Programmeren 1 22 januari 2014 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: Candy Crush Light **(7pt)**

In deze examenvraag gaan we een light-versie van het populaire spel Candy Crush maken. Het spel wordt gespeeld met een rechthoekig rooster van $n \times m$ vakjes. In elk vakje bevindt zich een snoepje met een bepaalde kleur en type.

In deze versie van het spel kan een speler twee willekeurige snoepjes van plaats verwisselen. Nadat de twee snoepjes van plaats verwisseld zijn, gaan we na of ze zich in een horizontale aaneengesloten reeks van drie of meer horizontale snoepjes met dezelfde kleur bevinden.

Als dat voor minstens één van beide het geval is, dan worden de één of twee aaneengesloten reeksen van het rooster verwijderd. Vervolgens schuiven de snoepjes uit alle bovenliggende rijen een plaats naar onder op om de vrijgekomen plaatsen op te vullen. Tot slot worden de vrijgekomen plaatsen bovenaan het rooster opgevuld met willekeurige nieuwe snoepjes.

Lees onderstaande opgave volledig: je moet het geheel overzien om de rol van de onderdelen te begrijpen.

1. Implementeer de klasse `Rooster` die een rooster voorstelt.

   - Voorzie een constructor om een nieuw rooster van gegeven afmetingen te maken. Vul het nieuwe rooster op met willekeurige snoepjes. Gebruik hiervoor de volgende hulpmethode.
   - De methode

     ```
     private void plaatsWillekeurigSnoepje(int r, int k)
     ```

     plaatst een willekeurig nieuw snoepje (rood of blauw, gewoon snoepje of ijssnoepje) op de gegeven plaats (rij `r`, kolom k) in het rooster.
     BONUS: Je moet deze methode niet implementeren; je mag ze als gegeven beschouwen. Implementeer je ze wel, dan kan je 1 extra punt verdienen. (Je totale score kan uiteraard nooit groter dan 20/20 worden.)
   - Voorzie de methode

     ```
     public void verwissel(int r1, int k1, int r2, int k2)
     ```

     die de snoepjes op de gegeven posities van plaats verwisselt volgens de regels van het spel. Verwissel dus de snoepjes van plaats. Ga na met de methode `vindReeks` van de klasse `Snoepje` of dit resulteert in te verwijderen snoepjes. Indien ja, verwijder dan de nodige snoepjes, schuif de snoepjes uit alle bovenliggende rijen een plaats op naar onder, en voeg bovenaan willekeurige nieuwe snoepjes toe.

- Voorzie gepaste velden en hulpmethodes waar nodig.

2. Implementeer de klasse `Snoepje` die een snoepje voorstelt.

   - Voorzie een constructor waaraan het rooster, de plaats op het rooster en de kleur (in de vorm van een String) kunnen meegegeven worden.
   - Voorzie gepaste velden voor de klasse en eventuele `set-` en `get-`methodes.
   - Voorzie de methode

     ```
     public boolean zelfdeKleur(Snoepje ander)
     ```

     die aangeeft of het andere snoepje dezelfde kleur heeft.
   - Voorzie de methode

     ```
     public void vindReeks(Set<Snoepje> set)
     ```

     die de de langst mogelijke horizontale aaneengesloten reeks van snoepjes rondom het huidige snoepje bepaalt zodat de snoepjes in de reeks allemaal dezelfde kleur hebben. Als de reeks uit minstens 3 snoepjes bestaat, dan worden alle snoepjes uit de reeks toegevoegd aan de gegeven verzameling. Doe hiervoor een beroep op de volgende hulpmethode.
   - Voorzie de hulpmethode

     ```
     protected void verwijder(Set<Snoepje> set)
     ```

     die het snoepje opgeeft als te verwijderen door het toe te voegen aan de gegeven verzameling.

3. Implementeer de subklasse `IJsSnoepje`. Een ijssnoepje verdwijnt pas effectief de tweede keer dat het verwijderd wordt. (De eerste keer smelt het alleen.)

   Voorzie zelf de nodige constructoren en velden.

Let op: implementeer niet meer dan wat hier gevraagd wordt – dit is geen volledige versie van Candy Crush.

```
public class Rooster
{
    private Snoepje[][] grid;

    public Rooster(int n, int m) {
        grid = new Snoepje[n][m];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                plaatsWillekeurigSnoepje(i,j);
            }
        }
    }

    public void verwissel(int x1, int y1, int x2, int y2) {
        Snoepje c1 = grid[x1][y1];
        Snoepje c2 = grid[x2][y2];

        c1.verplaats(x2,y2);
        c2.verplaats(x1,y1);
        Set<Snoepje> todo = new HashSet<Snoepje>();
        c1.vindReeks(todo);
        c2.vindReeks(todo);
        if (!todo.isEmpty()) {
            for (Snoepje c : todo) {
                clearSnoepje(c);
            }
        }
    }

    public void set(int x, int y, Snoepje candy) {
        grid[x][y] = candy;
    }

    public Snoepje get(int x, int y) {
        if (x >= 0 && x < grid.length
            && y >= 0 && y < grid[x].length) {
                return grid[x][y];
            } else {
                return null;
            }
    }

    private void clearSnoepje(Snoepje c) {
        int x = c.getX();
        int y = c.getY();
        for (int y1 = y - 1; y1 > 0; y1--) {
            grid[x][y1].verplaats(x,y1+1);
        }
        plaatsWillekeurigSnoepje(x,0);
    }
```

```
    private void plaatsWillekeurigSnoepje(int x, int y) {
        if (new Random().nextBoolean()) {
            new Snoepje(this,x,y,randomColor());
        } else {
            new SpecialSnoepje(this,x,y,randomColor());
        }
    }

    private String randomColor() {
        if (new Random().nextBoolean()) {
            return ``red'';
        } else {
            return ``blue'';
        }
    }

}

// ------------------------------------------------- //

public class Snoepje
{
    private Rooster grid;
    private int x;
    private int y;
    private String color;

    public Snoepje(Rooster grid, int x, int y, String color) {
        this.grid = grid;
        this.x = x;
        this.y = y;
        grid.set(x,y,this);
        this.color = color;
    }

    protected void verwijder(Set<Snoepje> set) {
        set.add(this);
    }

    public void verplaats(int x, int y) {
        this.x = x;
        this.y = y;
        grid.set(x,y,this);
    }

    public void vindReeks(Set<Snoepje> set) {
        int upper = x+1;
        while (zelfdeKleur(grid.get(upper,y))) {
            upper++;
        }
        int lower = x-1;
```

```
        while (zelfdeKleur(grid.get(lower,y))) {
            lower--;
        }

        if (upper - lower - 1 > 2) {
            for (int x1 = lower + 1; x1 < upper; x1++) {
                grid.get(x1,y).verwijder(set);
            }
        }
    }

    public boolean zelfdeKleur(Snoepje other) {
        return other != null && this.color.equals(other.color);
    }

    protected int getX() {
        return x;
    }

    protected int getY() {
        return y;
    }

    protected Rooster getRooster() {
        return grid;
    }

}

// ---------------------------------------------------- //

public class IJsSnoepje extends Snoepje
{
   private boolean gesmolten;

    public IJsSnoepje(Rooster grid, int x, int y, String color) {
        super(grid,x,y,color);
        gesmolten = false;
    }

    protected void verwijder(Set<Snoepje> set) {
        if (gesmolten) {
            super.verwijder(set);
        } else {
            gesmolten = true;
        }
    }

}
```

Vraag 2: Methode-oproepen (2pt)

```
public abstract class A {
  public A() {}
  public int n() {
    return m() * 2;
  }
  public int m() {
    return 3;
  }
}
```

```
public class B extends A {
  public B() {}
  public int m() {
    return n() * 2;
  }
}
```

```
public class C extends B {
  public C() {}
  public int n() {
    return 7;
  }
}
```

```
public class D {
  private List<Object> list;
  public D() {
    List<Object> list = new ArrayList<Object>();
  }
  public int size() {
    return list.size();
  }
}
```

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

(a)	`new A().m()`	**compilatiefout: instantiatie abstracte klasse**
(b)	`new B().m()`	**oneindige lus**
(c)	`new C().m()`	**14 = 7 * 2**
(d)	`new D().size()`	**NullPointerException treedt op**

Vraag 3: Exceptions (1pt)

```
public void m(Object o) throws Exception {
  if (o.equals(null)) {
    throw new IOException("oh no, something is terribly wrong");
  } else {
    throw new IllegalArgumentException("bad argument");
  }
}
public void n(Object o) throws Exception {
  try {
    m(o);
    System.out.println(1);
  } catch (RuntimeException e) {
    System.out.println(2);
  } catch (Exception e) {
    System.out.println(3);
  }
}
```

Wat wordt afgedrukt bij volgende oproepen?

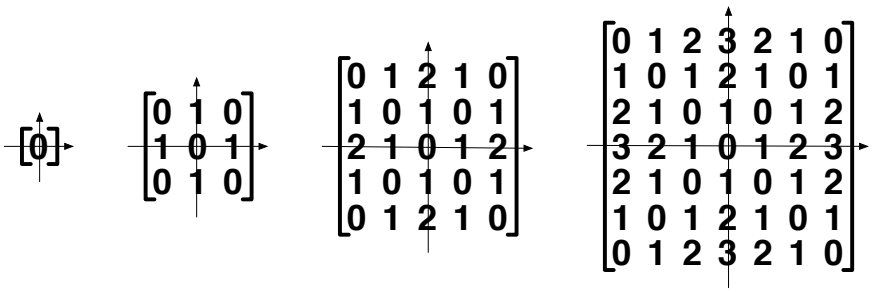
(a)	`n(new Object())`	**2**
(b)	`n(null)`	**2 (NullPointerException treedt op)**

Vraag 4: Patroon **(3pt)**

Schrijf een methode met signatuur

```
public int[][] patroon(int n)
```

De methode geeft een geneste array terug die een $(2\mathtt{n}+1) \times (2\mathtt{n}+1)$ matrix voorstelt. De matrix is opgevuld met getallen van 0 tot $\mathtt{n}-1$. Hieronder zie je het resultaat voor $n = 0$, $n = 1$, $n = 2$ en $n = 3$. Veralgemeen het patroon schrijf code die werkt voor alle $n \geq 0$.




```
public int[][] patroon(int n) {
    int size = 2 * n + 1;
    int[][] result = new int[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[i][j] = Math.abs(Math.abs(i - n) - Math.abs(j - n));
        }
    }
    return result;
}
```

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

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

(a)	`int[][] arr = new int[1][1];`	**1 = 1 * 1**

(b)	`int[][] arr = new int[5][];`
	`for (int i = 0; i < arr.length; i++) {`
	`  arr[i] = new int[i % 3];`
	`}`
		4 = 0 + 1 + 2 + 0 + 1

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

Herschrijf onderstaande methodes zodat: (a) lijsten i.p.v. arrays gebruikt, (b) een iterator i.p.v. een for-each lus gebruikt, en (c) een while-lus met index i.p.v. for-each lus gebruikt.

(a)
```
public void m1(int[] a, int[] b) {
  for (int i = 0; i < a.length; i+=2) {
    a[i] = b[i];
  }
}
```
```
public void m1(List<Integer> a, List<Integer> b) {
  for (int i = 0; i < a.size(); i+=2) {
    a.set(i,b.get(i));
  }
}
```

(b)
```
public int m2(Set<String> set) {
  int result = 0;
  for (String el: set) {
    result += el.length();
  }
  return result;
}
```
```
public int m2(Set<String> set) {
  int result = 0;
  Iterator<String> it = set.iterator();
  while (it.hasNext()) {
    result += it.next().length();
  }
  return result;
}
```

(c)
```
public String m3(ArrayList<String> l) {
  String result = "";
  for (String s : l) {
    result += s;
  }
  return result;
}
```
```
public String m3(ArrayList<String> l) {
  String result = "";
  while (i < l.size()) {
    result += l.get(i);
    i++;
  };
  return result;
}
```

Vraag 7: Eindscore **(3pt)**

De evaluatie van het vak *Functionele en Logische Programmeertalen* bestaat uit twee projecten en een examen. Elk onderdeel wordt gequoteerd op 10 punten. De eindscore staat op 20 punten; de projecten tellen daarin elk mee voor 25% en het examen voor 50%. Alle scores zijn positieve gehele getallen; afrondingen gebeuren naar onder.

Schrijf een methode met de volgende signatuur die de deelscores uitleest uit het `invoerbestand` en de eindscore wegschrijft naar het `uitvoerbestand`:

```
public void berekenEindscore(String invoerbestand, String uitvoerbestand)
```

Het invoerbestand bestaat uit een aantal regels met daarop de loginnaam van een student, gevolgd door de drie scores (respectievelijk project 1, project 2 en examen). Bijvoorbeeld:

```
ajanssen 7 8 5
ppeeters 4 6 9
```

Het uitvoerbestand bevat een aantal regels met daarop telkens de loginnaam gevolgd door de eindscore. Bijvoorbeeld:

```
ajanssen 12
ppeeters 14
```

```
public void berekenEindscore(String in, String uit) {
    try {
        Scanner s = new Scanner(new File(in));
        PrintWriter w = new PrintWriter(new FileWriter(uit));
        while (s.hasNext()) {
            String login = s.next();
            int p1 = s.nextInt();
            int p2 = s.nextInt();
            int e  = s.nextInt();
            int score = (p1 + p2 + 2 * e) / 2;
            w.println(login + " " + score);
        }
        w.close();
        s.close();
    } catch (IOException e) {
    }
}
```

Einde