Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Ansvarig lärare: Magnus Myréen

DAT050 Objektorienterad programmering

Tentamen, onsdag, 2022-08-24, 14:00-18:00

Vitsordsgränser: 3=24p, 4=36p, 5=48p, max 60p.

Kom ihåg att inte fastna på en uppgift. Bestäm i förväg din egen tidsgräns per uppgift. ***Lycka till!***

Uppgift 1: [5 poäng] *grunder i objektorientering*

Beskriv skillnaden mellan abstrakta klasser (`abstract`) och Javas interface (`interface`). För fulla poäng bör man: förklara vad man kan göra med abstrakta klasser som man inte kan göra med interface; och också förklara vad man kan göra med interface som man inte kan göra med abstrakta klasser. Använd gärna korta kodexempel i svaret.

Uppgift 2: [10 poäng totalt] *private, arv, referensvärden*

Obs: Det finns inga syntax fel i koden i denna uppgift.

A. Accepterar Java kompilatorn följande? Om den gör det, vad skrivs ut när programmet nedan körs? Förklara kortfattat ditt svar. [3 poäng]

```
public class Q2 extends Q1 {
    public int getX() {
        return super.x;
    }
    public static void main(String[] args) {
        Q2 q2 = new Q2();
        System.out.println(q2.getX());
    }
}

public class Q1 {
    private int x = 34;
}
```

B. Accepterar Java kompilatorn följande? Om den gör det, vad skrivs ut när programmet nedan körs? Förklara kortfattat ditt svar. [3 poäng]

```
public class R1 {

    private String str = "Hello";
    public String getStr() {
        return str;
    }
    public static void main(String[] args) {
        System.out.println(getStr());
    }

}
```

C. Accepterar Java kompilatorn följande? Om den gör det, vad skrivs ut när programmet nedan körs? Förklara kortfattat ditt svar. [4 poäng]

```
public class T1 {
    public static void main(String[] args) {
        T2[] array = { null, null, null };
        T2 x = new T2();
        T2 y = new T2();
        array[0] = x;
        array[1] = x;
        array[2] = y;
        array[1].setInt(2);
        System.out.println(array[0].getInt());
        System.out.println(array[1].getInt());
        System.out.println(array[2].getInt());
    }
}

public class T2 {
    private int i = 1;
    public void setInt(int i) { this.i = i; }
    public int getInt() { return i; }
}
```

Uppgift 3: [20 poäng totalt] *att skriva klasser, användning av standard klasser*

A. Din uppgift är att skriva en ***immutable*** klass `IntPair` som uppfyller följande krav:

- har privata `int` instansvariabler som heter `fst` och `snd` [2 poäng]
- har en konstruktor som tar in två `int` parametrar och initialiserar instansvariablerna [2 poäng]
- har `get` metoder för instansvariablerna [2 poäng]
- implementerar en `toString` metod så att kod nedan skriver ut "`(1,2)`" [2 poäng]

```
IntPair ip = new IntPair(1,2);
System.out.print(ip.toString());
```

- implementerar en `equals` metod som följer Javas konventioner [3 poäng]
- överskuggar `hashCode` metoden från Javas `Object` klass med en implementation som följer Javas konventioner [2 poäng]

B. I denna del bör du skriva kod för en ***mutable*** klass som representerar en "oändlig" tvådimensionell spelbräda. Koordinaterna är indexerade av ett par heltal av typen `int` (och därför är spelbrädan inte egentligen oändlig). Metoderna som ska vara med i koden är beskrivna i följande kodskiss. [7 poäng]

```
public class Board2D<A> {

    /** Stores content into cell (x,y) */
    public void setCell(int x, int y, A content) { ... }

    /** Returns previously stored content of cell (x,y)
        Returns null if no content has been stored at (x,y) */
    public A getCell(int x, int y) { ... }

}
```

Tips: En lösning är att representera tillståndet internt som en `HashMap<IntPair,A>`.

Uppgift 4: [10 poäng] *testning, exceptions*

Denna uppgift handlar om att skriva kod som hittar skillnader mellan två metoder, `foo_fast` och `foo_slow`, som bör implementera samma sak. Deras signaturer är:

```
public static int foo_fast(int i) throws Exception { ... }

public static int foo_slow(int i) throws Exception { ... }
```

Koden som kör alla test är följande. Denna kod anropar på en funktion `foo_test`, som du bör implementera.

```
public static void main(String[] args) {
    for (int i = -1000; i < 1000; i++) {
        if (!foo_test(i)) {
            System.out.println("Test failed for input: " + i);
            return;
        }
    }
}
```

Din uppgift är att implementera `foo_test` metoden så att den, givet input `i`, kollar att `foo_fast` och `foo_slow` returnerar samma int, eller att båda kastar något exception för input `i`. Din `foo_test` metod får inte släppa ut exceptions. Om både `foo_fast` och `foo_slow` kastar exception bör testet säga att de har samma beteende oberoende om exceptions som kastas är olika. Din `foo_test` metod bör ha följande signatur.

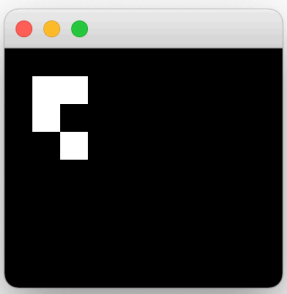
```
public static boolean foo_test(int i) { ... }
```

Uppgift 5: [15 poäng totalt] *grafik, model view controller*

Din uppgift är att implementera en `View` klass som ärver `JPanel`. Denna `View` klass bör rita ut 20x20-pixel stora rutor fyllda med svart eller vitt enligt `whiteCell` metoden i `m` som `View` konstruktorn får som input. Koden nedan bör resultera i ett fönster som ser ut som på bilden.

```
public class Control extends JFrame {
    public Control() {
        Model m = new Model();
        View w = new View(m);
        add(w);
        setSize(200,200);
        setVisible(true);
    }
    public static void main(String[] args) {
        Control f = new Control();
    }
}

public class Model {
    public boolean whiteCell(int x, int y) {
        if (x == 1 && y == 1) { return true; }
        if (x == 2 && y == 1) { return true; }
        if (x == 1 && y == 2) { return true; }
        if (x == 2 && y == 3) { return true; }
        return false;
    }
}
```



Utdrag ur Javas API. **Obs:** Man behöver inte använda alla dessa klasser. Man får också använda annat från Javas API.

Class Color extends Object
- Color(int r, int g, int b)
  Creates an opaque sRGB color with the specified red, green, and blue values in the range (0 - 255).
- BLACK
  The color black.
- WHITE
  The color white.

Class Component extends Object
- void repaint()
  Repaints this component.

Class Graphics extends Object
- void fillOval(int x, int y, int width, int height)
  Fills an oval bounded by the specified rectangle with the current color.
- void fillRect(int x, int y, int width, int height)
  Fills the specified rectangle.
- void setColor(Color c)
  Sets this graphics context's current color to the specified color.

Class HashMap<K,V> extends AbstractMap<K,V> extends Object implements Map<K,V>
- HashMap()
  Constructs an empty HashMap with the default initial capacity (16) and the default load factor (0.75).
- V get(Object key)
  Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- V put(K key, V value)
  Associates the specified value with the specified key in this map.

Class JComponent extends Container extends Component extends Object
- void paintComponent(Graphics g)
  Calls the UI delegate's paint method, if the UI delegate is non-null.

Class JPanel extends JComponent extends Container extends Component extends Object
- JPanel()
  Creates a new JPanel with a double buffer and a flow layout.
- int getHeight()
  Returns the current height of this component.
- int getWidth()
  Returns the current width of this component.
- void paintComponent(Graphics g)
  Calls the UI delegate's paint method, if the UI delegate is non-null.

Class LinkedList<E> extends AbstractCollection<E> extends Object implements List<E>
- LinkedList()
  Constructs an empty list.
- boolean add(E e)
  Appends the specified element to the end of this list.
- E get(int index)
  Returns the element at the specified position in this list.
- int size()
  Returns the number of elements in this list.

Class Pair<K,V> extends Object
- Pair(K key, V value)
  Creates a new pair
- K getKey()
  Gets the key for this pair.
- V getValue()
  Gets the value for this pair.

Class Object
- boolean equals(Object obj)
  Indicates whether some other object is "equal to" this one.
- int hashCode()
  Returns a hash code value for the object.