Objektorienterad programvaruutveckling, fk, TDA550

Exempeltentamen

Uppgift 1.

a) Vad skriver programmet ut? Motivera!

```
public class A {
    public void f() {
        System.out.println("A.f");
    }
    public void h() {
        f();
    }
}
```

```
public class B extends A {
    public void g() {
        h();
    }
}
```

```
public class C extends B {
    public void g(A obj) {
        obj.h();
        g();
    }
    public void f() {
        System.out.println("C.f");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        C c = new C();
        c.g( new B() );
    }
}
```

b) Studera klasserna nedan.

```
public interface Int {
    public void h();
    public void f();
}

public class Base {
    public void f() {}
    public void g() {}
}

public class Sub1 extends Base {
    public void h() {}
}

public class Sub2 extends Base implements Int {
    public void h() {}
}
```

Vilka av metodanropen nedan är tillåtna och vilka ger kompileringsfel? Motivera!

i)
```
Base obj1 = new Sub1();
obj1.f();
obj1.g();
obj1.h();
```

ii)
```
Base obj2 = new Sub2();
obj2.f();
obj2.g();
obj2.h();
```

iii)
```
Int obj3 = new Sub2();
obj3.f();
obj3.g();
obj3.h();
```

(5 poäng)

Uppgift 2.

Betrakta klasserna:

```
public class USBStick {
   public USBStick(int x) { ... }
   public int occupiedSpace() { ... }
}

public class DVDMemory {
   public DVDMemory(int x) { ... }
   public int getMemoryUsage() { ... }
}

public class SSDUnit {
   public SSDUnit(int x) { ... }
   public int memoryConsumption() { ... }
}
```

samt följande klientkod:

```
public class Main {
   public static long getMemoryUsage(List<Object> units) {
      long usage = 0;
      for (Object o : units) {
         if (o instanceof USBStick)
            usage += ((USBStick)o).occupiedSpace();
         else if ( o instanceof DVDMemory )
            usage += ((DVDMemory)o).getMemoryUsage();
         else if (o instanceof SSDUnit)
            usage += ((SSDUnit)o).memoryConsumption();
      }
      return usage;
   }

   public static void main(String[] args) {
      List<Object> units = new LinkedList<>();
      units.add(new USBStick(1000));
      units.add(new DVDMemory(80000));
      units.add(new SSDUnit(20000));
      System.out.println(getMemoryUsage(units));
   }
}
```

a) Koden ovan bryter mot en viktig designprincip, vilken?

b) Refaktorera koden så att den baseras på polymorfism.

(5 poäng)

Uppgift 3.

En Brittisk webshop prissätter sina varor i valutan USD. Kunder inom Storbrittanien betalar i valutan GBP, övriga europeiska kunder i EURO och resten av världen i USD. En PriceCalculator är ett objekt som räknar ut det totalpris inklusive frakt som kunden skall betala i den valda valutan:

```
public abstract class PriceCalculator {
   public static boolean isBrittish(String country) { ... }
   public static boolean isEuropean(String country) { ... }
   public abstract float calculateTotalPrice(float subTotal, float weight, String country);
}

public class GBPCalculator extends PriceCalculator {
   public GBPCalculator(String country) { ... }
   public float calculateTotalPrice(float subTotal, float weight, String country) { ... }
}

public class EUROCalculator extends PriceCalculator {
   public EUROCalculator(String country) { ... }
   public float calculateTotalPrice(float subTotal, float weight, String country) { ... }
}

public class USDCalculator extends PriceCalculator {
   public USDCalculator(String country) { ... }
   public float calculateTotalPrice(float subTotal, float weight, String country) { ... }
}

public class BadMain {
   public static void main(String[] arg) {
      String country = arg[0];
      PriceCalculator priceCalculator;
      float subTotal = 12345.0F;  // USD (priset antas vara satt i dollar)
      float weight = 123.3F;
      if (PriceCalculator.isBrittish(country))
         priceCalculator = new GBPCalculator(country);
      else if (PriceCalculator.isEuropean(country))
         priceCalculator = new EUROCalculator(country);
      else
         priceCalculator = new USDCalculator(country);

      System.out.println(priceCalculator.calculateTotalPrice(subTotal, weight, country));
   }
}
```

a) Klassen BadMain bryter mot en viktig designprincip, vilken?

b) Refaktorera koden enligt designmönstret *Factory*. (De fyra första klasserna skall ej ändras.)

(5 poäng)

Uppgift 4.

Antag att följande klasser är givna:

```
public class Furniture {...}
public  class CoffeeTable extends Table {...}
public class Table extends Furniture{...}
public class DinnerTable extends Table {...}
```

Antag vidare att vi I ett program har gjort följande deklarationer:

```
Object o;
Furniture f;
CoffeeTable ct;
Table t;
DinnerTable dt;
List<? extends Furniture> lef;
List<? extends Table> let;
List<? super Table> lst;
```

Ange för var och en av nedanstående satser om satsen är korrekt eller ger kompileringsfel. Motivera!

a) lef.add(t);
b) lef.add(o);
c) lst.add(t);
d) lst.add(o);
e) let.add(dt);
f) o = lef.get(1);
g) f = let.get(1);
h) ct = let.get(1);
i) t = lst.get(1);
j) o = lst.get(1);

(5 poäng)

Uppgift 5.

a) Rangordna nedanstående villkoren efter avtagande logisk styrka (A => B):
   i) i < 20
   ii) true
   iii) 40 < i || i < 30
   iv) false
   v) 0 > i && i < 10

b) Uppfyller någon av implementationerna ArraySearchImpl1 och ArraySearchImpl2 kontraktet för ArraySearch? Är de utbytbara mot varandra? Motivera!

```
public interface ArraySearch {
   //@Pre a != null
   //@Post returns the index of an arbitrary occurrence of x if such an element exists in a, otherwise -1
   int find(int[] a,int x);
}

public class ArraySearchImpl1 implements ArraySearch {
   //@Pre True
   //@Post returns the index of the first occurrence of x if such an element exists in a, otherwise -1
   public int find(int[] a,int x) { ... }
}
public class ArraySearchImpl2 implements ArraySearch {
   //@Pre a != null && a.size > 0
   //@Post returns the index of an arbitrary occurrence of x if such an element exists in a
   public int find(int[] a,int x) { ... }
}
```

(4 poäng)

Uppgift 6.

En gles vektor är en vektor där de flesta elementen har värdet 0. Exempel på en gles vektor med 14 element varav 2 är skilda från 0:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
0.0	0.0	0.0	0.0	0.0	0.0	3.12	0.0	0.0	0.0	0.0	4.21	0.0	0.0

Att lagra glesa vektor i ett fält av typen **double**[] är ineffektivt avseende minnesåtgång, eftersom även alla nollor måste lagras. En betydligt bättre metod är att endast lagra element som inte är 0 i en Map, där nyckel/värde-paret är elementets index respektive elementets värde.

En gles matris specificeras av nedanstående interface:

```
public interface  SparseVector {
    /**
     * Puts val in element with index i.
     * @param i index
     * @param val the new value
     * @throws IllegalArgumentException if i < 0 or i >= the size of this vector
    */
    public void put(int i, double val);

    /**

     * Gets the value in the element with index i.
     * @param i index
     * @throws IllegalArgumentException if i < 0 or i >= the size of this vector
    */
    public double get(int i);

     /**
     * Returns the size of this vector.
     * @return the size of this vector
    */
    public int size();

    /**
     * Computes the dot product of this vector and v.
     * @param v the other vector
     * @return the dot product
     * @throws IllegalArgumentException if the two vectors have different sizes
    */
    public double dot(SparseVector v);
}
```

Skriv en klass SparseVectorMap som implementerar interfacet SparseVector. Internt skall klassen SparseVectorMap representera den glesa matrisen med hjälp av en Map. Klassen ska ha två konstruktorer:

public SparseVectorMap(**int** size)	som skapar en instans av SparseVectorMap med storleken size, i vilken alla element är 0.
public SparseVectorMap(**double**[] v)	som skapar en instans av SparseVectorMap med samma element som i fälet v.

Tips: Metoden dot(SparseVector v) ska beräkna skalärprodukten (eng. dot product) på följande sätt:

$$a \cdot b = a_0 b_0 + a_1 b_1 + ... + a_{n-1} b_{n-1}$$

Tänk på att parametern v har typen SparseVector och inte SparseVectorMap.

(10 poäng)

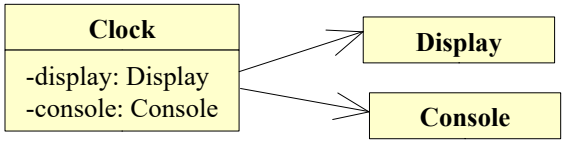
Uppgift 7.

Nedan har vi de tre klasserna Clock, Display och Console. Klassen Clock används för att avbilda en klocka och innehåller bl.a. metoden tick som räknar fram tiden. Klasserna Display och Console har båda metoder för att visa aktuell tid, metoderna show respektive print.

```
public class Clock {
    private Time time;
    private Display display;
    private Console console;

    public Clock(Display display, Console console) {
        this.display = display;
        this.console = console;
    }

    public void tick() {
        time.increase();
        display.show(time);
        console.print(time);
    }
    //more methods and constructors not shown here
}
public class Display {
    public void show(Time time) { ... }
    ...
}
public class Console {
    public void print(Time time) { ... }
    ...
}
```



Designen har bristen att Clock är beroende av klasserna Display och Console. Det borde vara tvärtom.

Ändra designen så att designmönstret Observer realiseras (genom att använda Observable och Observer eller genom att använda PropertyChangeSupport och PropertyChangeListener). Det räcker om du visar koden för klasserna Clock och Display (Console blir analog med Display).

(6 poäng)

Uppgift 8.

Följande interface och klass är givna:

```
public interface EnemyAttacker {
    public void fireWeapon();
    public void driveForward();
    public void assignDriver(String driver);
}
public class EnemyRobot {
    public void smashWithHand() {
        //code omitted
    }
    public void walkForward() {
        //code omitted
    }
    public void reactToHuman(String human) {
        //code omitted
    }
}
```

Skriv en klass EnemyRobotAdapter som implementerar desigmönstret *Adapter* för att anpassa gränssnittet för typen EnemyRobot till typen EnemyAttacker.

(5 poäng)

Uppgift 9.

Ett rationellt tal $\frac{p}{q}$ består av en täljare p och en nämnare q (där både p och q är heltal samt $q \neq 0$). Betrakta klassen Rational nedan som avbildar rationella tal:

```
//Class invariant: q > 0
public class Rational {
    private int p;
    private int q;
    public Rational(int p, int q) {
        this.p = p;
        this.q = q;
    }
    public Rational add(Rational other) {
        return new Rational(p*other.q + other.p*q, q*other.q);
    }
     // other artimethic operations omitted
    public boolean equals(Object other) {
        if (this == other)
            return true;
        if (other == null || getClass() != other.getClass())
            return false;
        Rational o = (Rational) other;
        return p == o.p && q == o.q;
    }
    private int gcd(int x,int y) {
        // returns the greatest common divisor of x and y.
        //code not shown here
    }
}
```

a) Rent matematiskt gäller förstås att t.ex. 1/3 + 1/3 = 2/3, men givet

```
Rational a = new Rational(1, 3);
Rational b = new Rational(2, 3);
```

kommer b.equals(a.add(a)) att returnera **false**! Varför?

Modifiera klassen på så sätt att x.equals(y) ger **true** om x och y avbildar två rationella tal som är matematiskt lika, dvs att anropet b.equals(a.add(a)) ovan skall ge värdet **true**. Metoderna equals och gcd får inte förändras.

Tips 1: När skall klassinvarianten vara uppfylld? Tips 2: Talet noll kan representeras som 0/1.

b) Är det lämpligt att lagra objekt av klassen Rational i ett HashSet? Om inte, komplettera klassen med det som saknas?

c) Gör klassen Membership kopierbar. Du kan förutsätta att SomeClass är kopierbar.

```
public class Membership {
    private String name;
    private int age;
    private LinkedList<SomeClass> someObjects;

    public Membership() {
        someObjects = new LinkedList<>();
    }
     // other members omitted
}
```

(7 poäng)

Uppgift 10.

Betrakta deklarationerna:

```
public class E1 extends Exception { ... }
public class E2 extends E1 { ... }
public class E3 extends E2 { ... }
public class E4 extends E1 { ... }

public class Base {
    public void f() throws E2 { ... }
    public void g() throws E1 {
        if (some condition)
            System.out.print("g succeeded");
        else throw new ... ;
    }
}

public class Sub1 extends Base{
    public void f() throws E1 { ... }
}

public class Sub2 extends Base{
    public void f() throws E3 { ... }
}

public class Sub3 extends Base{
    public void f() throws E2 {
        throw new E3();
    }
}

public class Sub4 extends Base{
    public void f() throws E3 {
        throw new E2();
    }
}
```

a) Är deklarationen av Sub1, Sub2, Sub3 respektive Sub4 typkorrekt? Motivera!

b) Vilka utskrifter är möjliga vid exekvering av kodavsnittet nedan? Motivera?

```
Base obj = new Base();
try {
   try {
      obj.g();
   }
   catch ( E2 e ) {
       System.out.print("catched E2");

   }
   catch ( E4 e ) {
       System.out.print("catched E4");
   }
}
catch ( E1 e ) {
    System.out.print("catched E1");
}
```

(4 poäng)