Lösningsförslag till omtentamen för TDA540 Objektorienterad Programmering

Institutionen för Datavetenskap

CTH HT-16, TDA540

Dag: 2017-08-24, *Tid:* 14.00-18.00

Uppgift 1

a) En abstrakt klass kan inte instansieras, utan endast kan användas som superklass vid arv. En abstrakt klass innehåller en eller flera abstrakta metoder, metoder som endast har signatur (namn, returtyp och parameterlista) men saknar implementation.

b) En klass kan ärva metoder och variabler från en annan (super) klass. Arv är ett sätt att abstrahera och återanvända kod.

c) Ett undantag (exception) är en oväntad händelse som inträffar under programkörningen. Ett undantag genereras (kastas) av en `throw`-sats. Man kan antingen fånga undantaget i en `try`-sats eller vidarebefordra det.

d) En variabel är en identifierare i ett program som representerar en minnesposition i vilken variabelns värde lagras under exekveringen av programmet.

Uppgift 2

4 Konstruktorn har inte samma namn som klassen.

9 Variabeln `n` är inte deklarerat, det borde vara `num`.

Uppgift 3

```java
public class Exp {
  public static double exp(double x) {
    double res = 1.0;
    double fac = 1.0;
    double pow = 1.0;

    for (int n = 1; n < 100; n++) {
      pow *= x;
      fac *= n;
      res += pow / fac;
    }

    return res;
  }
}
```

Uppgift 4

```java
public class Point {
  private int x, y;

  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
  }

  @Override
  public boolean equals(Object o) {
    if (o == null) {
      return false;
    } else if (!(o instanceof Point)) {
      return false;
    } else {
      return (x == ((Point) o).getX() && y == ((Point) o).getY());
    }
  }

  @Override
  public String toString() {
    return "(" + x + ", " + y + ")";
  }
```

```
}
```

```
import java.util.*;

public class Path implements Iterable<Point> {
  private List<Point> points;
  private boolean loops = false;

  public Path(Point p) {
    points = new LinkedList<Point>();
    points.add(p);
  }

  public void addPoint(Point p) {
    Point last = points.get(points.size() - 1);

    if (p.equals(last)) {
      System.out.println("Can't add same end point.");
    } else {
      // check for loops
      if (points.contains(p))
        loops = true;

      // Add new end point
      points.add(p);
    }
  }

  public Point[] getPoints() {
    Point[] res = new Point[points.size()];

    for (int i = 0; i < points.size(); i++)
      res[i] = points.get(i);

    return res;
  }

  public Iterator<Point> iterator() {
    return points.iterator();
  }

  public boolean loops() {
    return loops;
  }

  @Override
  public String toString() {
    String res = "[";

    // We always have at least one point
    for (Point p : points)
      res += p.toString() + ", ";

```

```
52        return res.substring(0, res.length() - 2) + "]";
53      }
54    }
```

Uppgift 5

Utskriften blir:

```
1
3
3
8
1
3
3
8
```

Uppgift 6

a)
```
static int digit(int n, int k) {
   int d = -1;
   for (int i = 1; i <= k; i++) {
      if (n == 0)  // no more digits
         return -1;
      d = n % 10;  // kth digit
      n = n / 10;  // remaining digits
    }
    return d;
}
```

b)
```
static boolean increasing(int n) {
   int k = 1;
   int d1, d2 = -1;  // dummy initialization for first iteration
   do {
      d1 = d2;
      d2 = digit(n, k);
      if (d1 > d2 && d2 != -1)  // !(d1 <= 2) && d2 is a valid digit
        return false;
      k += 1;
   } while (d2 != -1);  // d2 == -1 iff n has less than k digits
   return true;  // all digits checked successfully
}
```

Uppgift 7

a) 1. error: interfaces cannot be instantiated
   2. accepted: subtype to supertype

3. accepted: subtype to supertype
4. accepted: same type
5. error: supertype to subtype
6. accepted: subtype to supertype
7. error: types not related by inheritance
8. accepted: subtype to supertype
9. OK: same type
10. error: types not related by inheritance

b) 1. `true`
2. `false`
3. `false`
4. compile-time error: Immutable doesn't have method put

Uppgift 8

a)
```
public int multip(G x) {
  int result = 0;
  for (G e: elements)
    if (e == x)
      result++;
  return result;
}
```

b)
```
public boolean permutation(Sequence<G> t) {
  for (G e: this.elements)
    if (multip(e) != t.multip(e))
      return false;
  for (G e: t.elements)
    if (multip(e) != t.multip(e))
      return false;
  return true;
}
```