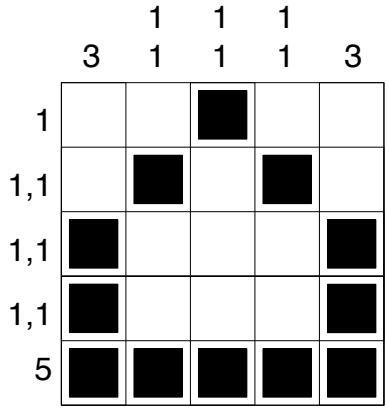
Programmeren 1 29 augustus 2012 Prof. T. Schrijvers

Instructies

Schrijf al je antwoorden op deze vragenbladen (op de plaats die daarvoor is voorzien). Geef ook je kladbladen af. Bij heel wat vragen moet je zelf Java-code schrijven. Hou dit *kort en bondig*. Je hoeft geen commentaarregels te schrijven en ook eventuele `import`-opdrachten zijn niet nodig.
Pas op voor de addertjes onder het gras.
Veel succes!

Vraag 1: Nonogrammen (7pt)



Een **nonogram** is een Japanse beeldpuzzel die bestaat uit een leeg diagram met getallen boven en links van het diagram. Elk getal staat voor een sequentie van één of meer aaneengesloten zwarte vakjes in de betreffende rij of kolom. Opeenvolgende sequenties zijn gescheiden door één of meer aaneengesloten witte vakjes. Door de vakjes op de juiste manier zwart en wit in te kleuren, vormt zich een afbeelding in het diagram: de oplossing van de puzzel. Hiernaast zie je zo'n opgeloste puzzel.

In deze vraag gaan we geen puzzels oplossen, maar controleren of een gegeven *oplossing* correct is ten opzichte van een gegeven *opgave*. We stellen een oplossing voor door een geneste $n \times m$ array van `boolean`s: een zwart vakje is `true`, en een wit vakje is `false`. Een opgave voor een lijn, de lengtes van de zwarte sequenties, stellen we voor door een `int[]`. Bijvoorbeeld, de opgave "1,1" stellen we voor als `new int[]{1,1}`. De opgave voor alle lijnen in een richting (bv. alle rijen) stellen we voor als een `int[][]`.

Omdat er geen essentieel verschil is tussen het controleren van een rij of een kolom uit de oplossing, gaan we abstractie maken van de volgorde van doorlopen van de geneste array van booleans die de oplossing voorstelt. Abstractie maken we met behulp van de volgende interface, die geïnspireerd is op de gekende Java-iteratoren.

```
public interface ArrayIterator {

  public boolean hasNext();

  public boolean next();

  public boolean hasNextLine();

  public void nextLine();

}
```

Deze `ArrayIterator` laat toe om een geneste array van `booleans` *lijn per lijn* te doorlopen. Hierbij kan een lijn ofwel een rij ofwel een kolom zijn. Je gebruikt de interface als volgt:

- Alvorens aan een nieuwe lijn te beginnen, moet je de controleren of er een volgende lijn is met behulp van de methode `hasNextLine()`. Als er een volgende lijn is, laat je vervolgens de `ArrayIterator` overgaan naar die lijn met `nextLine()`.
- Als de `ArrayIterator` zich op een lijn bevindt, kan je vragen of er een volgend element is op de lijn met `hasNext()`. Dat volgende element, een `boolean` die zwart of wit voorstelt, kan je vervolgens opvragen met `next()`.

Opgave

1. Maak een klasse `RijIterator` die de interface `ArrayIterator` implementeert. Ze doorloopt een geneste array rij per rij. Voeg ook een constructor toe die een geneste `boolean` array als parameter heeft, en voorzie de nodige velden.
2. Maak een klasse `KolomIterator` die de interface `ArrayIterator` implementeert. Ze doorloopt een geneste array kolom per kolom. Voeg ook een constructor toe die een geneste `boolean` array als parameter heeft, en voorzie de nodige velden.
3. Maak een klasse `NonogramChecker`.
   - Implementeer een methode die nagaat of een ingevulde `puzzel` voldoet aan de opgegeven sequenties voor `rijen` en `kolommen`:

     ```
     public static boolean isOplossing(boolean[][] puzzel,
                                       int[][] rijen, int[][] kolommen)
     ```

   - Maak een hulpmethode `checkLijnen(ArrayIterator it, int[][] lijnen)` die je vanuit `isOplossing` oproept om zowel de rijen als de kolommen te controleren. Deze hulpmethode controleert alle lijnen (die horizontaal of verticaal lopen) overeenkomstig de gegeven `ArrayIterator`.
   - Maak een hulpmethode `checkLijn(ArrayIterator it, int[] lijn)` die de huidige lijn van de gegeven `ArrayIterator` controleert. Roep deze hulpmethode op vanuit `checkLijnen`.

     Tip: Je houdt best bij in een lokale variabele of je in een lopende sequentie van zwarte vakjes zit of niet. Dan kan je op basis van de kleur van het volgende vakje beslissen wat je moet doen.

...

Vraag 2: Methode-oproepen **(2pt)**

```
public class A {
  public static int m() {
    boolean b = (1 == 2);
    if (b = true) {
      return 42;
    } else {
      return -1;
    }
  }
}
```

```
public abstract class B {
  public B() {}
  public int m() {
    return n();
  }
  public int n() {
    return m() - 2;
  }
}
```

```
public class C {
  private int[] items;
  public C() {
    int[] items = new int[7];
  }
  public int first() {
    return items[0];
  }
}
```

```
public class D extends B {
  public D() {}
  public int m() {
    return 5 % 3;
  }
}
```

Gegeven bovenstaande klassendefinities, geef aan wat het resultaat is van onderstaande expressies:

(a)	`A.m()`	
(b)	`new B().n()`	
(c)	`new C().first()`	
(d)	`new D().n()`	

Vraag 3: Exceptions **(1pt)**

```
public void m(int x) throws XXX {
  if (x % 2 == 0) {
    throw new IOException(something is wrong");
  } else {
    throw new IllegalArgumentException("something else is wrong");
  }
}
public int n(int x) {
  try {
    x-=1;
    m(x);
  } catch (IOException e) {
    x += 2;
  } catch (Exception e) {
    x *= x;
  }
  return x;
}
```

(a)	Wat is de meest specifieke klassenaam waarmee je `XXX` kan vervangen?	
(b)	Wat is het resultaat van `n(12)`?	

Vraag 4: Afbeeldingen Verkleinen **(2,5pt)**

Schrijf een methode met signatuur

```
public int[][] verklein(int[][] afbeelding)
```

De methode neemt een $2n \times 2m$ matrix `afbeelding` voorgesteld door een geneste array, en geeft een $n \times m$ matrix terug.

De parameter stelt een grote zwart-wit afbeelding voor; elk element in de matrix stelt de grijswaarde (in het bereik 0-255) van een beeldpunt voor. De uitvoer is een verkleinde afbeelding die bekomen wordt door 4 naburige beeldpunten te reduceren tot 1 beeldpunt. De grijswaarde van dat gereduceerde beeldpunt is het gemiddelde van de grijswaarden van de oorspronkelijke beeldpunten. Hier zie je het effect voor een 4×6 matrix. Je code moet werken voor alle mogelijke waarden van n en m.

$$\left[\begin{array}{cc|cc} a_{1,1} & a_{2,1} & a_{3,1} & a_{4,1} \\ a_{1,2} & a_{2,2} & a_{3,2} & a_{4,2} \\ \hline a_{1,3} & a_{2,3} & a_{3,3} & a_{4,3} \\ a_{1,4} & a_{2,4} & a_{3,4} & a_{4,4} \\ \hline a_{1,5} & a_{2,5} & a_{3,5} & a_{4,5} \\ a_{1,6} & a_{2,6} & a_{3,6} & a_{4,6} \end{array}\right] \Longrightarrow \left[\begin{array}{cc} \frac{(a_{1,1}+a_{2,1}+a_{1,2}+a_{2,2})}{4} & \frac{(a_{3,1}+a_{4,1}+a_{3,2}+a_{4,2})}{4} \\ \frac{(a_{1,3}+a_{2,3}+a_{1,4}+a_{2,4})}{4} & \frac{(a_{3,3}+a_{4,3}+a_{4,3}+a_{4,4})}{4} \\ \frac{(a_{1,5}+a_{2,5}+a_{1,6}+a_{2,6})}{4} & \frac{(a_{3,5}+a_{4,5}+a_{4,5}+a_{4,6})}{4} \end{array}\right]$$

Vraag 5: Arrays **(1pt)**

Hoeveel verschillende `int` waarden passen in geneste array `arr` na uitvoering van de code?

(a)	`int[][] arr = new int[3][7];`	

(b)	`int[][] arr = new int[3][];`
	`arr[0] = new int[5];`
	`for (int i = 1; i < arr.length; i++) {`
	`  arr[i] = arr[i-1];`
	`}`	

Vraag 6: Herschrijven **(3pt)**

Herschrijf onderstaande methodes zodat: (a) lijsten ipv. arrays gebruikt, (b) een iterator ipv. for-each lus gebruikt, en (c) een while lus ipv. een for lus gebruikt.

(a)
```
public void m1(int[] k, int[] l) {
  for (int i = 0; i < k.length; i++) {
    k[i] = l[k.length - i];
  }
}
```

(b)
```
public void m2(Set <String> set) {
  for (String t: set) {
    System.out.println(t);
  }
}
```

(c)
```
public void m3(int[] a) {
  for (int i = 1; i < 10; i++) {
    a[i] = i * a[i-1];
  }
}
```

Vraag 7: Examen opstellen **(3,5pt)**

Het bestand `vragenlijst.txt` bestaat uit tientallen mogelijke examenvragen; één op elke regel. Genereer een bestand `examen.txt` dat 7 willekeurig gekozen vragen bevat; ook één op elke regel. De gekozen vragen moeten onderling verschillend zijn.

Schrijf hiervoor een stuk code, zonder methode-hoofding. Vang mogelijke checked exceptions op en druk een foutboodschap af.

Einde