Tentamen för TDA550 **Objektorienterad programvaruutveckling IT, fk**

DAG: 12-12-19 TID: 8:30 – 12:30

Ansvarig: Christer Carlsson, ankn 1038

Förfrågningar: Christer Carlsson

Resultat: erhålls via Ladok

Betygsgränser:

3:a	24 poäng
4:a	36 poäng
5:a	48 poäng
maxpoäng	60 poäng

Siffror inom parentes: anger maximal poäng på uppgiften.

Granskning: Måndag 21/1 kl 10-12 och torsdag 24/1 kl 15-17, rum 6128 i EDIT-huset.

Hjälpmedel: Inga hjälpmedel är tillåtna förutom bilagan till tesen.

Var vänlig och: Skriv tydligt och disponera papperet på lämpligt sätt.

Börja varje uppgift på nytt blad. Skriv ej på baksidan av papperet.

Observera: Uppgifterna är ej ordnade efter svårighetsgrad. Titta därför igenom hela tentamen innan du börjar skriva.

Alla program skall vara välstruktruerade, lätta att överskåda samt enkla att förstå.

Vid rättning av uppgifter där programkod ingår bedöms principella fel allvarligare än smärre språkfel.

LYCKA TILL!!!!

Uppgift 1.

Betrakta nedanstående klasser:

```
abstract public class A {
    abstract public void  whiz(int i);
}//A

public class B extends A {
    public B( ) { }
    public void  whiz(int i) {
        System.out.println("class B");
    }
    public void bam(int i) {
        System.out.println(i);
        whiz(i+1);
    }
}//B
```

```
public class C extends B {
    public C( ) {  }
    public void  whiz(int i) {
        super.whiz(i-1);
        System.out.println("class C");
    }
    public void bam(double d) {
        System.out.println("fraction");
        whiz(10);
    }
}//C
```

a) Vad blir resultatet för var och en av följande satser (ger kompileringsfel, ger exekveringsfel, skriver ut xxx, etc)?

```
A obj = new B();
obj.bam(3);
```

b) Vad blir resultatet för var och en av följande satser (ger kompileringsfel, ger exekveringsfel, skriver ut xxx, etc)?

```
B obj = new B();
obj.bam(3);
```

c) Vad blir resultatet för var och en av följande satser (ger kompileringsfel, ger exekveringsfel, skriver ut xxx, etc)?

```
B obj = new C();
obj.bam(3);
```

(3 poäng)

Uppgift 2.

Betrakta nedanstående (ofullständiga) klasser:

```
public class Button {
  private Lamp itsLamp;
  // other necessary instance variables

  public void poll() {
    if (! itsLamp.isOn())
      itsLamp.turnOn();
  }//poll
  // other necessary methods
}//Button
```

```
public class Lamp {
    private boolean on;
    // other necessary instance variables

    public boolean isOn() {
        return on;
    }//isOn
    public void turnOn() {
        //some code
    }//turnOn
    // other necessary methods
}//Lamp
```

Klasserna bryter mot en grundläggande designprincip. Vilken? Om du inte vet namnet designprincipen så beskriv vad principen säger. Refaktorera koden så att designprincipen följs!

(3 poäng)

Uppgift 3.

Antag att vi har nedanstående klass för att avbilda cirklar i en grafisk 2D-tillämpning:

```
public class Circle {
    private int x; // x and y center coordinates
    private int y;
    private int radius; // radius
    //constructors and methodes not shown here

    // two Circles are considered to be equal if they have the same center coordinates
    public boolean equals(Object other) {
        if (this == other)
            return true;
        if (other == null)
            return false;
        if (this.getClass() != other.getClass())
            return false;
        Circle c = (Circle) other;
        return this.x == c.x && this.y == c.y;
    }//equals
}//Circle
```

Nedan ges 4 förslag på implementeringar av metoden hashCode() för klassen:

1)
```
public int hashCode() {
    return x;
}
```

2)
```
public int hashCode() {
    return 5*x + 11*y;
}
```

3)
```
public int hashCode() {
    return x + y + radius;
}
```

4)
```
public int hashCode() {
    return 42;
}
```

a) Vilka av ovanstående metoder uppfyller kraven, som *The Java Language Specification* ställer, för att vara en korrekt implementation. Motivera ditt svar.

b) Vilken av ovanstående metoder bör/skall väljas för att inkluderas i klassen Circle? Motivera ditt svar!

(4 poäng)

Uppgift 4.

Betrakta nedanstående kodsegment:

```
int [ ] input = {100, 37, 49 };
boolean result1 = contains(input, new Prime());
boolean result2 = contains(input, new PerfectSquare());
boolean result3 = contains(input, new Negative());
```

Metoden contains tar som parameter ett heltalsfält samt ett objekt som specificerar en egenskap. Metoden returnerar värdet **true** om fältet uppfyller egenskapen som specificeras av objektet, annars returnera metoden värdet **false**.

Det avsedda resultat som skall erhållas från ovanstående kod är att

result1 skall bli **true** eftersom heltalsfältet input innehåller primtalet 37,
result2 skall bli **true** eftersom heltalsfältet input innehåller talen 100 och 49 som båda är kvadrattal
result3 skall bli **false** eftersom heltalsfältet input inte innehåller några negativa tal

a) Definiera ett interface som behövs för att beskriva den andra parametern till metoden contains.

b) Implementera metoden contains (som en statisk metod). Metoden skall använda interfacet från deluppgift a).

c) Implementera klassen Negative, som uppfyller interfacet i deluppgift a)).

d) Vilket designmönster har du använt i implementeringen ovan?

(6 poäng)

Uppgift 5.

Betrakta nedanstående två klasser för att representera punkter i planet respektive polygoner:

```
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 void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
}//Point
```

```
public class Polygon {
    private ArrayList<Point> points;
    private int noOfPoints;
    //constructors and methods not shown here.
}//Polygon
```

a) Skriv en clone-metod för klassen Point, samt ange vad som i övrigt behöver göras för att klassen Point skall bli kloningsbar.

b) Skriv en clone-metod, som returnerar en djup kopia, för klassen Polygon, samt ange vad som i övrigt behöver göras för att klassen Polygon skall bli kloningsbar.

(6 poäng)

Uppgift 6.

Betrakta nedanstående klass (som använder klassen Point i föregående uppgift):

```
import java.util.*;
public class Harmful {
    private List<String> names;
    private final List<String> fixedNames;
    private final List<Point> points;
    public Harmful() {
        names = new ArrayList<String>();
        fixedNames = new ArrayList<String>();
        points = new ArrayList<Point>();
    }
    public List<String> getNames() {
        return names;
    }
    public List<String> getFixedNames() {
        return fixedNames;
    }
    public String getFirstName() {
        if (names != null)
            return names.get(0);
        else
            return null;
    }
    public Point getFirstPoint() {
        if ( points != null)
            return points.get(0);
        else
            return null;
    }
    //other methods not shown here
}//Harmful
```

Ange för var och en av metoderna getNames, getFixedNames, getFirstName och getFirstPoint om metoden riskerar att exponera den interna representationen. Motivera!

(4 poäng)

Uppgift 7.

Betrakta nedanstående fyra specifikationer för metoden

double sqrt(**double** x)

som returnerar kvadratroten för argumentet x.

I) @requires x >= 0
@return y such that |y*y – x| <= 0.0001

II) @requires x >= 0
@return y such that |y*y – x| <= 0.01

III) @return y such that |y*y – x| <= 0.0001
@throws IllegalArgumentException if x < 0

IV) @requires x > 0
@return y such that |y*y – x| <= 0.0001

Ange för vart och ett av nedanstående par av specifikationer, vilken av specifikationerna som är starkast. Om det inte går att avgöra vilken specifikation som är starkast skall detta anges. Motivera dina svar!

a) I och II

b) I och III

c) II och III

d) II och IV

(4 poäng)

Uppgift 8.

Antag att vi har följande två klasser:

```
public class A { ... }

public class B extends A { ... }
```

a) Vad händer om du försöker kompilera och köra nedanstående program? Uppstår ett kompileringsfel, ett exekveringsfel eller kommer programmet att avslutas på ett kontrollerats sätt? Motivera ditt svar!

```
public class Aarray {
    public static void update( A[] aarr, A a ) {
        aarr[0] = a;
    }
    public static void main( String[] args ) {
        B[] barr = new B[5];
        update(barr, new A());
    } // main
} // Aarray
```

b) Vad händer om du försöker kompilera och köra nedanstående program? Uppstår ett kompileringsfel, ett exekveringsfel eller kommer programmet att avslutas på ett kontrollerats sätt? Motivera ditt svar!

```
import java.util.*;
public class ASet {
    public static void update( Set<A> aset, A a ) {
        aset.add(a );
    }
    public static void main( String[] args ) {
        Set<B> bset = new TreeSet<B>();
        update(bset, new A());
    } // main
} // ASet
```

(4 poäng)

Uppgift 9.

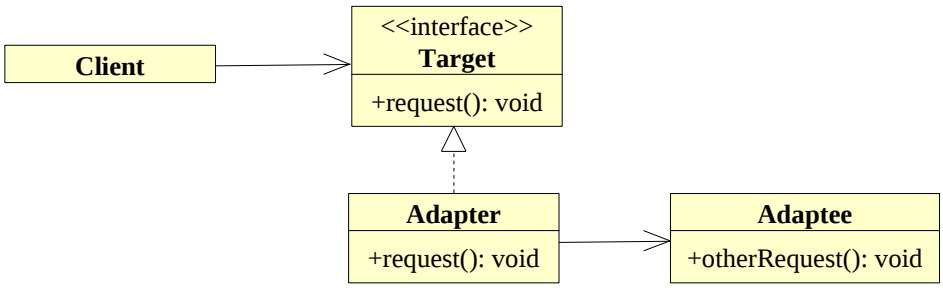
Följande interface och klass är givna:

```
interface Set {
    public int size();
    public boolean isEmpty();
    public boolean member(int x); //check if x is a member of the set
    public void add(int x); //add x to the set
    public void remove(int x); //remove x from the set if x is a member of the set
}//Set

public class Stack {
    private int[] contents;
    private int top = 0;
    public Stack(int maxSize) {
        contents = new int[maxSize];
    }//constructor
    public int size() {
        return top;
    }//size
    public boolean isEmpty() {
        return top == 0;
    }//isEmpty
    public int top() {
        return contents[top-1];
    }//top
    public void pop() {
        top--;
    }//pop
    public void push(int value) {
        contents[top++] = value;
    }//push
}//Stack
```

Skriv en klass StackToSetAdapter som implementerar designmönstret *Adapter* för att anpassa gränssnittet för typen Stack till typen Set. För att implementera metoderna member och remove i klassen StackToSetAdapter får du endast använda datastrukturen Stack.

Designmönstret Adapter har följande utseende:



(8 poäng)

Uppgift 10.

Betrakta följande klass:

```
public class SongsDatabase {
    private Map<String, Set<String>> map;

    public SongsDatabase() {
        //skall implementeras i deluppgift a)
    }

    public void addSong(String genre, String songTitle) {
        //skall implementeras i deluppgift b)
    }

    public Set<String> getSongs(String genre) {
        //some code
    }

    public String getGenreOfSong(String songTitle) {
        //skall implementeras i deluppgift c)
    }
}
```

Klassen SongsDatabase används för att hålla reda på sångtitlar genom att klassificera dem efter stilart (t.ex pop, rock, country, etc). Klassen använder sig av en Map, i vilken nyckel-värdeparet utgörs av stilart (som representeras av strängar) och ett Set av sångtitlar (där sångtitlarna representeras av strängar).

a) Implementera konstruktorn som skapar en tom Map.

b) Implementera metoden addSong, som lägger till en sångtitel för angiven stilart. Om stilarten inte redan finns i Map:en skall en ”map entry” för stilarten skapas.

c) Implementera metoden getGenreOfSongs, som returnerar vilken stilart en sångtitel tillhör.

(8 poäng)

Uppgift 11.

a) För att, i ett Java-program, kunna spara ett objekt på en fil och sedan läsa in och återskapa objektet igen, så måste objektet uppfylla en viss grundförutsättning. Vilken?

b) En backup-demon är ett *aktivt objekt* som periodiskt sparar ett annat objekt i en fil. Flera backup-demoner kan vara aktiva samtidigt och spara sina respektive objekt med olika inbördes tidsintervall.

Skriv en klass BackupDeamon som implementerar backup-demoner enligt ovan. Klassen BackupDeamon skall utöka klassen Thread.

Inparametrar till konstruktorn är objektet som skall sparas, namnet på filen där objekt sparas, samt med vilket tidsintervall i sekunder som backuperna tas.

Exempel:

Om följande satser exekveras

```
SomeClass theObject = new SomeClass();
BackupDeamon deamon = new BackupDeamon(theObject, "backupFile", 120);
```

så sparas theObject en gång varannan minut, till filen backupFile.

För att spara ett objekt till en fil får du förutsätta att det finns en färdiga klass

```
public class FileHandler {
    //skall implementeras i deluppgift c)
    public FileHandler(Object object, String backupFile) {
        //skall implementeras i deluppgift c)
    }
    public void save() {
        //skall implementeras i deluppgift c)
    }
}
```

där metoden save utför själva skrivningen av objektet object till filen backupFile.

c) Slutför implementationen av klassen FileHandler.

(10 poäng)

Tentamen 121219- LÖSNINGSFÖRSLAG

Uppgift 1.

a) Ett kompileringsfel inträffar, eftersom metoden bam inte definieras i klassen A.

b) Utskriften blir:
3
class B

c) Utskriften blir:
3
class B
class C

Uppgift 2.

Designen följer inte *Expert pattern*, d.v.s. att det objekt som har informationen för att kunna utföra en uppgift skall utföra uppgiften. I detta fall är det Lamp-objektet som vet om det är tänt eller inte innan det skall tändas!

För att följa Expert pattern görs följande ändringar:

I klassen Button:

```
public void poll() {
      itsLamp.turnOn();
}//poll
```

I klassen Lamp:

```
public void turnOn() {
     if (!on)
         //some code
}//turnOn
```

Uppgift 3.

a) Det skall gälla att

x.equals(y) ⇒ x.hashCode() == y.hashCode()

således är #1, #2 och #4 korrekta implementeringar. Däremot är #3 en felaktig implementation eftersom två instanser c1 och c2 av klassen Circel som har samma värden på x och y kan ha olika värden på radius vilket leder till att c1.equals(c2) är true men c1.hashCode() != c2.hashCode().

b) Implementation #2 skall väljas eftersom denna ger en större spridning hashkoden som beräknas. Ju mer information som används om ett objekts tillstånd vid beräkningen av hashkoden ju större är sannolikheten att olika värden på hashkoden erhåll för på Circle-objekt med olika koordinater.

Uppgift 4.

a)

```
public interface Property {
    public boolean compute(int value);
}
```

b)

```
public static boolean contains(int[] arr, Property obj) {
    for (int i = 0; i < arr.length; i++) {
        if (obj.compute(arr[i]))
            return true;
    }
    return false;
}
```

c)

```
public class Negative implements Property {
    public boolean compute(int value) {
        return value < 0;
    }
}
```

d) Strategy

Uppgift 5.

a)

```
public class Point implements Cloneable {
    // som tidigare
    public Point clone() {
        try {
            return (Point) super.clone();
        }
        catch (CloneNotSupportedException e) {
            throw new InternalError();
        }
    }//clone
}//Point
```

b)

```
public class Polygon implements Cloneable {
    // som tidigare
    public Polygon clone() {
        try {
            Polygon copy = (Polygon) super.clone();
            copy.points = (ArrayList<Point>) points.clone();
            for (int i = 0; i < points.size(); i++)
                copy.points.set(i, points.get(i).clone());
            return copy;
        }
        catch (CloneNotSupportedException e) {
            throw new InternalError();
        }
    }//clone
}//Polygon
```

Uppgift 6.

- getNames - exponerar representationen. En klient kan lägga till eller ta bort element ur listan names.
- getFixedNames - exponerar representationen. En klient kan lägga till eller ta bort element ur listan fixedNames. Att fixedName är deklarerad final betyder att fixedName inte kan ändra värde, referera till en annan lista. Dock kan innehållet i listan förändras.
- getFirstName - exponerar inte representationen eftersom typen String icke-muterbar.
- getFirstPoint - exponerar representationen, eftersom typen Point är muterbar.

Uppgift 7.

a) Specifikation I är starkast, eftersom eftervillkoret är starkare.

b) Specifikation III är starkast, eftersom förvillkoret är svagare.

c) Specifikation III är starkast, eftersom förvillkoret är svagare och eftervillkoret är starkare.

d) Specifikation IV är starkare, eftersom förvillkoret är svagare och eftervillkoret är starkare.

Uppgift 8.

a) Kompileringen går bra, men vid exekveringen blir det ett runtime-fel, eftersom det inte går att lägga ett A-objekt i ett B-fält (Men notera att B[] anses vara en subtyp till A[] under kompileringen).

```
Exception in thread "main" java.lang.ArrayStoreException: A
    at Aarray.update(Aarray.java:3)
    at Aarray.main(Aarray.java:7)
```

b) Här blir det fel redan i kompileringen, eftersom Set<B> INTE anses vara en subtyp till Set<A> under kompileringen.

Uppgift 9.

```
public class StackToSetAdapter implements Set {
    private Stack theStack;
    public StackToSetAdapter(Stack theStack) {
        this.theStack = theStack;
    }//constructor
    public int size() {
        return theStack.size();
    }//size
    public boolean isEmpty() {
        return theStack.isEmpty();
    }//isEmpty
    public boolean member(int x) {
        Stack temp = new Stack(theStack.size());
        boolean found = false;
        while (!found && !theStack.isEmpty()) {
            int elem = theStack.top();
            theStack.pop();
            temp.push(elem);
            if (elem == x)
                found = true;
        }
        while (!temp.isEmpty()) {
            int elem = temp.top();
            temp.pop();
            theStack.push(elem);
        }
        return found;
    }//member
    public void add(int x) {
        theStack.push(x);
    }//add
    public void remove(int x) {
        Stack temp = new Stack(theStack.size());
        boolean found = false;
        while (!found && !theStack.isEmpty()) {
            int elem = theStack.top();
            theStack.pop();
            if (elem != x)
                temp.push(elem);
            else
                found = true;
        }
        while (!temp.isEmpty()) {
            int elem = temp.top();
            temp.pop();
            theStack.push(elem);
        }
    }//remove
}// StackToSetAdapter
```

Uppgift 10.

```
import java.util.*;
public class SongsDatabase {
    private Map<String,Set<String>> map;
    public SongsDatabase() {
        map = new HashMap<String, Set<String>>( );
    }

    public void addSong(String genre, String songTitle) {
        Set<String> g = map.get(genre);              // get set of songs for genre
        if (g == null) {                             // no set exists
            g = new HashSet<String>( );              // create new set
            map.put(genre, g);                       // add to map
        }
        g.add(songTitle);                            // add song to set for genre
    }//addSong

    public String getGenreOfSong(String songTitle) {
        Set<String> myKeySet = map.keySet( );        // set of genre names
        for (String s : myKeySet)  {                         // iterate through genres
            if (map.get(s).contains(songTitle))              // song title in Set<String> for genre
                return s;                            // return genre name
        }
        return null;                                 // song title not found
    }//getGenreOfSong
}//SongsDatabase
```

Uppgift 11.

a) Objektet måste implementera interfacet Serializable.

b)

```
public class BackupDeamon extends Thread {
    private int backupInterval;
    private FileHandler fileHandler;
    public BackupDeamon(Object theObject, String backupFile, int backupInterval) {
        this.backupInterval = backupInterval;
         this.fileHandler = new FileHandler(theObject, backupFile);
        start();
    }
    public void run() {
        while (true) {
            try {
                Thread.sleep( backupInterval*1000 );
                fileHandler.save();
                System.out.println( "Backup taken!" );
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }//run
}//BackupDeamon
```

c)

```
import java.io.*;
public class FileHandler {
    private  Object theObject;
    private String backupFile;
    public FileHandler(Object theObject, String backupFile) {
        this.backupFile = backupFile;
        this. theObject = theObject;
    }
    public void save() {
        try {
            ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(backupFile));
            out.writeObject(theObject);
            out.close();
        }
        catch (IOException e) {
            System.out.println( "Cannot write to: " + backupFile);
        }
    }//saveObject
}//FileHandler
```