Programmering Fortsättning

Övning 4, Design och lite Arv

Joachim von Hacht

1 Code smell

1. Vad är ditt omdöme om koden (finns troligen flera tänkbara invändningar)? Antag att koden kompilerar (bortse från kompileringsfel). 1p

```
package _1_1;
public class OrderValidator {
    /**
     * Validates the order and returns the order ID
     */
    private String validate(Order order) {
        String custID = order.getCustomerID();
        String itemID = order.getItemID();
        String product = order.getProduct();
        int qty = order.getQty();
        double price = order.getPrice();
        if (custID != null && itemID != null && !custID.equals("")
                && !itemID.equals("") && qty > 0 && price > 0.0) {
            return OrderUtils.getUniqueOrderId();
        }
        return null;
    }
}
```

2. Switch-satserna nedan betraktas som dålig kod (Switch code smell). Förklara varför? Förbättra! 2p

```
package _1_2;
import static java.lang.Math.PI;
public class Geometer {
```

```
    private static final int SQUARE = 0;
    private static final int RECTANGLE = 1;
    private static final int CIRCLE = 2;
    private double a, b, r;

    public double computeArea(int shape) {
        double area = 0;
        switch (shape) {
        case SQUARE: area = a * a; break;
        case RECTANGLE: area = a * b;break;
        case CIRCLE: area = PI * r * r;break;
        }
        return area;
    }

    public double computePerimeter(int shape) {
        double perimeter = 0;
        switch (shape) {
        case SQUARE:perimeter = 4 * a;break;
        case RECTANGLE:perimeter = 2 * (a + b);break;
        case CIRCLE:perimeter = 2 * PI * r;break;
        }
        return perimeter;
    }
}
```

2 Kopiering

1. Klassen DoubleBoxClonable är en variant av klassen DoubleBox från övning 1 (klassen Box är identisk och visas inte här). Implementera clone och en kopieringskonstruktor för DoubleBoxCloneable och subklassen ExtendDoubleBoxCloneabl så att allt i main fungerar. 2p

```
package _2_1;
public class DoubleBoxCloneable implements Cloneable {
   private Box innerBox = new Box();
   public DoubleBoxCloneable() {
   }
   public DoubleBoxCloneable(DoubleBoxCloneable orig) {
      // TODO
   }
   public DoubleBoxCloneable clone() {
```

```
      // TODO
      return null;
   }
   public void setValue(int v) {
      innerBox.setValue(v);
   }
   public int getValue() {
      return innerBox.getValue();
   }
}

package _2_1;
public class ExtendDoubleBoxCloneable extends DoubleBoxCloneable {
   public Box extendedInnerBox = new Box();
   public ExtendDoubleBoxCloneable clone() {
      // TODO
      return null;
   }
   public ExtendDoubleBoxCloneable(ExtendDoubleBoxCloneable orig) {
      // TODO
   }
   public ExtendDoubleBoxCloneable() {
   }
   public void setExtendedValue(int v) {
      extendedInnerBox.setValue(v);
   }
   public int getExtendedValue() {
      return extendedInnerBox.getValue();
   }
}

package _2_1;
public class Main421 {
   public static void main(String[] args) {
      DoubleBoxCloneable d1 = new DoubleBoxCloneable();
      d1.setValue(111);
      DoubleBoxCloneable d2 = (DoubleBoxCloneable) d1.clone();
      d2.setValue(222);
      System.out.println("d1 is " + d1.getValue());
      System.out.println("d2 is " + d2.getValue());
      ExtendDoubleBoxCloneable e1 = new ExtendDoubleBoxCloneable();
      e1.setValue(333);
      e1.setExtendedValue(333000);
      ExtendDoubleBoxCloneable e2 = e1.clone();
```

```
        e2.setValue(444);
        e2.setExtendedValue(444000);
        // The values for e1 and e2 should NOT be the same
        // but all e1 values should be the same (and so for e2)
        System.out
              .println("e1 is " + e1.getValue() + ":" + e1.getExtendedValue());
        System.out
              .println("e2 is " + e2.getValue() + ":" + e2.getExtendedValue());
        // Using copyctor
        DoubleBoxCloneable d3 = new DoubleBoxCloneable(d1);
        d3.setValue(555);
        System.out.println("d1 is " + d1.getValue());
        System.out.println("d3 is " + d3.getValue());

        // Using copy ctor and subclass
        ExtendDoubleBoxCloneable e3 = new ExtendDoubleBoxCloneable(e1);
        e3.setValue(666);
        e3.setExtendedValue(666000);
        // The values for e1 and e3 should NOT be the same
        System.out
              .println("e1 is " + e1.getValue() + ":" + e1.getExtendedValue());
        System.out
              .println("e3 is " + e3.getValue() + ":" + e3.getExtendedValue());
    }
}
```

3 Likhet

1. Programmeraren har gjort ett fel! Förväntade sig utskriften ”someValue” men får null? Varför? Åtgärda! Tips: Hur fungerar Objects hashCode(). 2p

```
package _3_1;
import java.util.HashMap;
import java.util.Map;
public class Main431 {
    public static void main(String[] args) {
        MyClass m1 = new MyClass();
        MyClass m2 = new MyClass();
        System.out.println(m1.equals(m2)); // This outputs true
        Map<MyClass, String> map = new HashMap<>();
        map.put(m1, "someValue");
        System.out.println(map.get(m2));
    }
}
```

```
package _3_1;

public class MyClass {
    private final StringBuffer b = new StringBuffer();

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((b == null) ? 0 : b.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        MyClass other = (MyClass) obj;
        return b.toString().equals(other.b.toString());
    }
}
```

4 Jämförelse[1]

1. Antag att vi vill sortera klassen Customer nedan. Det kan tänkas att man inte kan förutspå alla totala ordningar som kan vara av intresse vid en viss tidpunkt. Ett vanligt fall är då vi i en applikation visar en tabell och man genom att klicka på kolumnernas rubriker bestämmer om posterna ska sorteras i stigande eller fallande ordning med avseende på värdet i kolumnen. Givet att vi har n kolumner så kan dessa ordnas på $n!$ olika sätt. Om vi dessutom lägger till egenskapen att vi ska kunna ha både stigande och fallande ordning för varje fält så får vi $2^n n!$ möjliga ordningar. Även för små n blir detta snabbt ohanterligt. Vi skulle t.ex. behöva 384 olika komparatorer för att få alla totala ordningar av 4 instansvariabler (16x24 = 384).

   Ett flexiblare sätt är att skapa komparatorer för varje instansvariabel och (dynamiskt) seriekoppla dessa efter behov (då behöver vi bara 4 st). Använd denna idé för att implementera en flexibel sorteringslösning för Customer.

[1] Tack till Pelle E. och Danial E. W.

TIPS: Använd gränssnittet Comparator kombinerat med (ev. varianter på) Decorator och Template mönstren. 3p

```
package _4_1;
public class Customer {
   private final String pNumb;
   private final String firstName;
   private final String lastName;
   public Customer(String pnumb, String firstName, String lastName) {
      this.pNumb = pnumb;
      this.firstName = firstName;
      this.lastName = lastName;
   }
   public String getPNumb() {
      return pNumb;
   }
   public String getFirstName() {
      return firstName;
   }
   public String getLastName() {
      return lastName;
   }

   @Override
   public int hashCode() {
      final int prime = 31;
      int result = 1;
      result = prime * result
            + ((firstName == null) ? 0 : firstName.hashCode());
      result = prime * result + ((lastName == null) ? 0 : lastName.hashCode());
      return result;
   }

   // Note different behavior for null
   // Equals return false
   @Override
   public boolean equals(Object obj) {
      if (this == obj)
         return true;
      if (obj == null)
         return false;
      if (getClass() != obj.getClass())
         return false;
      Customer other = (Customer) obj;
```

```
        if (firstName == null) {
            if (other.firstName != null)
                return false;
        } else if (!firstName.equals(other.firstName))
            return false;
        if (lastName == null) {
            if (other.lastName != null)
                return false;
        } else if (!lastName.equals(other.lastName))
            return false;
        return true;
    }

    @Override
    public String toString(){
        return pNumb + " " + firstName + " " + lastName;
    }
}
```