Programmeren 1 20 januari 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: Cryptografie **(7pt)**

Cryptografie of geheimschrift houdt zich bezig met het versleutelen van informatie zodat het makkelijk is voor rechtmatige personen om de informatie te reconstrueren, maar moeilijk (liefst praktisch onmogelijk) voor onrechtmatige personen.

Er zijn heel wat technieken:

1. Een klassieker is het *Caesarcijfer*, ook wel "Rot-3" genoemd, waarbij elke letter uit het alfabet 3 plaatsen naar rechts wordt geroteerd bij het vercijferen, en 3 plaatsen naar links bij het ontcijferen. Bijvoorbeeld `ABC` wordt `DEF`. Op het einde van het alfabet roteren we naar het begin van het alfabet. Zo wordt `XYZ` vercijferd naar `ABC`.

   Een veralgemening van Rot-3 is Rot-n voor een gegeven $n > 0$.

2. De letters van een *sleutelwoord* worden gecombineerd met de overeenkomstige letters van de te vercijferen boodschap. Zo kunnen we de letters optellen modulo 26 om te vercijferen, en aftrekken modulo 26 om te ontcijferen. Als de boodschap langer is dan de sleutel, dan herhalen we de sleutel. Bijvoorbeeld met sleutel `JAVA` versleutelen we `ABCDEFGHIJ` naar `KCYEOGCISK`.

   ```
     JAVAJAVAJA
     ABCDEFGHIJ
   + ----------
     KCYEOGCISK
   ```

3. Een manier om een complexe vercijfertechniek te bekomen is om twee eenvoudigere technieken samen te stellen door ze na elkaar toe te passen. Zo kan je bijvoorbeeld eerst een boodschap roteren alvorens er een sleutel bij op te tellen. Bij het ontcijferen moet je dan in omgekeerde volgorde werken.

We hebben de technieken hierboven geïllustreerd op informatie bestaande uit 26 letters. In het digitale tijdperk werken we echter met bytes, getallen in het bereik 0 - 255. In deze vraag werken we verder met dat grotere bereik, dat we voor de eenvoud voorstellen als `int`s in Java.

Opgave

1. Implementeer de abstracte klasse `Cijfer` die een vercijfertechniek voorstelt.

   - Voorzie twee abstracte methodes

     ```
     public abstract int vercijfer(int)

     public abstract int ontcijfer(int)
     ```

     die een 'letter' uit de boodschap vercijferen of ontcijferen.
   - Voorzie een concrete methode

     ```
     public int[] vercijfer(int[])
     ```

     die een hele reeks van 'letters' ineens vercijfert.
   - Voorzie een concrete methode

     ```
     public void reset()
     ```

     die het `Cijfer` klaarmaakt om een nieuwe boodschap te verwerken. Voorzie een lege implementatie die je op gepaste wijze overschrijft waar nodig in de subklassen.

2. Implementeer de subklasse `RotatieCijfer` voor de Rot-n techniek. Zorg ook dat n gekozen kan worden, en voorzie een gepaste constructor.

3. Implementeer de subklasse `SleutelCijfer`. Zorg ook dat het sleutelwoord kan opgegeven worden als een `int[]`, en voorzie een gepaste constructor.

4. Implementeer de subklasse `SamengesteldCijfer`. Een samengesteld cijfer stelt de compositie van twee andere cijfers voor. Het past de twee gegeven cijfers na elkaar toe om te vercijferen en ontcijferen. Voorzie een gepaste constructor.

5. Zorg dat grote getallen (zoals 256) maar op 1 plaats in je oplossing letterlijk voorkomen.

```
public abstract class Cijfer {

  protected static final int MAX = 256;

  public abstract int vercijfer(int l);

  public abstract int ontcijfer(int l);

  public int[] vercijfer(int[] ls) {
    int[] resultaat = new int[ls.length];
    for (int i = 0; i < resultaat.length; i++) {
      resultaat[i] = vercijfer(ls[i]);
    }
    return resultaat;
  }

  public void reset() {}

}


public class RotatieCijfer extends Cijfer {

  private final int rot;

  public RotatieCijfer(int n) {
    rot =  n;
  }

  public int vercijfer(int l) {
    return (l + rot) % MAX;
  }

  public int ontcijfer(int l) {
    return (l + MAX - rot) % MAX;
  }

}


public class SleutelCijfer extends Cijfer {

  private final int[] sleutel;
  private int i;

  public SleutelCijfer(int[] sleutel) {
    this.sleutel = sleutel;
    i = 0;
```

```
  }

  public int vercijfer(int l) {
    int resultaat = (l + sleutel[i]) % MAX;
    moveRight();
    return resultaat;
  }

  public int ontcijfer(int l) {
    int resultaat = (l + MAX - sleutel[i]) % MAX;
    moveRight();
    return resultaat;
  }

  private void moveRight() {
    i = (i + 1) % sleutel.length;
  }

  public void reset() {
    i = 0;
  }

}


public class SamengesteldCijfer extends Cijfer {

  private final Cijfer cijfer1, cijfer2;

  public SamengesteldCijfer(Cijfer c1, Cijfer c2) {
    cijfer1 = c1;
    cijfer2 = c2;
  }

  public int vercijfer(int l) {
    return cijfer2.vercijfer(cijfer1.vercijfer(l));
  }

  public int ontcijfer(int l) {
    return cijfer1.ontcijfer(cijfer2.ontcijfer(l));
  }

  public void reset() {
    cijfer1.reset();
    cijfer2.reset();
  }

}
```

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

```
public class A {
  private int[] items;
  public A() {
    int[] items = new int[7];
  }
  public int size() {
    return items.length;
  }
}
```

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

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

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

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

(a)	`new A().size()`	**NullPointerException** (ongeinitialiseerd veld)
(b)	`new B().n()`	**compiler error** (abstracte klasse intantieren)
(c)	`C.m()`	**-1**
(d)	`new D().n()`	**6**

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

```
public int m(int x) throws ??? {
  throw new ???("something is wrong");
}
public int n() {
  int x = 4;
  try {
    x = 3 + m(x);
  } catch (IOException e) {
    x += 2;
  } catch (Exception e) {
    x *= x; ;
  }
  return x;
}
```

Wat is het resultaat van de oproep `n()` als we `???` vervangen door een van onderstaande woorden:

(a)	`NullPointerException`	**16**
(b)	`IOException`	**6**

Vraag 4: Diagonaalproduct **(2,5pt)**

Schrijf een methode met signatuur

```
public int[] diagonaalProduct(int[][] a, int[][] b)
```

De methode neemt twee $n \times n$ matrices `a` en `b`, voorgesteld door geneste arrays, en geeft hun "diagonaalproduct" $a \otimes b$ terug. Dit is een vector (array) van de scalaire producten van de overeenkomstige diagonalen in de twee matrices. Bij de geneste arrays volgen we de conventie dat de binnenste arrays de (horizontale) rijen van een matrix voorstellen. Je code moet werken voor elke $n \geq 1$. Hieronder zie je het resultaat voor $n = 2$ en $n = 3$.

$$\begin{bmatrix} a_{1,1} & a_{2,1} \\ a_{1,2} & a_{2,2} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{2,1} \\ b_{1,2} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,2} * b_{1,2} \\ a_{1,1} * b_{1,1} + a_{2,2} * b_{2,2} \\ a_{2,1} * b_{2,1} \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{2,1} & b_{3,1} \\ b_{1,2} & b_{2,2} & b_{3,2} \\ b_{1,3} & b_{2,3} & b_{3,3} \end{bmatrix} = \begin{bmatrix} a_{1,3} * b_{1,3} \\ a_{1,2} * b_{1,2} + a_{2,3} * b_{2,3} \\ a_{1,1} * b_{1,1} + a_{2,2} * b_{2,2} + a_{3,3} * b_{3,3} \\ a_{2,1} * b_{2,1} + a_{3,2} * b_{3,2} \\ a_{3,1} * b_{3,1} \end{bmatrix}$$

Pas op: er zijn meerdere mogelijke oplossingen, de ene al makkelijker dan de andere.

```
public int[] diagonaalProduct(int[][] a, int[][] b) {
  int n = a.length;
  int[] resultaat = new int[2 * n - 1];
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      resultaat[j-i+n-1] += a[i][j] * b[i][j];
    }
  }
  return resultaat;
}
```

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

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

(a)	`int[][] arr = new int[2][0];`	**0**

(b)	`int[] rij = new int[3];`
	`int[][] arr = new int[2][];`
	`for (int i = arr.length-1; i > -1; i--) {`
	`  arr[i] = rij;`
	`}`
		3

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

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

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

(b)
```
public void m2(Set<String> set) {
  Iterator<String> it = set.iterator();
  while (it.hasNext()) {
    System.out.println(it.next());
  }
}
⇓
public void m2(Set<String> set) {
  for (String text: set) {
    System.out.println(text);
  }
}
```

(c)
```
public void m3(int[] a) {
  for (int i = 1; i < 10; i++) {
    a[i] = i * a[i-1];
  }
}
⇓
public void m3(Map<Integer,Integer> a) {
  for (int i = 1; i < 10; i++) {
    map.put(i, i * a.get(i-1);
  }
}
```

Vraag 7: Belspelletje **(3,5pt)**

Help Gaëtan om belspelletjes te winnen. Schrijf een methode `sleutel` met drie parameters (een string en twee mappen) die een geheel getal teruggeeft. De string-parameter `tekst` bestaat uit een aantal woorden, gescheiden door spaties. De eerste map-parameter beeldt letters af op gehele getallen en de tweede map beeldt woorden af op gehele getallen.

De vergezochte[1] sleutel bestaat erin om, overeenkomstig de twee mappen, de waarde van elke letter en elk woord uit `tekst` op te tellen. Als een woord meermaals voorkomt, dan tel je het maar 1 keer mee. Als een letter meermaals voorkomt, dan tel je alle voorkomens mee. Letters en woorden die niet in de mappen voorkomen hebben waarde 0.
Bijvoorbeeld met de mappen $\{$`'i'` $\mapsto 1$, `'j'` $\mapsto 1$, `'v'` $\mapsto 5$, `'x'` $\mapsto 10$, `'l'` $\mapsto 50$, `'c'` $\mapsto 100\}$ en $\{$`"een"` $\mapsto 1$, `"miljoen"` $\mapsto 1000000\}$, levert de tekst

`"een miljoen en een miljoen is veel poen"`

de waarde **1000161** op.

```
  'i' * 3                3
  'j' * 2                2
  'v' * 1                5
  'l' * 3              150
  "een"                  1
  "miljoen"        1000000
+ ----------------------
                   1000161
```

```
public int sleutel(String tekst, Map<Character,Integer> map1
                                      , Map<String,Integer> map2) {
   int resultaat = 0;
   for (int i = 0; i < tekst.length(); i++) {
     resultaat += coerce(map1.get(tekst.charAt(i)));
   }
   String[] woorden = tekst.split(" ");
   Set<String> s = new HashSet<String>();
   for (String w: woorden) {
     s.add(w);
   }
   for (String w: s) {
     resultaat += coerce(map2.get(w));
   }
   return resultaat;
}

private int coerce(Integer itg) {
     if (itg != null) {
        return itg;
     } else {
        return 0;
     }
}
```

Einde

[1] Wat had je anders gedacht?