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

Modelsvar för
Tentamen för Objektorienterad programvaruutveckling, TDA545

Måndag, 2015-01-05, 14:00-18:00, byggnad M

Ansvarig lärare:	Magnus Myréen; Alexander Sjösten besöker tentan (076 199 90 55)
Vitsords gränser:	3=28p, 4=38p, 5=48p, max 60p.
Hjälpmedel:	på sista sidan av dokument finns ett utdrag ur Javas API
Resultat:	skickas per epost från Ladok
Lösningar:	kommer att finnas på kurshemsidan
Granskning av rättning:	tider för detta kommer att finnas på kurshemsidan

---

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

Uppgift 1: [4 poäng totalt] *arv, gränssnitt, exceptions*

Svara sant eller falskt på följande påståenden.

a) I Java kan man skriva: `public class A extends B implements I {` [1 poäng]

sant

b) I Java kan man skriva: `public class A extends B, C {` [1 poäng]

falskt

c) I Java kan man skriva: `public class A implements I, J {` [1 poäng]

sant

d) I Java kan man skriva: `public class A throws IllegalArgumentException {` [1 poäng]

falskt

Uppgift 2: [7 poäng totalt] *rekursion, strängar, loopar*

Funktionen `f` är definierad som nedan.

```
public static String f(int k) {
  if (k < 1) {
    return "";
  } else {
    return f(k-1) + k + f(k-1);
  }
}
```

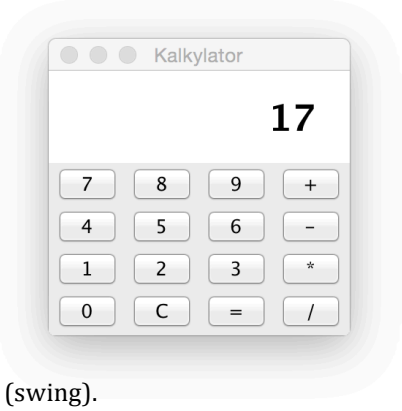
a) Vad skrivs ut när `System.out.println(f(4))` körs? [3 poäng]

`121312141213121`

b) Skriv en ny version av `f` som inte använder rekursion. Den nya version bör utföra precis samma beräkning som koden för `f` ovan. [4 poäng]

```
public static String f(int k) {
  String res = "";
  for (int i=1; i<=k; i++) {
    res = res + i + res;
  }
  return res;
}
```

Uppgift 3: [18 poäng] *gränssnitt, att skriva klass*



Denna uppgift handlar om att implementera logik delen av en mycket enkel kalkylator.

```
public interface Calc {
  public void pressKey(String key);
  public String getDisplayText();
}
```

Skriv kod för en klass som implementerar (implements) gränsnittet (interface) `Calc` ovan. [18 poäng]

Obs. Denna uppgift handlar *inte* om grafiska användargränssnitt (swing).

Kalkylator programmet är organiserat så att när användaren trycker på en av knapparna skickas texten som står på knappen till din implementation av gränssnittet `Calc`. Exempel: om man trycker `C` (dvs knappen som bör återställa kalkylatorn) då anropas `pressKey` metoden med indata `"C"`; sen anropar programmet `getDisplayText` metoden för att få texten som ska visas till användaren. (I bilden ovan har `getDisplayText` returnerat 17).

Se till att din kod klarar av enkla test som t.ex.

```
public boolean calcTest1(Calc c) {
  c.pressKey("C");
  c.pressKey("1");
  c.pressKey("2");
  c.pressKey("+");
  c.pressKey("5");
  c.pressKey("=");
  return (c.getDisplayText()).equals("17");
}

public boolean calcTest2(Calc c) {
  c.pressKey("C");
  c.pressKey("1");
  c.pressKey("2");
  c.pressKey("+");
  c.pressKey("5");
  c.pressKey("C");
  return (c.getDisplayText()).equals("0");
}

public boolean calcTest3(Calc c) {
  c.pressKey("C");
  c.pressKey("1");
  c.pressKey("+");
  c.pressKey("2");
  c.pressKey("*");
  c.pressKey("3");
  c.pressKey("=");
  return (c.getDisplayText()).equals("9");
}
```

Tips: Innan du skriver kod, tänk hur en enkel kalkylator fungerar. Vad händer när man trycker på "+" knappen? Ja, kalkylatorn kommer ju ihåg vad den skall göra när man senare trycker på "=" knappen. Men en enkel kalkylator kommer inte ihåg mera än det.

Obs. Ni behöver inte hantera division med 0.

```
public class CalcImpl implements Calc {

  int display = 0;
  int memory = 0;
  String op_key = "+";

  public void computeResult() {
    if (op_key.equals("+")) {
      memory = memory + display;
    } else if (op_key.equals("-")) {
      memory = memory - display;
    } else if (op_key.equals("*")) {
      memory = memory * display;
    } else if (op_key.equals("/")) {
      memory = memory / display;
    }
  }

  public void pressKey(String key) {
    if (key.equals("0") || key.equals("1") || key.equals("2") ||
        key.equals("3") || key.equals("4") || key.equals("5") ||
        key.equals("6") || key.equals("7") || key.equals("8") ||
        key.equals("9")) {
      display = display * 10 + Integer.parseInt(key);
    } else if (key.equals("+") || key.equals("-") ||
               key.equals("*") || key.equals("/")) {
      computeResult();
      op_key = key;
      display = 0;
    } else if (key.equals("=")) {
      computeResult();
      op_key = "+";
      display = memory;
      memory = 0;
    } else if (key.equals("C")) {
      op_key = "+";
      display = 0;
      memory = 0;
    }
  }

  public String getDisplayText() {
    return display + "";
  }

}
```

Uppgift 4: [16 poäng totalt] *testning, array, slumptal, loopar*

Denna uppgift ber dig skriva testkod för en sorteringsfunktion.

Sorteringsfunktionen, `petersSort`, tar in en array av `int` och arrangera om elementen i arrayn så att `a[i] <= a[i+1]`, för varje index `i`.

```
for (int i=0; i<1000; i++) {
  int[] a = generateRandomArray();
  petersSort(a);
  if (isInOrder(a)) {
    System.out.println("OK");
  } else {
    System.out.println("Fel!");
  }
}
```

Din uppgift:

a) Implementera funktionen `generateRandomArray`, så att den returnerar en array som är slumpmässig både i längd och innehål. [4 poäng]

```
public int[] generateRandomArray() {
  Random r = new Random();
  int[] a = new int[r.nextInt(10000)];
  for (int i=0; i<a.length; i++) {
    a[i] = r.nextInt();
  }
  return a;
}
```

b) Implementera funktionen `isInOrder`, som returnerar `true` endast ifall alla element är i rätt ordning [4 poäng]

```
public boolean isInOrder(int[] a) {
  for (int i=0; i<a.length-1; i++) {
    if (a[i] > a[i+1]) { return false; }
  }
  return true;
}
```

c) Förklara varför följande felaktiga implementation av `petersSort` inte kommer att fångas av testkoden ovan. [3 poäng]

```
public static void petersSort(int[] a) {
  for (int i=0; i<a.length; i++) {
    a[i] = i;
  }
}
```

Test koden testar bara att elementen i arrayn är i rätt ordning. Den testar inte om elementen är en om ordning av orginal arrayn.

d) Beskriv (men inte nödvändigtvis med konkret kod) hur testkoden kan förbättras så att felaktiga implementationer som ovan inte kommer igenom testet. [5 poäng]

Test koden kan förbättras genom att ta en kopia av arrayn innan den går in i `petersSort`. Efter att `petersSort` har kört kan man kolla om elementen stämmer med orginal arrayn. Ett bra sätt att göra detta på är att köra en enkel och säkert korrekt sortering implementation på kopian på orginal arrayn, och sedan jämföra resultaten. Här är exempel kod:

```
for (int i=0; i<1000; i++) {
  int[] a = generateRandomArray();
  int[] b = copyOfArray(a);
  petersSort(a);
  bubbleSort(b);
  boolean correct = true;
```

```
    for (int j=0; j<a.length; j++) {
      if (a[j] != b[j]) { correct = false; }
    }
    if (correct) {
      System.out.println("OK");
    } else {
      System.out.println("Fel!");
    }
  }
```

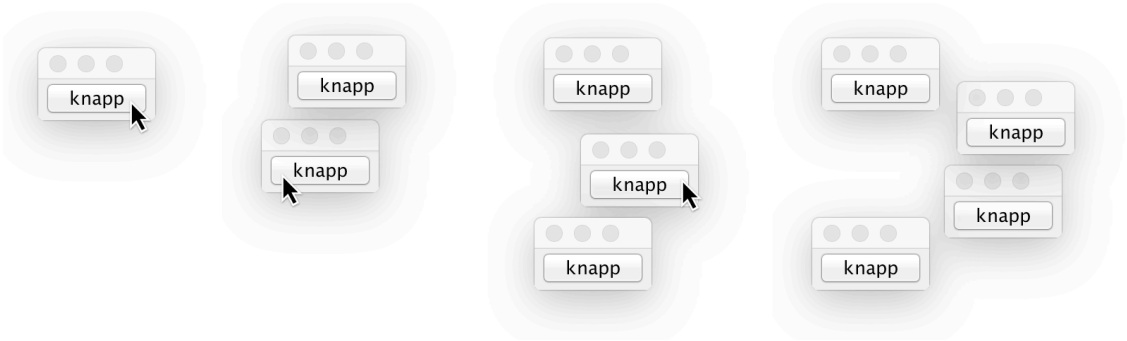
Uppgift 5: [15 poäng] *swing, händelsehantering*

Skriv ett program som består av fönster. I varje fönster finns det endast en knapp. När man trycker på knappen försvinner fönstret med knappen som tryckts och samtidigt skapas två likadana fönster, som har samma beteende. När programmet startas bör det endast finnas ett fönster.

Obs. Det viktiga är att få ovanstående beteende implementerat korrekt. Fönstrens positionen och storleken m.m. är inte viktigt.

Exempel: När programmet körs bör det se ut ungefär så här:

När program-met startar:	Efter ett knapptryck:	Efter ett till knapptryck:	Efter det tredje knapptrycket:




```
public class Click extends JFrame implements ActionListener {

  public static void main(String[] args) {
    Click c = new Click();
  }

  public Click() {
    JPanel p = new JPanel();
    JButton b = new JButton("Knapp");
    b.addActionListener(this);
    p.add(b);
    this.add(p);
    this.setVisible(true);
  }

  public void actionPerformed(ActionEvent e) {
    Click c1 = new Click();
    Click c2 = new Click();
    this.setVisible(false);
  }

}
```

Class AbstractButton extends JComponent extends Container extends Component extends Object

`public void addActionListener(ActionListener l)`
Adds an ActionListener to the button.

Interface ActionListener

`void actionPerformed(ActionEvent e)`
Invoked when an action occurs.

Class ActionEvent extends AWTEvent extends EventObject extends Object

`ActionEvent(Object source, int id, String command)`
Constructs an ActionEvent object.
`String getActionCommand()`
Returns the command string associated with this action.

Class String extends Object

`int length()`
Returns the length of this string.

Class Component extends Object

`int getHeight()`
Returns the current height of this component.
`int getWidth()`
Returns the current width of this component.

Class Container extends Component extends Object

`Component add(Component comp)`
Appends the specified component to the end of this container.

Class FlowLayout extends Object, implements LayoutManager

`FlowLayout()`
Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.

Class Graphics extends Object

`void drawRect(int x, int y, int width, int height)`
Draws the outline of the specified rectangle.
`abstract void fillRect(int x, int y, int width, int height)`
Fills the specified rectangle.

Class JButton extends AbstractButton extends JComponent extends Container extends Component …

`JButton(String text)`
Creates a button with text.

Class JComponent extends Container extends Component extends Object

`protected void paintComponent(Graphics g)`
Calls the UI delegate's paint method, if the UI delegate is non-null.

Class JFrame extends Frame extends Window extends Container extends Component extends Object

`Container getContentPane()`
Returns the contentPane object for this frame.

Class JPanel extends JComponent extends Container extends Component extends Object

`JPanel()`
Creates a new JPanel with a double buffer and a flow layout.

Interface LayoutManager

Class Object

`boolean equals(Object obj)`
Indicates whether some other object is "equal to" this one.
`String toString()`
Returns a string representation of the object.

Class Random extends Object

`Random()`
Creates a new random number generator.
`int nextInt()`
Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence.
`int nextInt(int n)`
Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Class Window extends Container extends Component extends Object

`void setVisible(boolean b)`
Shows or hides this Window depending on the value of parameter b.