INSTITUTE OF TECHNOLOGY TRALEE

WINTER EXAMINATIONS AY 2014-2015

Object Oriented Programming 2

Module Code 61004_PROG

CRN 43840

External Examiner: Ms Sabrina Spillane

Internal Examiner: Mr John Walsh

Duration: 2 Hours

Instructions to Candidates: Please answer **Question 1** and any **two other Questions.**

Use of a pencil is permitted is for writing code.

Question 1

a) Write the class definition for a *Book.* The *Book* should have the following two **attributes**: title:String and author:String. The class should have a **no-argument constructor** that sets up the attributes indirectly via a call to a second **2-argument constructor**. The class should have **mutator** and **accessor** methods defined for both attributes as well as a **toString** method that uses the **String format method** to prepare its output. An example of the toString output is shown below:

Book: The Hobbit
Author: J RR Tolkien

(10 Marks)

b) An additional **class variable**/**attribute** is required for the *Book* class that **tracks the number of book objects created** using the class. The attribute should have an **appropriate accessor method**. Indicate where appropriate modifications are required to your code written for a) and write the code.

(6 Marks)

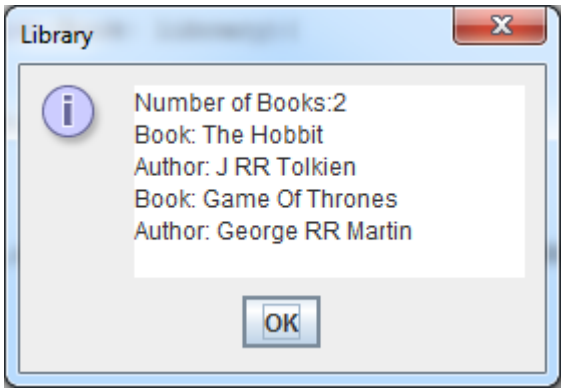
c) Java provides a powerful tool called **javadoc** for generating web pages to describe classes. Write javadoc style comments, including a general **description**, **@param** and **@return** tags for one accessor and one mutator method defined in part a) above.

(4 Marks)

d) Write a minimal **driver** class called **BookDriver**, including any import statements, that creates and an **ArrayList** of books called *library*. The values for each attribute of a book should be requested from the user using **input dialog boxes**. The application should cater for multiple books, terminating when the user enters "finished". See **Appendix 2** for some useful information from the Java API.

(8 Marks)

e) Write an additional class method called **displayBooks** for inclusion in the driver class written for part d) that displays the number of books and the ArrayList of Books. The statement calling this method from the main method is **displayBooks(library);**. The output is displayed in a message dialog box containing a **JTextArea**. An example output is shown below. See **Appendix 2** for some useful information from the Java API.



(12 Marks)

Question 2

As part of a Library system, additional methods are added to the driver class developed in Question 1. One method is called **returnBook()** that caters for returning books. The method returns the number of days that a book has been out on loan. A second method is called *calcFine* that calculate and returns the fine owed for late returns. The statement calling this method from the main method is **calcFine(numDays);** where numDays is the number of days a book is on loan.

a) As part of the **returnBook** method, the date the book was borrowed is entered via an input box in the form "dd/mm/yyyy". This is then parsed to give the day, month and year for the borrow date. Write the appropriate code to take in and parse the date. See **Appendix 2** for some useful information from the Java API.

(8 Marks)

b) In addition to part a), the **returnBook** method calculates the number of days that books have been on loan. The return date is taken as today's date. Write appropriate code to calculate the number of days that a book has been on loan. You can assume that the date the book is borrowed and returned are in the same calendar year. See **Appendix 2** for some useful information from the Java API.

(9 Marks)

c) The **returnBook** method returns the number of days a book has been on loan. The number of days is used in the method **calcFine(numDays)** to determine if a fine is to be paid by a library member. If the book has been on loan for more than 30 days then a fine of 20 cent per additional day is applied. Write code, including the method header, for the *calcFine* method that will return the fine amount due on a book.

(9 Marks)

d) Draw a UML diagram representation of the new **BookDriver** class.

(4 Marks)

Question 3

The class LibrarySystem.java (**Appendix 1**) is a **JFrame** based **GUI** program that allows the user to set up and maintain a **collection** of books for a Library. Each book is an object of the Book class developed in Question1. The collection is held in an **ArrayList** of Book objects called booklist.

a) Explain what input and output **streams** are in Java, differentiating between **low level** and **high level i/o**.

(6 marks)

b) Write code for the 'open' method, suitable for inclusion in LibrarySystem.java, that will **read** the **entire array** of books from a binary file called 'books.dat'. See **Appendix 2** for some useful information from the Java API.

(6 marks)

c) Write a note on how **exceptions** are handled in Java. Include **examples** of checked and unchecked exceptions that can occur.

(8 marks)

d) Rewrite the code for part b) such that any IOException or FileNotFoundException is dealt with by the 'open' method while any ClassNotFoundException exception is propagated back to the calling method. You should also prompt the user what type of exception occurred before the program crashes.

(10 marks)

Question 4

The class Library.java **(Appendix 1)** is a JFrame based GUI program that allows the user to set up and maintain a collection of books for a Library. The menu system has two menus, one called 'File' that allows for creating, opening and saving a file as well as quitting the program, the other called 'Book' that allows for adding a new book, displaying the entire array of books, returning books and calculating fines. Examine the code and answer the following questions.

a) Explain why there is no event handler class required for this program.

(4 marks)

b) Write code for the 'display' method suitable for inclusion in VetSurgery.java. The method should use a **JTextArea** object to display the details of the pet and use an **enhanced for loop** to scan through the **ArrayList** to retrieve the appropriate details for each Book object.

(9 marks)

c) The event-handling model in Java is the **delegation-based event model**. Explain how this model works. Include a diagram as part of your explanation.

(7 marks)

d) The Library program is rewritten so that an inner class is developed **to handle the Book menu events.** Write the code for this handler class. The class should invoke four methods called, addBook, displayBook, returnBook and calcFine. See See **Appendix 1 and Appendix 2** for some useful information from the Java API.

(10 marks)

Appendix 1 Library.java

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

// manages a collection of Books, holding the info in a ArrayList
public class LibrarySystem extends JFrame implements ActionListener{

    JMenu fileMenu,libraryMenu;
    ArrayList <Book> books;

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

   public LibrarySystem( ) {
      newSystem();
      setTitle    ( "LibrarySystem" );
      setSize     ( 400,200 );
      setLocation  ( 100,100 );
      Container pane = getContentPane();
      setDefaultCloseOperation( EXIT_ON_CLOSE );
      createFileMenu();
      createLibraryMenu();
      JMenuBar menuBar = new JMenuBar();
      setJMenuBar(menuBar);
      menuBar.add(fileMenu);
      menuBar.add(libraryMenu);
   } // end LibrarySystem

  public void newSystem() {
          books = new ArrayList<Book>();
   } // end new system

   public void save() throws IOException {
          //code not given
   } //end save

   public void open(){
          // code required for this part
    } // end open()

   public void addBook(){
          //code required for this part
    } // end addBook

   public void display(){
          //code required for this part
     } // end display

     // returnBook method code required
    // calcFine method code required
```

```
   // code for handling events is required


    private void createFileMenu(){
      // create the menu
           //code not given
    }

    private void createLibraryMenu(){
           libraryMenu = new JMenu("Book");
           JMenuItem item;
           item = new JMenuItem("Add");
           item.addActionListener(this);
           libraryMenu.add(item);
           item = new JMenuItem("Display");
           item.addActionListener(this);
           libraryMenu.add(item);
           item = new JMenuItem("Return Book");
           item.addActionListener(this);
           libraryMenu.add(item);
           item = new JMenuItem("Calculate Fine");
           item.addActionListener(this);
           libraryMenu.add(item);     }

} //end class
```

Appendix 2: Extracts from the Java API

Class String

Method Summary

Methods

Modifier and Type	Method and Description
`CharSequence`	`subSequence(int beginIndex, int endIndex)` Returns a new character sequence that is a subsequence of this sequence.
`String`	`substring(int beginIndex)` Returns a new string that is a substring of this string.
`String`	`substring(int beginIndex, int endIndex)` Returns a new string that is a substring of this string.

Class ArrayList<E>

Constructor Summary

Constructors

Constructor and Description
`ArrayList()` Constructs an empty list with an initial capacity of ten.
`ArrayList(Collection<? extends E> c)` Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
`ArrayList(int initialCapacity)` Constructs an empty list with the specified initial capacity.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`add(E e)` Appends the specified element to the end of this list.
`void`	`add(int index, E element)` Inserts the specified element at the specified position in this list.

int	**size**() Returns the number of elements in this list.
Object[]	**toArray**() Returns an array containing all of the elements in this list in proper sequence (from first to last element).
<T> T[]	**toArray**(T[] a) Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.

Class JTextArea

Constructor Summary

Constructors

Constructor and Description
JTextArea() Constructs a new TextArea.

Method Summary

Methods

Modifier and Type	Method and Description
void	**append**(String str) Appends the given text to the end of the document.

Class GregorianCalendar

Constructor Summary

Constructors

Constructor and Description
GregorianCalendar() Constructs a default GregorianCalendar using the current time in the default time zone with the default locale.
GregorianCalendar(int year, int month, int dayOfMonth) Constructs a GregorianCalendar with the given date set in the default time zone with the default locale.

Fields inherited from class java.util.Calendar

ALL_STYLES, AM, AM_PM, APRIL, areFieldsSet, AUGUST, DATE, DAY_OF_MONTH, DAY_OF_WEEK, DAY_OF_WEEK_IN_MONTH, DAY_OF_YEAR, DECEMBER, DST_OFFSET, ERA, FEBRUARY, FIELD_COUNT, fields, FRIDAY, HOUR, HOUR_OF_DAY, isSet, isTimeSet, JANUARY, JULY, JUNE, LONG, MARCH, MAY, MILLISECOND, MINUTE, MONDAY, MONTH, NOVEMBER, OCTOBER, PM, SATURDAY, SECOND, SEPTEMBER, SHORT, SUNDAY, THURSDAY, time, TUESDAY, UNDECIMBER, WEDNESDAY, WEEK_OF_MONTH, WEEK_OF_YEAR, YEAR, ZONE_OFFSET

Methods inherited from class java.util.Calendar

after, before, clear, clear, compareTo, complete, get, getAvailableLocales, getDisplayName, getDisplayNames, getFirstDayOfWeek, getInstance, getInstance, getInstance, getInstance, getMinimalDaysInFirstWeek, getTime, getTimeInMillis, internalGet, isLenient, isSet, set, set, set, set, setFirstDayOfWeek, setLenient, setMinimalDaysInFirstWeek, setTime, setTimeInMillis, toString

Class ActionEvent

Method Summary

Methods

Modifier and Type	Method and Description
String	getActionCommand() Returns the command string associated with this action.
int	getModifiers() Returns the modifier keys held down during this action event.

Interface ActionListener

Method Summary

Methods

Modifier and Type	Method and Description
void	actionPerformed(ActionEvent e) Invoked when an action occurs.

Class FileInputStream

Constructor Summary

Constructors

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

Class ObjectInputStream

Constructor Summary

Constructors

Modifier	Constructor and Description
`protected`	`ObjectInputStream()` Provide a way for subclasses that are completely reimplementing ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.
	`ObjectInputStream(InputStream in)` Creates an ObjectInputStream that reads from the specified InputStream.

Method Summary

Methods

Modifier and Type	Method and Description
`String`	`readLine()` **Deprecated.** *This method does not properly convert bytes to characters. see DataInputStream for the details and alternatives.*
`long`	`readLong()` Reads a 64 bit long.