Question 1

At **Victoria University of Technology**, all academic staff members are of three categories: tenured, contract and sessional.

- All employees have an identification number, name and job title.
- Tenured employees have to contribute 7% of their gross pay towards superannuation that is to be deducted from their gross pay.
- Contract employees have the terms of the contract in number of months (e.g. 24 months).
- Sessional employees are paid according to the number of hours worked and hourly rate of pay.

For simplicity, we will ignore the employee's starting date.
The class `Employee` extracts the commonality from the three kinds of employees:

```
class Employee {
   protected int id;
   protected String name;
   protected String jobTitle;
   public Employee (int id, String name, String title) {
         this.id=id;
         this.name = name;
         this.jobTitle = title;
   }
   public String getID() { return id; }
   public String getName() { return name; }
   public String getTitle() { return jobTitle; }
   public double computePay() { return 0.0; }
   public String toString() {
         return( "Employee Detail: " + getID() + "\t" +getName() +
         + "\t" + getTitle() );
   }
}
```

Write the constructors, any other appropriate methods and a `toString()` method for each class below:

a) A class `PermEmployee` for tenured employees.
b) A class `ContractEmployee` for contract employees.
c) A class `SessionEmployee` for sessional employees.
d) Write an application to create an array of employees containing the following objects:

```
Melinda is a tenured senior lecturer and has a gross pay of $3000
per fortnight
Joe is a lecturer on a 12-month contract and is paid $2100 per
fortnight
David is a tutor who worked 16 hours per fortnight and is paid $60
per hour
```

Print the records of the above employees using an iteration technique to display each employee's detail as well as the net pay each received per fortnight. (20 marks)

Question 2

```
interface Product {
   static final String COMPANY = "Noahs Ark";
   public int getPrice();
}//interface
```

```
abstract class Animal {
   protected String type;
   protected String name;

   public Animal (String aType, String aName) {
         type = aType;
         name = aName;
   }
   public abstract void setType (String aType);
   public abstract void setName (String aName);
} //Animal
```

Use the above information to implement a `Pet` class, which extends the `Animal` class and implements the `Product` interface. Write an application to create a dog called `Scooby`, which costs $500 and print out the data related to the pet `Scooby`. (10 marks)

Question 3

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class TestButton extends Applet implements ActionListener {
   private TextField t1 = new TextField(20);
   private TextField t2 = new TextField(20);
   private Button b = new Button("*”);
   private TextArea log = new TextArea (5,20);
   public void init() {
         t1.setEditable(true);
         add(t1);
         t2.setEditable(true);
         add(t2);
         b.addActionListener(this);
         add(b);
         add(log);
   }
   public void actionPerformed (ActionEvent e) {
         //Code to be supplied by you
   }
}
```

a) The code above involves a button that does nothing. Modify the code so that a button click causes whatever appears in the `TextField t1` to be multiplied by `TextField t2`. Display the result in a text area that serves as an ongoing log of all multiplications.

b) Rewrite the `TestButton` class using two classes instead of one. Modify the class `TestButton` and include another class `ButtonHandler` to be the handler class that the event will be delegated to. (15 marks)

Question 4

The file `JointAcctTest` defines the classes `JointAcctTest`, `JointAcct` and `Person`. The `main()` in `JointAcctTest` creates an instance of `JointAcct` which is passed as an argument in the creation of an instance of `Person`. The instance of `JointAcct` is sent the message `drawMoney()` as defined below.

```
public class JointAcctTest {
   public static void main (String[] args) {
         JointAcct jAcct = new JointAcct();
         jAcct.setBalance(500);
         Person p = new Person(jAcct);
         p.start();
         jAcct.printBalance();
   }
}

class JointAcct {
   private int balance;
   public JointAcct() {balance=0;}
   public int getBalance() { return balance; }
   public void setBalance(int i) { balance=i; }


   public void drawMoney(int amt) {
         if (balance() > amt) {
               try { sleep(1000); } catch (Exception e) {;}


               balance -= amt;
         else ; //insufficient funds
         }
   }
   public void printBalance() {
         System.out.println("Balance is "  + balance);
   }
}

class Person extends Thread {
   private JointAcct acct;
   Person(JointAcct jA) { acct=jA; }
   public void run() {
         acct.drawMoney(300);

   }
}
```

a) Describe two ways to create a thread. Rewrite the class `Person` using another way of creating a thread instead of inheritance and modify the necessary statement in the class `JointAcctTest`.

b) Under what circumstances do threads need to be synchronized? When used in a method heading, what does the keyword `synchronized` mean?

c) Modify the above code by creating two instances of `Person` sharing an instance of `JointAcct`. Introduce synchronization into the class `JointAcct` and send the appropriate message to the object upon which it has a lock so that it is placed in a queue with that object. Ensure that the output of the account balance is always correct.

(15 marks)

Question 5

Suppose we have a file called **STUDENT.DAT** that has the following format:

```
Alice      19    67
Kim        14    59
John       20    76
May        12    50
Paul       15    66
Angela     11    44
...
```

The first column is the student's name, the second column is the student's assignment mark out of 20 and the third column is the student's final examination mark out of 80. Write a program to read this file and display the name and the corresponding total mark for each student to an output file called **RESULT.DAT**. Also display the student that has the highest total score to the screen. Include appropriate file exception handling. (20 marks)

Question 6

a) Provide a brief answer for each of the following.

- **i)** How do you create a server socket? What port number can be used? What happens if a requested socket number is already in use? Can a port connect to multiple clients?
- **ii)** How does a client program initiate a connection?
- **iii)** How does a server accept a connection?

b) The following is an example of a server program and a client program. On the client side, a text field is used to enter the radius of a circle and the client sends the radius to the server and receives the area of the circle from the server. On the server side, a message is displayed in the text area when the server receives a message from a client. Assuming that the client and the server are running on the same machine, a sample run of the programs are shown in figure 1 and figure 2. Given the complete implementation of the server program, complete the program by writing the missing code in the client program.

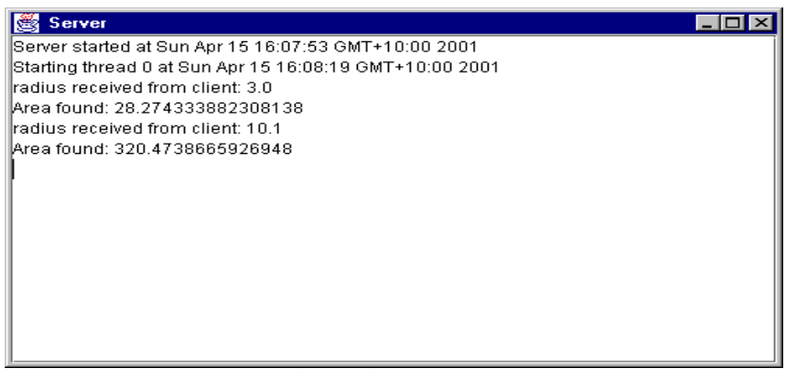

figure 1

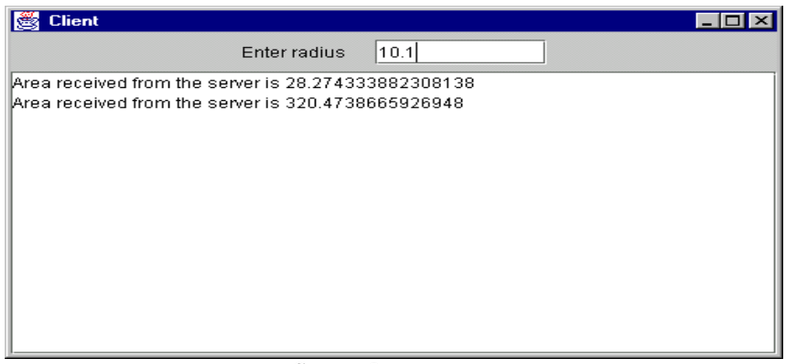

figure 2

```
// Server.java:  Question 5
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Server extends JFrame
{ private JTextArea jta = new JTextArea();

  public static void main(String[] args)
  {    new Server();
  }
  public Server()
  {
    // Place text area on the frame
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(new JScrollPane(jta), BorderLayout.CENTER);

    // It is necessary to show the frame here!
    setTitle(" Server");
    setSize(500, 300);
    setVisible(true);
    try
    {
      // Create a server socket
      ServerSocket s = new ServerSocket(8000);
      jta.append("Server started at " + new Date() + '\n');

      // Label a thread
      int i = 0;

      while (true)
      {
        // Listen for a new connection request
        Socket connectToClient = s.accept();

        // Print the new connect number on the console
        jta.append("Starting thread "+i+" at "+new Date()+'\n');

        // Create a new thread for the connection
        ThreadHandler  t = new ThreadHandler(connectToClient, i);

        // Start the new thread
        t.start();
```

```
        // Increment i to label the next connection
        i++;
      }
    }
    catch(IOException ex)
    {
      System.err.println(ex);
    }
  }

// Inner class
// Define the thread class for handling new connection
class ThreadHandler extends Thread
{
  private Socket connectToClient; // A connected socket
  private int counter; // Label for the connection

  public ThreadHandler(Socket c, int i)
  {
    connectToClient = c;
    counter = i;
  }

  public void run()
  {
    try
    {
      // Create data input and print streams
      BufferedReader isFromClient = new BufferedReader(
      new InputStreamReader(connectToClient.getInputStream()));
      PrintWriter osToClient =
        new PrintWriter(connectToClient.getOutputStream(), true);

      // Continuously serve the client
      while (true)
      {
        // Receive data from the client in string
        StringTokenizer st = new StringTokenizer
          (isFromClient.readLine());

        // Get radius
        double radius = new Double (st.nextToken ()).doubleValue ();
        jta.append("radius received from client: "+radius+'\n');

        // Compute area
        double area = radius*radius*Math.PI;

        // Send area back to the client
        osToClient.println(area);
        jta.append("Area found: "+area+'\n');
      }
    }
    catch(IOException e)
    {  System.err.println(e);
    }
  }
}
}
```

```
// Client.java:  Question 5
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Client extends JFrame  implements ActionListener
{
  private JTextField jtf;
  private JTextArea jta = new JTextArea();
  private JButton jbtStart, jbtStop;
  PrintWriter osToServer;
  BufferedReader isFromServer;

  public static void main(String[] args)
  {
    Client client=new Client();
    client.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent event)
      {  System.exit(0);
      }
    });
  }

  public Client()
  {
    Panel p1 = new Panel();
    p1.add(new Label("Enter radius"));
    p1.add(jtf = new JTextField(10));

    // It is necessary to show the frame here!
    setTitle(" Client");
    setSize(500,300);

    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(p1, BorderLayout.NORTH);
    getContentPane().add(new JScrollPane(jta), BorderLayout.CENTER);

    jtf.addActionListener(this);
    setVisible(true);

    try
    {
      // Create a socket to connect to the server
      // and other missing code to be written by you
      .
      .
      .

    }
    catch (IOException ex)
    {
      jta.append(ex.toString()+'\n');
    }
  }
```

```
  public void actionPerformed(ActionEvent e)
  {
    String actionCommand = e.getActionCommand();
    if (e.getSource() instanceof JTextField)
    {
      try
      {
        // write the missing code to read the radius,
        // send to server and obtain area from the server
        .
        .
        .
        // Print area on
        jta.append("Area received from the server is "
          +area + '\n');
      }
      catch (IOException ex)
      { System.err.println(ex);
      }
    }
  }
}
```

(20 marks)