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

DAT050 Objektorienterad programmering

Tentamen, torsdag, 2022-10-27, 08:30-12:30

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: [10 poäng] *grunder i objektorientering*

Immutable:

A. Vad betyder begreppet immutable i objektorienterad programmering? [2 poäng]
B. Hur programmerar man klasser så att objekten är immutable? [2 poäng]
C. Vilket problem undviker man om man skriver immutable kod? Varför? [2 poäng]

Kohesion och koppling:

D. Förklara kort vad (hög) kohesion och (låg) koppling betyder i samband med objekt-orienterad programmering. [2 poäng]
E. Model-View-Control (MVC) designmönstret hjälper en få hög kohesion och låg koppling. Beskriv kort relationerna mellan Model, View och Control klasserna i MVC. [2 poäng]

Uppgift 2: [9 poäng totalt] *arv, referensvärden*

Vad skrivs ut när följande program körs?

```
public class Main {
    public static void main(String[] args) {
        A a = new A();
        String[] strs = { "abc" };
        a.line1();
        a.line2(new B());
        a.line3(strs);
    }
}

public class A extends B {
    public void test() { System.out.println("test in A"); }
    public void line3(String[] strs) {
        strs[0].toUpperCase();
        System.out.println(strs[0]);
    }
}

public class B {
    public void test() { System.out.println("test in B"); }
    public void line1() { test(); }
    public void line2(Object other) {
        boolean b = (this.getClass() == other.getClass());
        System.out.println(b);
    }
}
```

Uppgift 3: [16 poäng totalt] *att skriva klasser, testning, exceptions*

Denna uppgift handlar om att skriva kod för en klass som representerar en *multimängd* (kallas multiset eller bag på engelska) av heltal (integers). En multimängd av heltal, dvs {1,5,1,1,2}, är lik en matematisk mängd, dvs {1,5,2}, på det sättet att ordningen av elementen inte har någon betydelse. Däremot spelar antalet element roll i en multimängd.

Exempel: Multimängden {1,5,1,1,2} är samma som {1,1,5,2,1}, men inte samma som multimängden {1,5,1,2} för att ett element 1 saknas.

Skriv dina lösningar så att de passar in med kodskissen nedan, som representerar multimängden {1,5,1,1,2} med en `LinkedList` som är [1,5,1,1,2].

```
import java.util.LinkedList;

/** en multimängd för integers */
public class IntBag {

    private LinkedList<Integer> bag = new LinkedList<Integer>();

    /** lägger till heltalet j till denna IntBag */
    public void insert(int j) {
        bag.add(j);
    }

    /** returnerar hur många av heltalet j det finns i denna IntBag */
    public int howMany(int j) {
        ... del A ...
    }

    /** kollar likhet enligt Javas konventioner */
    public boolean equals(Object obj) {
        ... del C ...
    }

}
```

Uppgifter:

A. Implementera metoden `howMany`. När man anropar `howMany(1)` på en `IntBag` som representerar {1,5,1,1,2} då bör den returnera 3 för att 1 finns tre gånger. [4 poäng]

B. Skriv JUnit kod som testar att `IntBag` klarar exemplet som är beskrivet i del A. Se till att test koden inte kraschar ifall ett exception kastas. Använd `assertEquals` mm. [4 poäng]

C. Skriv kod för `equals` metoden så att den returnerar sant ifall två multimängder representerar samma matematiska multimängd, dvs ordningen är inte viktig. [8 poäng]

Tips för del C: Man kan kolla om två multimängder, a och b, är samma genom att kolla om a är en delmängd av b och att b är en delmängd av a. Detta går smidigast om man implementerar en hjälpmetod som kollar om en multimängd är en delmängd av en annan.

```
/** kollar om this är en delmängd av other */
private boolean subset(IntBag other) { ... }
```

En multimängd a är en delmängd av multimängd b ifall alla element i a finns åtminstone lika många gånger i multimängden b.

OBS: Det är viktigt att koden fungerar rätt. Det är däremot inte viktigt att koden är snabb.
OBS: Om din `equals` metod muterar objekten måste du motivera varför det är OK.

Uppgift 4: [10 poäng] *interface, enum, exceptions*

Läs följande definitioner.

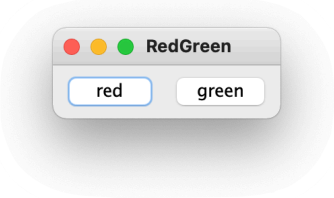
```
public interface CanCompare {

    /** jämför this med other,
        returnerar LESS ifall this är mindre än other,
        kastar aldrig exceptions, returnerar aldrig null */
    public CompareResult compare(CanCompare other);

}

public enum CompareResult { LESS, EQUAL, GREATER; }

public class EmptyException extends Exception {}
```

Din uppgift är att skriva en metod `findLeast` som, givet en array av objekt av typen `CanCompare`, returner ett element från arrayn så att det inte finns något element i array som är strikt mindre (`LESS`) än det elementet som returneras. Om det finns flera element som är lika små, då passar det att man returnerar vilket som helst av de minsta elementen.

Koden för `findLeast` bör kasta ett `EmptyException` exception ifall det inte finns något minsta element. Var noga med `null`. Se till att `EmptyException` är det enda exception som metoden kan kasta oberoende var `null` förekommer i indatan för metoden. Element av arrayn som är `null` bör behandlas helt som om de inte finns i arrayn.

OBS: Se till att signaturen av din metod tar i beaktande att `EmptyException` inte är en *unchecked exception* dvs inte en `RuntimeException`.

Uppgift 5: [15 poäng totalt] *grafiskt användargränssnitt, lyssnare*



Din uppgift är att implementera ett litet grafik program som visar en ruta som på bilden, dvs rutan bör innehålla två knappar, en med texten “red” och den andra med texten “green”. Varje gång man trycker på en av knapparna ska programmet skriva ut *en rad text* som visar *alla knapptryck som skett hittills* i den ordningen de har skett.

Exempel: Om man först trycker `red`, då bör programmet skriva ut raden:

```
red
```

När användaren sedan trycker `green`, då bör programmet skriva ut raden:

```
red green
```

Om användaren sedan trycker `green` igen, bör programmet skriva ut raden:

```
red green green
```

Om användaren sedan trycker `red`, då bör programmet skriva ut raden:

```
red green green red
```

OBS: Positionen av knapparna är inte viktig, men de ska synas.

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

```
Class ActionEvent
  String getActionCommand()
     Returns the command string associated with this action.
Interface ActionListener
  void actionPerformed(ActionEvent e)
     Invoked when an action occurs.
Class Collections extends Object
  static void sort(List<T> list)
     Sorts the specified list into ascending order, according to the natural ordering
     of its elements.
Class Component extends Object
  void repaint()
     Repaints this component.
Class Container extends Component extends Object
  Component add(Component comp)
     Appends the specified component to the end of this container.
Class GridLayout extends Object
  GridLayout(int rows, int cols)
     Creates a grid layout with the specified number of rows and columns.
Class Integer extends Number extends Object
  Integer(int value)
     Constructs a newly allocated Integer object that represents the given int value.
  int intValue()
     Returns the value of this Integer as an int.
Interface Iterator<E>
  boolean hasNext()
     Returns true if the iteration has more elements.
  E next()
     Returns the next element in the iteration.
Class JButton extends AbstractButton extends JComponent extends ... extends Object
  JButton(String text)
     Creates a button with text.
  void addActionListener(ActionListener l)
     Adds an ActionListener to the button.
  void setActionCommand(String actionCommand)
     Sets the action command for this button.
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 JFrame extends Frame extends Container extends Component extends Object
  JFrame(String title)
     Creates a new, initially invisible Frame with the specified title.
  void setLayout(LayoutManager manager)
     Sets the LayoutManager.
Class JPanel extends JComponent extends Container extends Component extends Object
  JPanel()
     Creates a new JPanel with a double buffer and a flow layout.
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.
  Iterator<E> iterator()
     Returns an iterator over the elements in this list in proper sequence.
  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 Object
  boolean equals(Object obj)
     Indicates whether some other object is "equal to" this one.
```