Programmering Fortsättning

Övning 1, Imperativ Programmering

Joachim von Hacht

Detta är mest repetition eftersom vi inte hunnit gå igenom så mycket. För att lösa problemen kan man naturligtvis testa koden själv (finns på kurssida). Boken, föreläsningsanteckningarna, Java Language Specification, webben (länkar från kurssida) och "Navigate" > "Go to Source" i NetBeans (visar källkoden för Javas standardklasser) är tänkta som källor.

1 Referenser

Uppgifterna nedan *skall* redovisas genom att man utifrån koden illustrerar förhållandet mellan variabler och referenser (ritar boxar och pilar)

1. Avgör för varje kommentar med frågetecken i koden om uttrycket är sant eller falskt[1]. Förklara! 2p

```
package hw1._1_1;
public class References {
    public void checkRefs() {
        String s1 = "True";
        String s2 = "True";
        // s1 == s2 ?

        String s3 = new String("False");
        String s4 = new String("False");
        //s3 == s4?

        String s5 = "True";
        String s6 = "Tr" + "ue";
        //s5 == s6?
```

[1] En grundregel är att alltid undvika språkets mörka hörn, men detta är en kurs så...

```
        String s7 = "False";
        String sx = "F";
        String s8 = sx + "alse";
        //s7 == s8 ?

        Integer i = new Integer(2);
        Integer j = new Integer(2);
        // i >= j ?
        // i <= j ?
        // i == j ?
    }
}
```

2. Betrakta klasserna SimpleSwapper, ValueHolderSwapper, ValueHolderSwapper2 och ValueHolder och avgör vad som kommer att skrivas ut i Main (main-metoden). Förklara! 2p

```
package hw1._1_2;
class SimpleSwapper {
    public void swap(Integer x, Integer y) {
        Integer temp = x;
        x = y;
        y = temp;
    }
}
```

```
package hw1._1_2;
class ValueHolderSwapper {
    public void swap(ValueHolder v1, ValueHolder v2) {
        Integer tmp = v1.i;
        v1.i = v2.i;
        v2.i = tmp;
    }
}
```

```
package hw1._1_2;
class ValueHolderSwapper2 {
    public void swap(ValueHolder v1, ValueHolder v2) {
        v1 = new ValueHolder(v2.i);
        v2 = new ValueHolder(v1.i);
    }
}
```


Produced with Lyx, the open source wordprocessor

```
package hw1._1_2;
public class ValueHolder {
    public Integer i;
    public ValueHolder(Integer i) {
        this.i = i;
    }
}

package hw1._1_2;
public class Main {
    public static void main(String[] args) {
        SimpleSwapper ss = new SimpleSwapper();
        ValueHolderSwapper vhs = new ValueHolderSwapper();
        ValueHolderSwapper2 vhs2 = new ValueHolderSwapper2();

        int a = 1;
        int b = 2;

        swap(a,b);
        System.out.println("a= " + a + " b= " + b);

        ss.swap(a, b);
        System.out.println("a= " + a + " b= " + b);

        Integer c = new Integer(1);
        Integer d = new Integer(2);
        ss.swap(c, d);
        System.out.println("c= " + c + " d= " + d);

        ValueHolder v1 = new ValueHolder(a);
        ValueHolder v2 = new ValueHolder(b);
        vhs.swap(v1, v2);
        System.out.println("a= " + v1.i + " b= " + v2.i);

        vhs2.swap(v1, v2);
        System.out.println("a= " + v1.i + " b= " + v2.i);
    }

    public static void swap( int x, int y){
     int temp = x;
        x = y;
        y = temp;
   }
}
```


Produced with Lyx, the open source wordprocessor

3. Lägg till en klass Box så att koden nedan kan kompileras och exekveras. Vad kommer att skrivas ut? Förklara! Ändra så att utskriften blir "bättre" (ni tolkar). 2p

```
package hw1._1_3;
public class DoubleBox {
    private Box innerBox = new Box();
    public void setValue(int v) {
        innerBox.setValue(v);
    }
    public int getValue() {
        return innerBox.getValue();
    }
    public DoubleBox copy() {
        DoubleBox newValue = new DoubleBox();
        newValue.innerBox = innerBox;
        return newValue;
    }
    public static void main(String[] args) {
        DoubleBox a = new DoubleBox();
        a.setValue(18);
        DoubleBox b = a.copy();
        a.setValue(23);
        System.out.println("b is " + b.getValue());
    }
}
```

4. Då det inte finns någon referens till ett objekt skräpsamlas det (minnet återvinns). För att snabba på processen kan man tilldela referensen null (och hoppas att det inte finns några andra referenser). En idé är att skapa en "återvinnare". Koden nedan skall fungera som en sådan (alltså återvinna parameter object). Fungerar det?? Vi antar att det innan metodanropet bara finns en referens till objektet. 1p

```
package hw1._1_4;
public class Reclaimer {
    public static void delete(Object object) {
        object = null;
    }
}
```


Produced with Lyx, the open source wordprocessor

2 Accessor- och mutator-metoder

1. Resonera om koden nedan. Några tveksamheter? Tips: Parametrar 1p

```
public class MyClass {

    private String name;
    public void setName(String name) {
        this.name = name.toLowerCase();
    }
    public String getName() {
        return this.name;
    }

}
```

3 Delat tillstånd

Uppgiften skall redovisas genom att man utifrån koden illustrerar förhållandet mellan variabler och referenser (ritar boxar och pilar)

1. Koden nedan är problematisk. Vad är problemet? Åtgärda. 2p

```
package hw1._3_1;
public class Entry {
    public Entry next;
    public Object element;
    Entry(Object e, Entry n) {
        element = e;
        next = n;
    }
}
```

```
package hw1._3_1;
/**
 * Invariant: this.size == | this.header.next* | - 2
 * (i.e. list itself and header not included in length of list)
 */
public class LinkedList {
    private final Entry header = new Entry(null, null);
    private int size;

    public void add(Object o){
        Entry e = new Entry(o, header.next);
```

```
            header.next = e;
            size++;
        }
        /**
         * result = list containing all but the first element
         */
        public LinkedList tail() {
            LinkedList tl = new LinkedList();
            tl.header.next = this.header.next.next;
            tl.size = this.size - 1;
            return tl;
        }
        public int size() {
            return size;
        }
        public void dump() {
         Entry pos = header.next;
            while( pos != null){
             System.out.println(pos.element);
             pos = pos.next;
            }
        }
    }
```


Produced with Lyx, the open source wordprocessor