SEMESTER ONE EXAMINATIONS 2006
SOLUTIONS

MODULE: **Object-Oriented Programming for Engineers - EE553**

COURSE: **M.Eng./Grad. Dip./Grad. Cert. in Electronic Systems**
M.Eng./Grad. Dip./Grad. Cert. in Telecomms. Eng.
RAEC – Remote Access to Continuing Eng. Education

1(a i) Passing an object by constant reference allows us to pass a large block of data that cannot be modified without the need to copy the contents into a temporary variable, like in pass by value. The method/function cannot modify the contents of the memory passed by constant reference.

1(a ii) The closest Java equivalent to the C++ member initialisation list is the super() call that can be made from a Java constructor. Importantly this call must be made as the first call within the constructor as it calls the parent constructor – In C++ and Java the parent constructor is called before the child constructor.

1(a iii) The closest equivalent to friend methods and classes are package classes with static methods. The “fiendly” access specifier is the default access specifier in Java and allows this type of relationship.

1(a iv) The system calls the GC when memory is low. The system calls the GC when the CPU is idle. The user can call the GC directly, but since the GC runs as a thread in a threaded environment, there is no guarantee that it will run immediately.

1(a v) C++ uses =0, Java uses the abstract keyword. In Java an abstract class can have no abstract methods – this cannot occur in C++.

1(a vi) C++ has a structure that allows us to have different types of data within the same variable. A union is particularly memory efficient as it calculates the minimum amount of memory to store the largest element in the structure. Therefore, contained data overlaps in memory.

1(a vii) Conditional operator ?. It has the form conditionalExpression ? trueExpression1 : falseExpression2, for example:

```
cout << (s.length() < t.length() ? s : t);  // Display the shorter of s and t
```

1(a) **[14 marks total – 2 for each part i to vii]**

1(b) In the development of large C++ applications that involve several different programmers, two programmers could use the same name for a global variable or class, representing related concepts. This would cause complications as such conflicts can have unpredictable results (more likely compiler errors). So, while individual code segments would work independently, when they are brought together errors may result. The namespace concept was introduced during the standardisation of C++, to allow the programmer to define a namespace that is limited in scope. When writing C++ applications we can make it explicit that we are using the standard namespace by a "using directive": **using namespace std;**

```
#include<iostream>

 namespace MolloySpace
 {
  float testAdd(float a, float b) { return a+b; }

  class Time
```

```
  {
            public: //etc.
  }
}

using std::cout;
using std::endl;

using namespace MolloySpace;
//using MolloySpace::Time;
```

Placing such a using directive at the start of every file is the same as placing all definitions in the global namespace - exactly what namespaces were invented to avoid.

[7 marks – 2 for desc, 3 for code, 2 for last part]

1(c)

```
public class TestParent
{
  protected int y=2; // example parent state

  public TestParent(int a) // example parent constructor
  {
    y=a;
  }
}

public class Test extends TestParent // inheritance
{
  private int x;

  public Test(int a)
  {
    super(a); // calls the parent constructor
    x = 5;
  }

  public void someMethod(int x)
  {
    this.x = x;     // this.x refers to the state of the class
    super.y = 5;        // sets the y state of the parent class to a //value of 5.

    someOtherMethod(this); // passes the current object to the
                        // someOtherMethod() method
  }
}
```

[4 marks, 1 for each this (x2), 1 for each super (x2)]

2(a) (Code in italics provided – non-italic code must be added)

```
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.text.*;
import java.awt.event.*;

/**
```

```
 * A Clock Applet - no pressure! ;-)
 * Question 3
*/

public class Clock extends Applet implements Runnable, MouseListener {
   private volatile Thread timer;
   private int lastxs, lastys, lastxm,
           lastym, lastxh, lastyh;
   private SimpleDateFormat formatter;     // Formats the date displayed
   private Date currentDate;               // Used to get date to display
   private Color handColor;                // Color of main hands and dial
   private Color numberColor;              // Color of second hand and numbers
   private int xcenter = 80, ycenter = 55; // Center position
   private boolean pauseClock = false;

   public void init() {
      int x,y;
      lastxs = lastys = lastxm = lastym = lastxh = lastyh = 0;
      formatter = new SimpleDateFormat ("EEE MMM dd hh:mm:ss yyyy",
Locale.getDefault());
      handColor = Color.blue;
      numberColor = Color.darkGray;
                  this.addMouseListener(this);
      resize(300,300);
   }

   public void paint(Graphics g) {

                  // Draw the circle and numbers
                  g.setColor(handColor);
                  g.drawArc(xcenter-50, ycenter-50, 100, 100, 0, 360);
                  g.setColor(numberColor);
                  g.drawString("9", xcenter-45, ycenter+3);
                  g.drawString("3", xcenter+40, ycenter+3);
                  g.drawString("12", xcenter-5, ycenter-37);
                  g.drawString("6", xcenter-3, ycenter+45);

                  int xh, yh, xm, ym, xs, ys;
                  int s = 0, m = 10, h = 10;
                  String today;
                  if (!pauseClock) currentDate = new Date();

                  formatter.applyPattern("s");
                  try
                  {
                           s = Integer.parseInt(formatter.format(currentDate));
                  }
                  catch (NumberFormatException n)
                  {
                           s = 0;
                  }
                  formatter.applyPattern("m");
                  try
                  {
                           m = Integer.parseInt(formatter.format(currentDate));
                  }
                  catch (NumberFormatException n)
                  {
                           m = 10;
                  }
```

```
                formatter.applyPattern("h");
                try
                {
                        h = Integer.parseInt(formatter.format(currentDate));
                }
                catch (NumberFormatException n)
                {
                        h = 10;
                }

                // Set position of the ends of the hands
                xs = (int) (Math.cos(s * Math.PI / 30 - Math.PI / 2) * 45 + xcenter);
                ys = (int) (Math.sin(s * Math.PI / 30 - Math.PI / 2) * 45 + ycenter);
                xm = (int) (Math.cos(m * Math.PI / 30 - Math.PI / 2) * 40 + xcenter);
                ym = (int) (Math.sin(m * Math.PI / 30 - Math.PI / 2) * 40 + ycenter);
                xh = (int) (Math.cos((h*30 + m / 2) * Math.PI / 180 - Math.PI / 2) * 30 +
xcenter);
                yh = (int) (Math.sin((h*30 + m / 2) * Math.PI / 180 - Math.PI / 2) * 30 +
ycenter);

                // Erase if necessary
                g.setColor(getBackground());
                if (xs != lastxs || ys != lastys)
                {
                        g.drawLine(xcenter, ycenter, lastxs, lastys);
                }
                if (xm != lastxm || ym != lastym)
                {
                        g.drawLine(xcenter, ycenter-1, lastxm, lastym);
                        g.drawLine(xcenter-1, ycenter, lastxm, lastym);
                }
                if (xh != lastxh || yh != lastyh)
                {
                        g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
                        g.drawLine(xcenter-1, ycenter, lastxh, lastyh);
                }
                g.setColor(numberColor);
                g.drawLine(xcenter, ycenter, xs, ys);
                g.setColor(handColor);
                g.drawLine(xcenter, ycenter-1, xm, ym);
                g.drawLine(xcenter-1, ycenter, xm, ym);
                g.drawLine(xcenter, ycenter-1, xh, yh);
                g.drawLine(xcenter-1, ycenter, xh, yh);
                lastxs = xs; lastys = ys;
                lastxm = xm; lastym = ym;
                lastxh = xh; lastyh = yh;
        }

          public void start() {
            timer = new Thread(this);
            timer.start();
          }

          public void stop() {
            timer = null;
          }

          public void run() {
            Thread me = Thread.currentThread();
            while (timer == me) {
```

```
                try {
                    Thread.currentThread().sleep(100);
                } catch (InterruptedException e) {
                }
                repaint();
            }
        }

    public void mouseEntered(MouseEvent e)
    {
            pauseClock = true;
    }
    public void mouseExited(MouseEvent e)
    {
            pauseClock = false;
    }
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {
            xcenter = e.getX();
            ycenter = e.getY();
            repaint();
    }

}
```

[15 marks]

6 marks for threading, 5 marks for mouse handlers etc. 4 marks misc and understanding.

2(b)

```
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.text.*;
import java.awt.event.*;

public class ClockApp extends Frame implements Runnable, MouseListener {

   private volatile Thread timer;
   private int lastxs, lastys, lastxm,
           lastym, lastxh, lastyh;
   private SimpleDateFormat formatter;     // Formats the date displayed
   private Date currentDate;               // Used to get date to display
   private Color handColor;                // Color of main hands and dial
   private Color numberColor;              // Color of second hand and numbers
   private int xcenter = 80, ycenter = 55; // Center position
        private boolean pauseClock = false;

   public ClockApp() {
      int x,y;
      lastxs = lastys = lastxm = lastym = lastxh = lastyh = 0;
      formatter = new SimpleDateFormat ("EEE MMM dd hh:mm:ss yyyy",
Locale.getDefault());
      handColor = Color.blue;
      numberColor = Color.darkGray;
                this.addMouseListener(this);
                this.setVisible(true);
      this.setSize(300,300);
   }
```

```
    public void paint(Graphics g) {

                    // Draw the circle and numbers
                    g.setColor(handColor);
                    g.drawArc(xcenter-50, ycenter-50, 100, 100, 0, 360);
                    g.setColor(numberColor);
                    g.drawString("9", xcenter-45, ycenter+3);
                    g.drawString("3", xcenter+40, ycenter+3);
                    g.drawString("12", xcenter-5, ycenter-37);
                    g.drawString("6", xcenter-3, ycenter+45);

                    int xh, yh, xm, ym, xs, ys;
                    int s = 0, m = 10, h = 10;
                    String today;
                    if (!pauseClock) currentDate = new Date();

                    formatter.applyPattern("s");
                    try
                    {
                            s = Integer.parseInt(formatter.format(currentDate));
                    }
                    catch (NumberFormatException n)
                    {
                            s = 0;
                    }
                    formatter.applyPattern("m");
                    try
                    {
                            m = Integer.parseInt(formatter.format(currentDate));
                    }
                    catch (NumberFormatException n)
                    {
                            m = 10;
                    }
                    formatter.applyPattern("h");
                    try
                    {
                            h = Integer.parseInt(formatter.format(currentDate));
                    }
                    catch (NumberFormatException n)
                    {
                            h = 10;
                    }

                    // Set position of the ends of the hands
                    xs = (int) (Math.cos(s * Math.PI / 30 - Math.PI / 2) * 45 + xcenter);
                    ys = (int) (Math.sin(s * Math.PI / 30 - Math.PI / 2) * 45 + ycenter);
                    xm = (int) (Math.cos(m * Math.PI / 30 - Math.PI / 2) * 40 + xcenter);
                    ym = (int) (Math.sin(m * Math.PI / 30 - Math.PI / 2) * 40 + ycenter);
                    xh = (int) (Math.cos((h*30 + m / 2) * Math.PI / 180 - Math.PI / 2) * 30 +
xcenter);
                    yh = (int) (Math.sin((h*30 + m / 2) * Math.PI / 180 - Math.PI / 2) * 30 +
ycenter);

                    // Erase if necessary
                    g.setColor(getBackground());
                    if (xs != lastxs || ys != lastys)
                    {
                            g.drawLine(xcenter, ycenter, lastxs, lastys);
```

```
                }
                if (xm != lastxm || ym != lastym)
                {
                        g.drawLine(xcenter, ycenter-1, lastxm, lastym);
                        g.drawLine(xcenter-1, ycenter, lastxm, lastym);
                }
                if (xh != lastxh || yh != lastyh)
                {
                        g.drawLine(xcenter, ycenter-1, lastxh, lastyh);
                        g.drawLine(xcenter-1, ycenter, lastxh, lastyh);
                }
                g.setColor(numberColor);
                g.drawLine(xcenter, ycenter, xs, ys);
                g.setColor(handColor);
                g.drawLine(xcenter, ycenter-1, xm, ym);
                g.drawLine(xcenter-1, ycenter, xm, ym);
                g.drawLine(xcenter, ycenter-1, xh, yh);
                g.drawLine(xcenter-1, ycenter, xh, yh);
                lastxs = xs; lastys = ys;
                lastxm = xm; lastym = ym;
                lastxh = xh; lastyh = yh;
}

public void start() {
    timer = new Thread(this);
    timer.start();
}

public void stop() {
    timer = null;
}

public void run() {
    Thread me = Thread.currentThread();
    while (timer == me) {
        try {
            Thread.currentThread().sleep(100);
        } catch (InterruptedException e) {
        }
        repaint();
    }
}

        public void mouseEntered(MouseEvent e)
        {
                pauseClock = true;
        }
        public void mouseExited(MouseEvent e)
        {
                pauseClock = false;
        }
        public void mousePressed(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {
                xcenter = e.getX();
                ycenter = e.getY();
                repaint();
        }

 public static void main(String args[])
```

```
        {
                ClockApp ca = new ClockApp();
                ca.start();

        }
}
```

[5 marks]
[1 change add main, 1 can const, 1 init->const, 1 call start, 1 frame and setsize, setvisible]

2(c)
In Java, when you declare an array of a class, you could say:
SomeObject[] a = new SomeObject[5];
But this creates only an array of references, you must then instantiate each reference, using:
a[1] = new SomeObject ();
Even for the default constructor (as shown). In C++, when you declare:
SomeObject a[5];
C++ would call the default constructor for each instance of the array, and there would be no need to explicitly call 'new' again to instantiate each reference of the array.This form of initialisation only allows you to call the default constructor (or constructor with no parameters). In C++, you can create an array of objects, with each object sitting right next to each other in memory, which lets you move from one object to another, simply by knowing the address of the first object. This is impossible in Java.

In C++ you can also create arrays, of pointer or references to objects. This is more closely related to how Java arrays work. Java arrays are always an array of references to objects (except if you create arrays of the basic data types, int, float, double etc.). If you were to use an array of pointers or references, you would have an array of null references (just like Java). The syntax would be like:
SomeObject **f = new SomeObject *[10];

[5 marks]

3(a) ***(Code in italics was provided)***

```
#include<iostream>
#include<string>
using namespace std;

class Person{
        string name, id;
  public:
        Person(string, string);
        virtual void display();
        virtual string getRole() = 0;
};

Person::Person(string nm, string anid): name(nm), id(anid) {}

void Person::display()
{
        cout << endl << "Role is: " << getRole() << endl;
        cout << "Name is: " << name << " and id is: " << id << endl;
}

class Student: public Person
{
        string programme;
        int year;
  public:
```

```
        Student(string, string, string, int);
        virtual void display();
        virtual string getRole();
};

Student::Student(string name, string id, string prog, int yr):
  Person(name, id), programme(prog), year(yr) {}

string Student::getRole() { return "student"; }

void Student::display()
{
        Person::display();
        cout << "Programme is: " <<  programme << " and year is: " << year << endl;
}

class Lecturer: public Person
{
        string office;
        int phoneNum;
  public:
        Lecturer(string, string, string, int);
        virtual void display();
        virtual string getRole();
};

Lecturer::Lecturer(string name, string id, string off, int ph):
  Person(name, id), office(off), phoneNum(ph) {}

string Lecturer::getRole() { return "lecturer"; }

void Lecturer::display()
{
        Person::display();
        cout << "Office is: " << " and phone number is: " << phoneNum << endl;
}

int main(void)
{
    // test of Student and Lecturer
    Student s("John","1234", "MENG", 1);
    s.display();
    Lecturer l("Derek", "5123", "S356", 5355);
    l.display();
}
```

[10 marks]

~1 for each of the 8 methods, 1 for main, and 1 for correct abstract use

3 (b)

```
template<class T, int size>
class Storage
{
        T values[size];
        int next;
  public:
        T& operator [](int index)
        {
          return values[index];
```

```
        }

        bool addElement(T value)
        {
          if (next>size) return false;
          values[next++]=value;
          return true;
        }

        int getSize() { return next-1; }
};
```

[7 marks]

2 for template syntax, 2 for operator and 2 for add, 1 for size

3(c)

```
Storage<Person*, 10> store;
store.addElement(new Lecturer("Derek Molloy", "5123", "S356", 5355));
store.addElement(new Student("John Doe","1234", "MENG", 1));
store.addElement(new Student("James Doe","1235", "MENG", 1));
for (int i=0; i<=store.getSize(); i++)
      store[i]->display();
```

[3 marks]

3(d)

```
vector<Person*> vStore;
vStore.push_back(new Lecturer("Derek Molloy", "5123", "S356", 5355));
vStore.push_back(new Student("John Doe","1234", "MENG", 1));
vStore.push_back(new Student("James Doe","1235", "MENG", 1));
for (int i=0; i<vStore.size(); i++)
   vStore[i]->display();
```

[5 marks]

4(a)

```
/*
 * Created on 20-Nov-2005
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;

public class EE553Painter extends JFrame implements ActionListener{

        public class PaintCanvas extends JComponent implements MouseListener
        {
                public class JShape{
                        public int c;
                        public void draw(Graphics g) {}
                }

                public class JLine extends JShape{
```

```
                        public int x1, y1, x2, y2;
                        public int c;
                        public JLine(int x1, int y1, int x2, int y2, int cindex) {
                                this.x1=x1; this.y1=y1; this.x2 = x2; this.y2 = y2; this.c =
cindex;
                        }
                        public void draw(Graphics g){
                                g.setColor(colors[c]);
                                g.drawLine(x1, y1, x2, y2);
                        }
                }

                public class JPoint extends JShape{
                        public int x1, y1;
                        public int c;
                        public JPoint(int x, int y, int cindex) { x1=x; y1=y; this.c=cindex; }
                        public void draw(Graphics g){
                                g.setColor(colors[c]);
                                g.drawLine(x1,y1,x1,y1);
                        }
                }

                public static final long serialVersionUID = 12346;
                private int tempX, tempY;
                private Vector<JShape> shapes= new Vector<JShape>();
                public boolean isLine = true;
                public int theColor = 0;

                public PaintCanvas()
                {
                        this.addMouseListener(this);
                }

                public void paint(Graphics g)
                {
                        for (int i=0; i<shapes.size(); i++)
                        {
                                shapes.elementAt(i).draw(g);
                        }
                }

                public void mousePressed(MouseEvent e) {
                        if (isLine)
                        {
                                tempX=e.getX();
                                tempY=e.getY();
                        }
                        else { shapes.add(new JPoint(e.getX(), e.getY(), theColor)); }
                }
                public void mouseReleased(MouseEvent e) {
                        shapes.add(new JLine(tempX, tempY, e.getX(), e.getY(), theColor));
                        this.repaint();
                }
                public void mouseClicked(MouseEvent e) {}
                public void mouseEntered(MouseEvent e) {}
                public void mouseExited(MouseEvent e) {}

        }

        public static final long serialVersionUID = 12345;
```

```
	private JButton lineB, pointB;
	private PaintCanvas canvas;
	private JComboBox colorCombo;
	private String[] colStr = { "Black", "Blue", "Red", "Green" };
	private Color[] colors = { Color.BLACK, Color.BLUE, Color.RED, Color.GREEN };

	public EE553Painter() {
		super("Painter");

		canvas = new PaintCanvas();
		canvas.setPreferredSize(new Dimension(300,250));

		JPanel controls = new JPanel(new FlowLayout());
		lineB = new JButton("Draw Line");
		pointB = new JButton("Draw Point");
		colorCombo = new JComboBox();
		for(int i=0; i<colStr.length; i++)    colorCombo.addItem(colStr[i]);
		controls.add(lineB);
		controls.add(pointB);
		controls.add(colorCombo);
		colorCombo.addActionListener(this);

		this.getContentPane().setLayout(new BorderLayout());
		this.getContentPane().add(controls, BorderLayout.NORTH);
		this.getContentPane().add(canvas, BorderLayout.SOUTH);

		this.pack();
		this.setVisible(true);
	}

	public void actionPerformed(ActionEvent e)
	{
		if (e.getSource().equals(lineB)) { canvas.isLine = true;  }
		else if (e.getSource().equals(pointB)) { canvas.isLine = false;  }
		else if (e.getSource().equals(colorCombo))
		{
			canvas.theColor = colorCombo.getSelectedIndex();
		}
	}

	public static void main(String[] args) {
		new EE553Painter();
	}

}
```

[20 marks]
Sample marks: 4 for interface, 4 for canvas, 4 for good shape structure, 4 for event structure, 4 for store and correct paint]

4(b)

Destructors [2 marks]

There are no destructors in Java, even thought there is a new keyword, there is no corresponding delete keyword, as Java takes care of all of the memory managment. Java has a special finalize() method that you can add to any class. The strange thing about this method is that you can use it for tidying up (printing a paper record for Account etc.), but you do not know when it will be called. In C++ this was easier, as the destructor is called when the object goes out of scope, but in Java the finalize() method will be called when the garbage collector

destroys the object. We do not know when this will be called! As discussed previously, we can request that the garbage collector should run, but we do not know when it will run! You can help the garbage collector slightly by setting all your references to null when you no longer need them, e.g. a = null; - will send a hint to the garbage collector that the object is no longer being used.

Nested Classes [1 mark]

In Java we have the facility to define a class inside of another class. We refer to this class as an *inner class*. The only issue worth mentioning with inner classes is that they have full access to the private data and methods of the class in which they are nested. Because of this, we do not need to pass state pointers to methods that need callback.

Access Specifiers [2 marks]

Like C++, in Java we have public, private and protected access specifiers, but we also have another access specifier "package". This is the default access specifier and means that all states and methods are accessible to all classes within the same package. There is no package access specifier keyword, it is simply the default if public, private or protected are not used. Access specifiers in Java are:

- public - accessible everywhere, an interface method (same as C++)
- private - accessible only in the class where it was defined (same as C++)
- protected - accessible in the class where it is defined, the derived classes and in the same package (almost the same as C++ with the e xception of the package)
- "package" - default (there is no package specifier keyword) and means that it is accessible by any class in the same package. You can equate this to the C++ friendly condition by saying that all of the classes in the same package (directory) are friendly.

[5 marks total]

5(a)

```
// The Transaction class - by Derek Molloy [13 marks]
import java.net.*;
import java.io.*;

public class Transaction implements Serializable
{
	public static final int LODGEMENT = 1;
	public static final int WITHDRAWAL = 2;
	public static final int BALANCEENQ = 3;
	public static final int CLOSEACT = 4;
	private int accountNumber;
	private int transactionType;
	private float amount;
	private String returnMessage = "None";
	private boolean successfulTransaction = false;

	public void display()
	{
		System.out.println("A transaction object");
		System.out.println("With Transaction Type: " + transactionType);
		System.out.println("With account number: " + accountNumber);
		System.out.println("With amount: " + amount);
	}

	public Transaction(int type, int actNum, float amount)
	{
		this.transactionType = type;
```

```
                this.accountNumber = actNum;
                this.amount = amount;
        }
        public Transaction(int type, int act Num)
        {
                this(type, actNum, 0.0f);
        }

        public void setReturnMessage(String s) { returnMessage = s; }
        public String getReturnMessage() {return returnMessage; }
        public void setReturnSuccess(boolean b) { successfulTransaction = b; }
        public boolean getReturnSuccess() { return successfulTransaction;}

        public boolean isForAccount(Account a)
        {
                if ( a.getAccountNumber() == accountNumber) return true;
                else return false;
        }

        public boolean applyTo(Account a)
        {
                if (this.transactionType == Transaction.LODGEMENT)
                        a.makeLodgement(amount);
                else if (this.transactionType == Transaction.WITHDRAWAL)
                        return a.makeWithdrawal(amount);
                else if (this.transactionType == Transaction.BALANCEENQ)
                        a.display();
                else if (this.transactionType == Transaction.CLOSEACT)
                        a.close();
                else return false;
                return true;
        }
}
```

Modifications to Server.java are: [2 marks]

```
break;
        } // create a new thread for the client
        ConnectionHandler con = new ConnectionHandler(clientSocket, theAccounts);
```

Modifications to HandleConection.java are: [5 marks]

```
        private Account[] theAccounts;
        // The constructor for the connecton handler

        public ConnectionHandler(Socket clientSocket, Account[] theAccounts)
        {
                this.clientSocket = clientSocket;
                this.theAccounts = theAccounts;
        }

        /** Receive and process incoming command from client socket */
        private boolean readCommand()
        {
                Transaction t = null;
                try
                {
                        t = (Transaction)is.readObject();
                }
                catch (Exception e)
```

```
                {
                        t = null;
                }
                if (t == null)
                {
                        this.closeSocket();
                        return false;
                }
                for (int i=0; i<=2; i++)
                {
                        if (t.isForAccount(theAccounts[i]))
                        {
                                System.out.println("Applying a transaction");
                                t.applyTo(theAccounts[i]);
                        }
                }
                return true;
        }
```

Modifications to Client.java are: [5 marks]

```
private void doTransactions()
{
        Transaction[] trans = { new Transaction(Transaction.LODGEMENT, 100001, 1000f),
                new Transaction(Transaction.BALANCEENQ, 100001),
                new Transaction(Transaction.WITHDRAWAL, 100001, 50f),
                new Transaction(Transaction.BALANCEENQ, 100001),
                new Transaction(Transaction.CLOSEACT, 100001),
                new Transaction(Transaction.BALANCEENQ, 100001) };
        for (int i=0; i<=5; i++)
                this.send(trans[i]);
}
```

[25 marks total]