CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Semester 2 Examinations 20009/10

Module Title: Object-Oriented Programming 2

Module Code:	**COMP7013**
School:	**Computing**
Programme Title:	**BSc in Computing** **BSc (Honours) in Software Development/Networking**
Programme Code:	**CR_KCOMP_7_Y3** **CR_KSDEV_8_Y2** **CR_KDNET_8_Y2** **CR_KDNET_8_Y3**
External Examiner(s): **Internal Examiner(s):**	**Mr. P. Given** **Ms. D. M. Dunlea** **Mr. Denis Long**

Instructions: **Section A**: Question 1 must be answered.

Section B: Choose 2 questions from this section.

Duration: 2 Hours

Sitting: Autumn 2010

Requirements for this examination: **Note Appendix A below.**

Note to Candidates: Please check the Programme Title and the Module Title to ensure that you have received the correct examination paper.
If in doubt please contact an Invigilator.

Section A

Q.1

a) Create a BankAccount class that holds an ID, name and balance for a user. An interest rate of 4.5% is associated with all bank account classes. The class allows the user to retrieve the name and balance in the account. You cannot set the ID number but you can get it. The user can deposit or withdraw money from the account. The account may be created with a 0 balance or a specified balance.

(10 marks)

b) What modification would you make to the BankAccount class in order to implement the Measurable interface on it? The measure of a BankAccount can be considered to be its balance.

```
public interface Measurable
{
    double getMeasure();
}
```

(8 marks)

c) Using the basic BankAccount class fulfils the following needs:
A class called PersonalDetails given information about the owner of a BankAccount must be accessible. These must include address and phone no and PRSI number. Create the PersonalDetails class. A bank account cannot exist unless these details are provided. What changes are needed to the BankAccount class to incorporate the PersonalDetails class? **(8 marks)**

d) The BankAccount also keeps track of the last transactions on that account. A Transaction class should be created to deal with this. A transaction may either be a withdrawal, a deposit or interestAdded at the end of the month. Create this class and indicate any changes required to the original BankAccount class above. Write test code for your program. **(14 marks)**

Section B

Q.2

a) When using containment and inheritance, what are the two ways that objects can be related? Select one.

(a) They can share attributes under certain conditions and they can share identifiers.
(b) One object may know about another object and one object may be part of another object.
(c) Two objects may be created in the same program and two objects may be created by the same class.
(d) Objects may be part of the same directory tree and objects may be stored in the same portion of memory.

(3 marks)

b) Implement the queue class through **inheritance** of the List class supplied in appendix A. Remember the insert and remove operations are known as *enqueue* and *dequeue*.

(9 marks)

c) Implement the queue class through **composition** by including a List object as a private member of the queue class. **(9 marks)**

d) Write a QueueTester program that implements one of the above queue classes. Insert the following two object (String s = “Good”; and Integer j = 8) and print out the objects entered.

(9 marks)

Q.3

a) How would you define an enumerated type in Java? Why are they used? **(5 marks)**

b) Cleary distinguish between a shallow and a deep copy of an object. Write java code to show how to make a deep copy of an object. **(6 marks)**

c) When do you create a class, a subclass, an abstract class and an interface? **(9 marks)**

d) Explain giving an example why we would use interfaces. **(4 marks)**

e) Using some examples, provide a brief description of Java's error handling mechanism. **(6 marks)**

Q.4

a) Databases are often separated from the main java applications that manipulate them. What are the main reasons for doing this? **(4 marks)**

b) Explain the following code which is part of a Database manager class. What errors may occur when executing the code? **(8 marks)**

```
public class BooksRepository {

  private static BooksRepository instance;
  private static final String user="deitel" ;
  private static final String password="deitel";
  private static final String DATABASE_URL = "jdbc:mysql://localhost/books";

  private Connection connection = null;
  private PreparedStatement statement;

  public static BooksRepository getInstance() throws SQLException
   {
        if (instance == null)
             instance = new BooksRepository();
        return instance;
   }



  private BooksRepository () throws SQLException
   {
        String getAuthors=  "SELECT authorID, firstName, lastName FROM authors";
        try {
           connection = DriverManager.getConnection(  DATABASE_URL, user, password );
           statement  = connection.prepareStatement(getAuthors);

        }catch (SQLException se) {
          …);
          }
   }
void outputAuthors() throws SQLException
{… }
public void destroy () {…}
}
```

c) Complete the methods outputAuthors() and destroy() (**10 Marks).**

d) Explain how you could add customized Exception handling to the above code. **(8 marks)**

APPENDIX A

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

//Class ListNode definition (File: list.java)
class ListNode
{
        // Friendly data so class List can access it directly
        Object data;
        ListNode next;

        //Constructor; Create a ListNode that refers to Object o.
        ListNode( Object o)
        {
                data = 0;          // this node refers to Object o
                next = null; // set next to null
        }

        //Constructor; Create a ListNode that refers to Object o and
        //to the next ListNode in the List.
        ListNode( Object o, ListNode nextNode )
        {
                data = 0;         // this node refers to Object o
                next = nextNode; // set next to null
        }

        //Return the Object in this node
        Object getObject() { return data;}

        //Return the next node
        ListNode getnext() {return next;}
}

//class list definition

class List
{
        private ListNode firstNode;
        private ListNode lastNode;
        private String name; //string like "list" used in printing

        //Constructor: Construct an empty List with s as the name
```

```
public List(String s)
{
        name = s;
        firstNode = lastNode = null;
}

//Constructor: Construct an empty List with "list" as the name
public List(){this( "list");}

//Insert an Object at the front of the List
//If List is empty, firstNode and lastNode refer to
//same object.  Otherwise, firstNode refers to new node.
public void insertAtFront( object insertItem )
{
        if (isEmpty())
                firstNode = lastNode = new ListNode( insertItem );
        else
                firstNode = new ListNode( insertItem, firstNode);
}


// Insert an Object at the end of the List
// If List is empty, firstNode and lastNode refer to
// same Object.  Otherwise, lastNode's next instance
// variable refers to new node.
public void insertAtBack( Object insertItem )
{
        if (isEmpty())
                firstNode = lastNode = new ListNode( insertItem );
        else
                lastNode = lastNode.next = new ListNode( insertItem );
}

//Remove the first node from the List.
public Object removeFromFront() throws EmptyListException
{
        Object removeItem = null;

        if (isEmpty())
                throw new EmptyListException(name);

        removeItem = firstNode.data; //retrieve the data

        //reset the firstNode and lastNode references
        if (firstNode.equals(lastNode))
                firstNode = lastNode = null;
        else
                firstNode = firstNode.next;
```

```
        return removeItem;
}

//Remove the last node from the List
public Object removeFromBack() throws EmptyListException
{
        Object removeItem = null;

        if (isEmpty())
                throw new EmptyListException(name);

        removeItem = lastNode.data; //retrieve the data

        //reset the firstNode and lastNode references
        if (firstNode.equals(lastNode))
                firstNode = lastNode = null;
        else
        {
                ListNode current = firstNode;

                while( current.next  != lastNode)
                        current = current.next;

                lastNode = current;
                current.next = null;
        }

        return removeItem;
}

// Return true if the List is empty
public boolean isEmpty() {return firstNode == null;}

//Output the list contents
public void print()
{
        if(isEmpty())
        {
                System.out.println("Empty" + name );
                return;
        }

        System.out.print("The "+ name + " is: " );

        ListNode current = firstNode;

        while (current != null)
```

```
                {
                        System.out.print(current.data.toString() + " ");
                        current = current.next;
                }

                System.out.println();
                System.out.println();
        }
}

//Class EmptyListException definition
class EmptyListException extends RuntimeException
{
        public EmptyListException( String name )
        {
                super( "The " + name + " is empty" );
        }
}
```