TENTAMEN: Objektorienterad programmering

Läs detta!

- *Uppgifterna är inte ordnade efter svårighetsgrad.*
- Börja varje hel uppgift på ett nytt blad.
- Ordna bladen i uppgiftsordning.
- Skriv din tentamenskod på varje blad (så att vi inte slarvar bort dem).
- **Skriv rent dina svar. Oläsliga svar r ä t t a s e j!**
- Programkod som finns i tentamenstesen behöver ej upprepas.
- Programkod skall skrivas i Java 5, eller senare version, och vara indenterad och renskriven.
- Onödigt komplicerade lösningar ger poängavdrag.
- Givna deklarationer, parameterlistor etc. får ej ändras.
- Läs igenom tentamenstesen och förbered ev. frågor.

I en uppgift som består av flera delar får du använda dig av funktioner klasser etc. från tidigare deluppgifter, även om du inte löst dessa.

Lycka till!

Uppgift 1

Välj **ett** alternativ för varje fråga! Garderingar ger noll poäng. Inga motiveringar krävs. Varje korrekt svar ger två poäng. Besvara delfrågorna 1.1-1.5 på ett blad.

1. Vilken av `clone`-metoderna kopierar `B` på korrekt sätt?

```
public class A {
   ...
   public A clone()  { ...
   ...
}
public class B {
   private ArrayList<A> list;
   private int count;
   private String name;
   ...
   public B clone() { se nedan }
   ...
}
public B clone() throws CloneNotSupportedException {  // 1
   B result = null;
   result = (B)super.clone();
   for ( int i = 0; i < list.size(); i++ )
      result.list.set(i,list.get(i).clone());
   return result;
}
public B clone() throws CloneNotSupportedException {  // 2
   B result = null;
   result = (B)super.clone();
   result.list = (ArrayList<A>)list.clone();
   return result;
}
public B clone() throws CloneNotSupportedException {  // 3
   B result = null;
   result = (B)super.clone();
   result.list = (ArrayList<A>)list.clone();
   for ( int i = 0; i < list.size(); i++ )
      result.list.set(i,list.get(i).clone());
   result.name = name.clone();
   return result;
}
public B clone() throws CloneNotSupportedException {  // 4
   B result = null;
   result.list = (ArrayList<A>)list.clone();
   for ( int i = 0; i < list.size(); i++ )
      result.list.set(i,list.get(i).clone());
   return result;
}
```

a. Endast variant 1 är korrekt.
b. Endast variant 2 är korrekt.
c. Endast variant 3 är korrekt.
d. Endast variant 4 är korrekt.
e. Ingen av 1-4 är korrekt.

2. Givet klassen `C`

```
public class C {
    private int[] arr;

    public C(int size) {
        arr = new int[size];
    }
}
```

Är följande korrekt?

```
C obj = new C();
```

a. Nej, undantaget `NullPointerException` kastas.
b. Nej, fältet `arr` blir tomt.
c. Nej, det blir kompileringsfel.
d. Ja.

3. Vilket påstående om en objektorienterad design stämmer bäst?
a. Hög kohesion medför ofta hög koppling.
b. Hög kohesion medför ofta låg koppling.
c. Låg kohesion medför ofta låg koppling.

4. Givet

```
public interface A
public class B
public class C extends B implements A
```

Vilket är typkorrekt?

a.
```
B x = new C();
A y = x;
```

b.
```
A x = new C();
B y = x;
```

c.
```
C y = new C();
B x = y;
```

d.
```
B y = new C();
C x = y;
```

e. Inget av ovanstående.

Vi kan tilldela sub till super —> super = sub

super.equals(sub)

sub instanceof super

5. Ett av fallen a, b eller c ger ett kompileringsfel, vilket?

```
public class A {
    private void f() {}
    protected void g() {}
    public void h() {}
}
public class B extends A {
    public void f() {}      // a
    public void g() {}      // b
    private void h() {}     // c
}
```

f() är private här och subklassen kan inte se detta och public f() i subklassen har inga relation med private f() i superklassen.

(10 p)

Uppgift 2

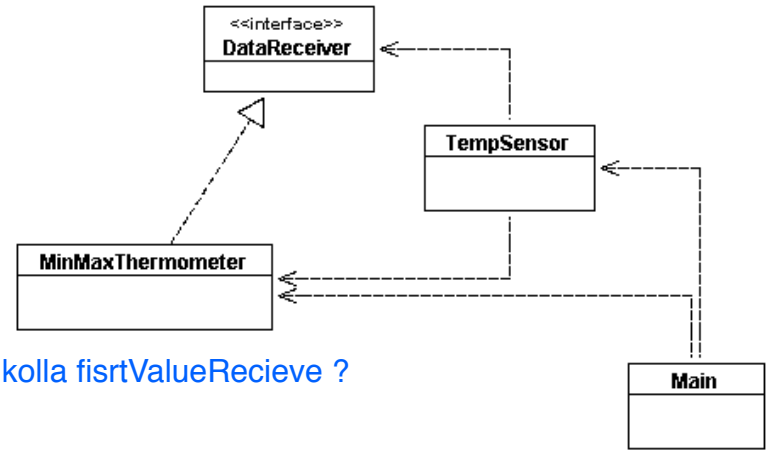
Antag att instanser av klassen `TempSensor` kan mäta temperaturer och skicka dessa till associerade mottagarobjekt. Vi går inte in tekniskt på hur en sådan klass kan definieras utan ger här bara de för uppgiften nödvändiga delarna:

```
public class TempSensor {
    public TempSensor(int updateIntervalMs) { ... }
    public void addReceiver(DataReceiver receiver)
    ...
}
```

Konstruktorn tar ett uppdateringsintervall mätt i millisekunder som argument. Efter varje sådant intervall kommer objektet att uppdatera ett mottagarobjekt med sitt aktuella temperaturvärde. Mottagarobjektet kopplas till med metoden `addReceiver`. Typen `DataReceiver` definieras

```
public interface DataReceiver {
    void setValue(double value);
}
```

Följande klassdiagram beskriver klassrelationerna som skall skapas i uppgifterna nedan:



- Varför behövs det kolla fisrtValueRecieve ?

a)
Konstruera klassen `MinMaxThermometer` som håller reda på aktuell temperatur samt de lägsta och högsta temperaturerna under mätperioden. En mätperiod pågår sedan objektet skapades, eller sedan metoden `reset` anropades senaste gången. Klassen skall ha följande metoder:

`public double` `getTemperature()`

Returnerar det aktuella temperaturvärdet.

`public double` `getMinTemperature()`

Returnerar det lägsta mottagna temperaturvärdet under mätperioden.

`public double` `getMaxTemperature()`

Returnerar det högsta mottagna temperaturvärdet under mätperioden.

`public void` `reset()`

Återställer lägsta och högsta temperaturvärdet till aktuellt temperaturvärde.

Klassen skall dessutom implementera gränssnittet `DataReceiver`, och den skall ha en konstruktor som initierar de tre temperaturvärdena till 0.

(9 p)

b)
Skriv en `main`-metod som skapar samt kopplar ihop ett sensorobjekt med ett objekt av klassen som konstruerades i uppgift a. Sensorobjektet skall rapportera temperaturvärden en gång per sekund.

(3 p)

Uppgift 3

En ornitologförening behöver ett IT-hjälpmedel för att hålla reda på medlemmarnas fågelobservationer. En observation innehåller uppgifter om fågelartens namn, observationsplats, datum, samt observatörens namn:

```
public interface Observation {
    String getBirdName();
    String getPlace();
    String getDate();
    String getObserver();
}
```

Uppgiften är att konstruera klassen `BirdClub` som håller reda på vilka olika fågelarter medlemmarna rapporterat. Klassen skall ha följande metoder:

`public void addObservation(String member,String birdName)`
Registrerar att `member` sett `birdName`. Om `member` ej är känd skall en ny medlem med det namnet läggas till.

`public void addObservation(List<Observation> l)`
Registrerar observationerna i `l` på respektive medlem.

`public int getRank(String member)`
Returnerar antalet olika arter som `member` rapporterat. Om medlemmen är okänd skall 0 returneras.

`public void print300()`
Den som rapporterat minst 300 olika fågelarter kan bli medlem i den s.k. ”Klubb 300”. Metoden skall skriva ut alla medlemmar i klubben som uppfyller detta kriterium.

Definiera instansvariabler av lämpliga typer för att lagra informationen om medlemmarna. Artnamnen som registreras för varje medlem skall vara unika. Lagra därför dessa i mängder (set). Skriv också en konstruktor som skapar lämpliga objekt.

(12 p)

Uppgift 4

Studera följande kod:

```
public abstract class A {
    private float y;
    public A(float z) { y = z; }
    public abstract float f(float x);
    public float g(float x) { return h(x - y); }
    public float h(float x) { return x+3; }
}

public class B extends A {
    public B(float x) { super(x); }
    public float f(float x) { return 100 - g(x); }
    public float h(int x) { return x*3; }
}

public class C extends A {
    public C(float x) { super(x); }
    public float f(float x) { return 100 + g(x); }
    public float h(float x) { return x*2; }
}

public class Main {
    public static void main(String[] args) {
        A[] arr = new A[]{new B(205),new C(205)};
        float result = 0;
        for ( A obj : arr )
            result = result + obj.f(50);
        System.out.println(result);
    }
}
```

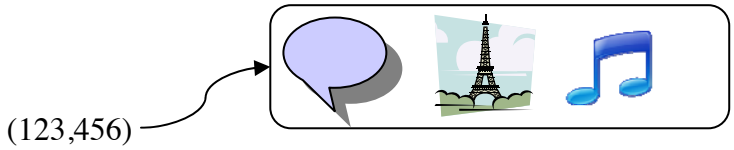
The B class doesn't override h() method on the subclass because they take different parameter.
but
The C class override h() and when we call h() method it is C.h() in fact.

Vad skriver `main` ut? Förklara också vilka metoder som anropas – och varför, med vilka parametervärden, samt vad anropen returnerar.

(10 p)

Uppgift 5

I denna uppgift skall du bygga en liten modell av ett system för ”omvänd geotaggning”. Vanligen innebär geotaggning att man knyter en koordinat till t.ex. en bild man tagit med en digitalkamera. Här skall vi istället göra så att man till en koordinat i kartan kan associera en lista av `Tag`-objekt. Ett sådant objekt kan innehålla en textkommentar, ett namn på en ljudfil som kan spelas upp, eller ett namn på en bildfil som kan visas. [1]



För att förenkla problemet byter vi ut GPS-koordinater mot punkter i planet:

```
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; }
}
```

a) Överskugga metoden `equals` i `Point`. Två punkter är lika om de anger exakt samma position i planet. Du kan räkna med att `Point` inte skall ha några subklasser.

(3 p)

b) Överskugga metoden `hashCode` i `Point`.

(3 p)

Gränssnittet `Tag` ser ut så här

```
public interface Tag {
    void play();
}
```

Tanken är att dynamiskt beteende som t.ex. att visa en bild, eller spela upp ett ljud skall implementeras av metoden `play` i subklasser till `Tag`. Exempel:

```
public class CommentTag implements Tag {
    public CommentTag(String comment) { … }
    public void play() { … } // skriver ut kommentartexten
}
```

Det finns en färdig klass för att visualisera en bildfil:

```
public class ImageViewer {
    public ImageViewer (String imageFileName) throws IOException …
    public void displayImage() …
}
```

c) Definiera klassen `ImageTag` så att den implementerar `Tag`. Konstruktorn skall ta namnet på en bildfil som parameter. Klassen skall använda `ImageViewer`. När `play` anropas skall bilden visas. Ev. undantag skall propageras vidare.

(1 p)

[1] System som bygger på den här principen används t.ex. för rapportering av fel i Göteborgs gatunät.

d) Vi skall nu skapa klassen `GeoTags` som håller reda på associationer mellan punkter och listor av taggar med hjälp av en avbildningstabell (map). För att hantera situationer med okända punkter finns undantagsklassen `UnknownPointException`.

Inför lämpliga instansvariabler och en konstruktor i klassen `GeoTags` och implementera följande metoder:

```
public boolean hasPoint(Point p)
```
- Returnerar **`true`** om `p` finns i tabellen, **`false`** annars.

```
public void addTag(Point p,Tag t)
```
- Adderar `t` till `p`:s taggar i tabellen. Om `p` ej finns i tabellen skall punkten först införas.

```
public Iterator<Tag> getTags(Point p) throws UnknownPointException
```
- Returnerar en iterator till `p`:s tagglista. Kastar `UnknownPointException` om `p` ej finns i tabellen.

(5 p)

e) Skriv ett kodavsnitt som skapar ett `GeoTags`-objekt samt lägger in följande information för punkten `p` med koordinaterna `(123,456)`:
- En texttagg med innehållet: `"5 m djupt hål i Kungsgatan"`.
- En bildtagg med filnamnet: `"kungsgatan.jpg"`.

Skriv därefter kod som spelar upp taggarna för `p`. Eventuella undantag skall hanteras och skrivas ut.

(4 p)