DAT043 – Objektorienterad programmering

Lösningsförslag till exempeltenta 2017

De tre första uppgifterna utgår från uppgifter i exempeltenta 3 och tentan i TDA547 som gick mars 2015.

---

1. (a) Alternativ i. `list` initieras implicit till `null` och om `add` anropas för ett objekt så inträffar ett undantag.

   (b)
   ```
   1
   1
   1
   ```

   (c)
   ```
   olika
   lika
   ```
   ==-operatorn jämför för referenstyper referenserna, d.v.s. om operanderna pekar på samma objekt. `sa` och `sc` innehåller samma sträng men är ej samma objekt. `sa` och `sb` pekar på samma objekt.

2. (a)
   ```
   public static boolean isIncreasing(int[] a) {
     for (int i = 0; i < a.length-1; i++) {
       if (a[i+1] < a[i]) return false;
     }
     return true;
   }
   ```

   (b)
   ```
   public class Increasing {
     public static boolean isIncreasing(int[] a) { ... }

     public static void main(String[] args) {
       int[] a = new int[args.length];
       for (int i = 0; i < args.length; i++) {
         a[i] = Integer.parseInt(args[i]);
       }
       System.out.println(isIncreasing(a));
     }
   }
   ```

3.
```
public class Find {

  public static void main(String[] arg) {
    if (arg.length != 3) {
      System.out.println("Felaktigt antal argument");
      System.exit(0);
    }
    Scanner s = null;
    try {
      s = new Scanner(new File(arg[0]));
    }
    catch (IOException e) {
      System.out.println("Kan inte öppna filen " + arg[0]);
      System.exit(0);
    }
    PrintStream p = null;
    try {
      p = new PrintStream(new File(arg[1]));
    }
    catch (IOException e) {
      System.out.println("Kan inte skapa filen " + arg[1]);
      System.exit(0);
    }
    while (s.hasNextLine()) {
      String line = s.nextLine();
      if (line.indexOf(arg[2]) >= 0) p.println(line);
    }
    s.close();
    p.close();
  }
}
```

4.
```
public class Person {
  private String name;
  private List<Person> children = new ArrayList<>();

  public Person(String name) {
    this.name = name;
  }

  public void addChild(Person child) {
    children.add(child);
  }

  public String getName() {
    return name;
  }

  public int getNChildren() {
    return children.size();
  }

  public Person getChild(int index) {
    return children.get(index);
  }

  public boolean isDescendantOf(Person n) {
    for (int i = 0; i < n.getNChildren(); i++) {
      if (n.getChild(i) == this) return true;
      if (isDescendantOf(n.getChild(i))) return true;
    }
    return false;
  }
}
```

5.
```
public static int[] merge(int[] a, int[] b) {
  int[] c = new int[a.length + b.length];
  int ia = 0, ib = 0, ic = 0;
  while (ia < a.length && ib < b.length) {
    if (a[ia] <= b[ib]) {
      c[ic++] = a[ia++];
    } else {
      c[ic++] = b[ib++];
    }
  }
  while (ia < a.length) {
    c[ic++] = a[ia++];
  }
  while (ib <= b.length) {
    c[ic++] = b[ib++];
  }
  return c;
}
```

Ett annat sätt är att bortse från att `a` och `b` är sorterade, kopiera alla element i dem till nya arrayen efter varandra och sedan sortera nya arrayen med t.ex. urvalssortering.

6.
```
public static <E> Set<E> union(Set<E> a, Set<E> b) {
  Set<E> c = new HashSet<E>();
  Iterator<E> i = a.iterator();
  while (i.hasNext()) {
    c.add(i.next());
  }
  i = b.iterator();
  while (i.hasNext()) {
    c.add(i.next());
  }
  return c;
}

public static <E> Set<E> intersection(Set<E> a, Set<E> b) {
  Set<E> c = new HashSet<E>();
  Iterator<E> i = a.iterator();
  while (i.hasNext()) {
    E e = i.next();
    if (b.contains(e)) c.add(e);
  }
  return c;
}
```