University of Cape Town

Department of Computer Science

Computer Science CSC1016S

Class Test 2

Wednesday 27 September 2006

Marks: 40

Time: 40 minutes

- Approximate marks per question are shown in brackets
- The use of calculators is permitted

Surname Initials

NAME:

STUDENT NO: **COURSE CODE:** CSC

This paper consists of 5 questions and 8 pages (including this cover page).

Mark Allocation							
Question	**Marks**	**Internal**	**External**	**Quest**	**Marks**	**Internal**	**External**
1	[5]			4	[10]		
2	[10]			5	[11]		
3	[4]						
	Total				**Total**		
					Grand Total		
					Final Mark		
Internal Examiner:				**External Examiner:**			

Question 1 [5 marks]

a) What interface should you implement if you want objects of the class to be copied in the way that Java recommends?

Cloneable

[1]

b) What method must you then override?

clone

[1]

c) From which class is that method inherited?

Object

[1]

d) What interface should a class implement if objects of it are going to be compared and ordered?

Comparable

[1]

e) What method must such a class define?

compareTo

[1]

Question 2 [10 marks]

Consider the definition of a data structure, ArrayQ, given below. ArrayQ contains positive integers. An item is added to the back and removed from the front only. We keep track of the number of items in the `ArrayQ` by updating the integer `n`. For example, `add(item)` puts `item` in location `n and` increments `n`. An empty `ArrayQ` is indicated by `n==0`.

```
public class ArrayQ
{
   private int[] array;
   private int n;
```

```
  public ArrayQ()
  {
    array = new int[100];
    n = 0;
  }

  public void add(int item)
  {// add only positive intgers
   // increment n
   // if item is negative do not add it
  }

  public int remove()
  {// remove only if queue not empty
   // decrement n
   // move all othe items to fill gap
   // otherwise return -1
  }

  public void output()
  {
    for(int i = 0; i < n; i++)
      System.out.println(array[i]);
  }

  public static void main(String[] args)
  {
    ArrayQ aq = new ArrayQ();
    aq.add(9);
    aq.add(5);
    aq.add(10);
    aq.add(-7);
    aq.add(4);
    System.out.println("The queue contains:");
    aq.output();
    System.out.println("Removed item from the front: "
              + aq.remove());
    aq.output();
  }

}
```

The `main` method produces the following output:

```
The queue contains:
9
5
10
4
Removed item from the front: 9
5
10
4
```

a) Complete the `add` method. Remember items are added to the back only and `n` is then incremented. Negative items are not added.

[3]

b) Complete the `remove` method. Items are removed from the front only and `n` is then decremented. All other items must be moved forward one step to fill the gap. If the queue is empty, the `remove` method returns -1. (Hint: fill gap by putting item 1 in location 0, item 2 in location 1, item i in location i-1)

[7]

Solution:

```
public class ArrayQ
{
  private int[] array;
  private int n;

  public ArrayQ()
  {
    array = new int[100];
    n = 0;
  }

  public void add(int item)
  {
    if(item>=0)
    {
      array[n] = item;
      n = n + 1;
    }
  }

  public int remove()
  {
    int front = array[0];
    if (n==0)
      return -1;
    for (int i = 0; i < n - 1; i++)
    {
      array[i] = array[i + 1];
    }
    n = n - 1;
    return front;
  }

  public void output()
  {
    for(int i = 0; i < n; i++)
      System.out.println(array[i]);
  }

  public static void main(String[] args)
  {
    ArrayQ aq = new ArrayQ();
    aq.add(9);
    aq.add(5);
    aq.add(10);
    aq.add(-7);
    aq.add(4);
    System.out.println("The ArrayQ contains:");
    aq.output();
```

```
      System.out.println("Removed item from the front: "
                + aq.remove());
      aq.output();
   }

}
```

Question 3 [4 marks]

Consider the interfaces A and B defined below.

```
public interface A
{
   public static final int NUMBER = 4;
   public String getNumber(); // returns NUMBER in String form
}

public interface B
{
}
```

a) Fill in interface B so that a class implementing both A and B will have **two** possible sources of conflict.

[2]

```
public interface B
{
   pulbic static final int NUMBER = 11;
   public int getNumber();
}
```

b) Consider the following program which uses the getNumber method from interface A above.

```
public class Program<T>
{
   T data;

   public String getData()
   {
      return data.getNumber();
   }
}
```

What requirement do we need to place on the types plugged in for T?

They must implement the A interface.

[1]

c) What is the code for specifying this requirement in the definition of Program?

Public class Program <T extends A>

[1]

Question 4 [10 marks]

a) What are **preorder**, **postorder**, and **inorder** processing of a binary tree?
Solution:

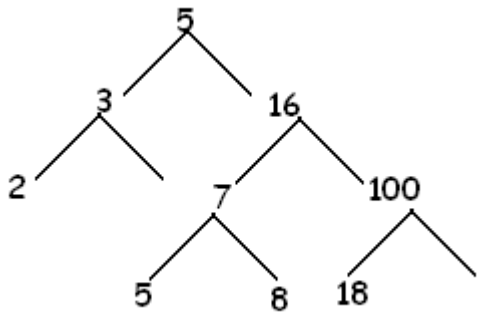
Preorder: root, left, right
Postorder: left, right, root
Inorder: left, root, right

[5]

b) Use the **binary search tree storage rule** to build a binary tree containing the following data. Start with the leftmost number, i.e., 5, then 16, and so on.

5, 16, 3, 7, 8, 100, 2, 5, 18.

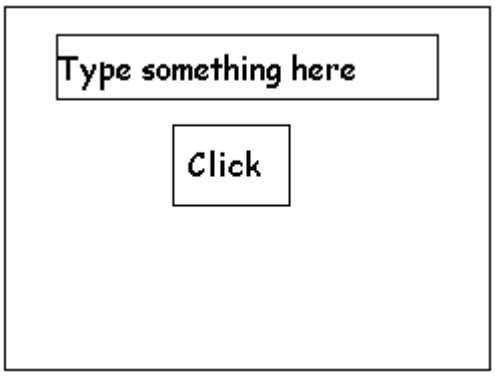
Solution



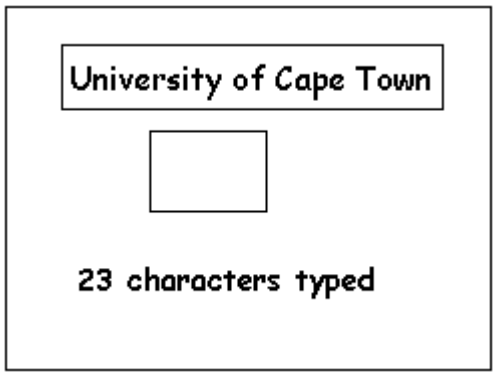
[5]

Question 5 [11 marks]

Complete the class, GuiDemo, to produce the following GUI. It consists of a frame containing a text field and a button. Use three panels and a GridLayout manager to arrange elements approximately as shown. The text field initially contains the words "Type something here" and the word "Click" is on the button. When the button is clicked the word "Click" disappears and "N characters typed" appears below the button as shown, where N is the number of characters in the text typed into the text field.



Initial GUI



After entering text and clicking button

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GuiDemo extends JFrame implements ActionListener
{
   public static final int WIDTH = 300;
   public static final int HEIGHT = 200;

   private JTextField textField;
   private JButton button;
   private JLabel label;
   private JPanel topPanel, midPanel, bottomPanel;

   public static void main(String[] args)
   {
      GuiDemo gui = new GuiDemo( );
      gui.setVisible(true);
   }

   public GuiDemo( )
   {
      super("Panel Demonstration");
      setSize(WIDTH, HEIGHT);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      setLayout(new GridLayout(3,1));
      topPanel = new JPanel();
      midPanel = new JPanel();
      bottomPanel = new JPanel();

      // add panels to frame
      // create and add components to panels
   }

   public void actionPerformed(ActionEvent e)
   {
   // code response to button click
   }
}
```

a) Complete the constructor [6]
b) Complete the actionPerformed method. [5]

Solution

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GuiDemo extends JFrame implements ActionListener
{
  public static final int WIDTH = 300;
  public static final int HEIGHT = 200;

  private JTextField textField;
  private JButton button;
  private JLabel label;
  private JPanel topPanel, midPanel, bottomPanel;

  public static void main(String[] args)
  {
    GuiDemo gui = new GuiDemo( );
    gui.setVisible(true);
  }

  public GuiDemo( )
  {
    super("Panel Demonstration");
    setSize(WIDTH, HEIGHT);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setLayout(new GridLayout(3,1));
    topPanel = new JPanel();
    midPanel = new JPanel();
    bottomPanel = new JPanel();
    add(topPanel);
    add(midPanel);
    add(bottomPanel);

    textField = new JTextField("Type something here");
    topPanel.add(textField);
    button = new JButton("Click");
    button.addActionListener(this);
    midPanel.add(button);
    label = new JLabel();
    bottomPanel.add(label);

  }

  public void actionPerformed(ActionEvent e)
  {
    button.setText("     ");
    int n = textField.getText().length();
    label.setText(n + " characters typed.");
  }
}
```