ett fält (array) för att lagra värdena, eftersom programmet kommer att behöva gå igenom alla värden en gång till efter att ha beräknat medelvärdet och man kan inte läsa in de igen!

(c) Spara, kompilera och testa med samma exempel som gavs ovan.

6. I filen *E6.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
Programmet är tänkt att användas för att beräkna något för den simuleringen som görs med programmet `E1` från uppgift 1.
Programmet `E1` simulerade M försök av att kasta N tärningar och skrev ut M, N och sedan en rad per försök. Till exempel, om M var 4 och N var 2 så kunde utskriften se ut

```
4
2
1 3
1 6
2 2
6 5
```

Programmet `E6` som du ska skriva nu beräknar vilket är den mest vanliga summan av tärningarna i ett försök samt hur många gånger den summan förekom. I exemplet ovan har vi summorna

```
1 + 3
1 + 6
2 + 2
6 + 5
```

alltså `4, 7, 4 och 11`. Den mest vanlig är 4 och den förekommer två gånger.
Programmet ska läsa all indata från standard input: M, N och sedan M rader med N tal mellan 1 och 6 i varje rad. Här ser du några exempel:

```
java E6
4
2
1 3
1 6
2 2
6 5
The most frequent sum is 4
It occurrs 2 times.
```

```
java E6
5
4
1 1 1 1
1 2 3 4
5 1 1 1
2 2 1 1
1 1 1 1
The most frequent sum is 4
It occurrs 2 times.
```

Observera att det enda som programmet skriver ut är

```
The most frequent sum is 4
It occurrs 2 times.
```

Tanken är att den ska användas för långa experiment. Och då vill man dirigera om standard input. Här ser du hur man kan göra för att input kommer från från det som `E1` skriver ut när den används för att göra 1000 försök med tre tärningar:

```
java E1 1000 3 | java E6
The most frequent sum is 11
It occurrs 150 times.
```

Det du ska göra är att lägga till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplen ovan.
Tips Använd dig av ett fält där positionerna i fältet är de möjliga summorna som kan åstadkommas i ett försök. Till exempel, om försöken använder tre tärningar kan summorna bli högst 18 (alltså 6 * 3). Programmet kan använda ett fält med 19 platser (0 till 18). På varje plats kan man räkna hur många gånger summan (platsen) förekommer bland försöken.

7. I filen *E7.java* finns ett Java program, inklusive en kommentar på engelska som förklarar vad som behöver göras.
   Programmet ska användas med ett kommandoradsargument N som ska vara ett positivt heltal. Programmet ska skriva ut talen 0 till N-1, ett per rad. I varje rad ska programmet även skriva ett annat tal som tas från en så kallad permutation (en omorganisation) av dessa tal. Programmet beräknar även hur många element i permutationen inte byte plats.
   Här ser du några exempel av vad programmet ska göra

```
java E7 5
0 1
1 2
2 3
3 4
4 0
0 elements are in place
```

Programmet skriver ut talen 0 till 4 i två kolumner. Första kolumnen har alla tal 0 till 4 i ordning. I andra kolumnen förekommer samma tal men i en annan ordning (en permutation). I exemplet var ingen av värdena i permutationen på sin ursprungliga plats. Därför utskriften `0 elements are in place` på slutet.

```
java E7 5
0 0
1 4
2 2
3 3
4 1
3 elements are in place
```

I denna körningen förblev 0, 2 och 3 i permutationen på sina platser, därför utskriften `3 elements are in place` på slutet.
Det du ska göra är att lägga till den kod som behövs för att åstadkomma det som programmet ska göra enligt beskrivningen och exemplen ovan.
I filen du får finns redan en beräkningen av permutationen av talen 0 till N-1. Värdena finns i ett fält som hette `permutation`. Din kod ska komma efter det som redan finns i programmet.

Inlämningsuppgifter - del 2

I filen *Bonus.java* finns ett Java program. Spara i rätt mapp och kompilera. Uppgiften kommer från kursboken: se uppgift 24 under rubriken **Web Exercises** i introcs.cs.princeton.edu/java/14array/.

Skriv ett program som

- läser två heltal M och N från kommandoraden,
- beräknar M slump permutationer av talen 0 till N-1 och
- skriver ut
  - permutationerna,
  - antal left-to-right-minima för varje permutation och
  - medelvärdet av dessa antal

Här är förklaring av left-to-right-minima:

Antalet left-to-right-minima for en följd av tal är hur många gånger man ser minsta talet hittills. Till exempel, om permutationen av talen 0 till 9 är

```
4 8 2 1 9 0 5 3 7 6
```

då är antalet left-to-right minima 4 eftersom:

- Först är 4 det minsta talet vi ser (första elementet är alltid minsta talet vi har sett hittills i början).
- Sedan är 2 minsta talet vi ser.
- Sedan är 1 minsta talet vi ser.
- Sedan är 0 minsta talet vi ser.

Så, 4 är hur många gånger ett tal är det minsta talet vi har sett hittills: antalet left-to-right-minima.

Ytterligare ett exempel. Om permutationen av talen 0 till 9 är

```
6 9 7 5 0 2 8 4 1 3
```

är antalet left-to-right-minima 3 eftersom vi först ser 6, sedan 5 och sedan 0 som minsta hittills.

Nu till programmet som ska skrivas och hur det ska testas. Programmet ska läsa in två kommandoradsargument: M (antalet permutationer som ska beräknas) och N (permutationerna ska vara av talen 0 till N-1). Här följer tre exempel:

```
java Bonus 5 10
5 0 2 1 8 4 6 3 9 7 (2)
4 2 3 7 8 6 9 1 0 5 (4)
9 1 8 3 0 7 2 4 6 5 (3)
0 7 2 5 1 9 6 3 4 8 (1)
0 7 6 4 3 8 9 2 5 1 (1)
```

Programmet beräknade och skrev ut permutationen

```
5 0 2 1 8 4 6 3 9 7
```

och beräknade antalet left-to-right-minima som är 2 och som skrevs ut bredvid permutationen inom parentes. Programmet beräknade fyra till permutationer, skrev ut de med respektive left-to-right-minima inom parentes. Slutligen skrev programmet ut medelvärdet `(2 + 4 + 3 + 1 +1) / 5` som är den `2.2` som syns.

Ett till exempel:

```
java Bonus 5 10
9 0 2 8 3 6 5 1 4 7 (2)
5 9 0 3 7 4 8 6 1 2 (2)
1 2 4 5 7 6 0 8 9 3 (2)
8 6 1 5 0 4 2 7 9 3 (4)
1 3 6 8 9 7 2 0 4 5 (2)

2.4
```

Och ett till:

```
java Bonus 10 5
1 2 3 0 4 (2)
4 1 3 0 2 (3)
1 0 4 2 3 (2)
0 2 1 3 4 (1)
0 3 2 1 4 (1)
3 1 4 0 2 (3)
3 1 2 0 4 (3)
0 4 2 1 3 (1)
0 4 1 3 2 (1)
2 0 4 1 3 (2)

1.9
```

Tips 1 Koden för att skapa en permutation finns i programmet `E7`.
Tips 2 Skriv först den delen av programmet som skapar en enda permutation, räknar antalet left-to-right minima för permutationen och skriver ut permutationen och detta tal inom parentes.

Uppgifter från gamla tentor som kan lösas efter vecka 3.

De uppgifter som använder fält i tentorna använder även funktioner som vi kommer att se nästa vecka. Så denna vecka finns inga tenta uppgifter.