No calculator permitted in this examination

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – Degree of BSc with Honours
Artificial Intelligence and Computer Science
Computer Science
Computer Science with Study Abroad
Computer Science with Business Studies

First Year – Degree of BEng/MEng with Honours
Computer Science/Software Engineering

Undergraduate Occasional
Computer Science/Software Engineering

06 18190

Software Workshop 1

Summer Examinations 2011

Time allowed: 3 hours

[Answer ALL Questions]

[Use a Separate Answer Book for EACH Section]

SECTION A

[Use a Separate Answer Book for THIS Section]

1. (a) Examine the following two functions.

```
private double calculate(int length) {
    return length / 2 * Math.PI;
}

private double calculate2(double length) {
    return length / 2 * Math.PI;
}
```

(i) Do the functions return the same result? [1%]

(ii) Provide your reasoning for your answer. [1%]

(b) A programmer has been asked to write code for a function to extract words from a character string. The programmer's proposed solution uses recursion.

```
public String extractWordFour(String text) {
    if (text.length() == 0 || text.charAt(0) == ' ') {
        return "";
    } else {
        return text.charAt(0) +
            extractWordFour(text.substring(1));
    }
}
```

(i) Rewrite the code to use a **`for`** loop. [4%]

(ii) Rewrite the code to use a **`while`** loop [4%]

(c) A new programmer is joining your team. They haven't used object-oriented languages before and are having difficulty with the difference between a class and an object. Jack, a member of your team, wants to explain this difference in terms of cookie cutter and dough. Which of cookie cutter and the dough would you relate to a class and object? [2%]

2. A programmer has to write a program for a GPS tracking program. He has recognised that both the Latitudinal and Longitudinal values are based on the same underlying coordinate data structure of degrees, minutes, seconds, and direction. As a result, he has written the following JUnit test code.

```
@Test
public void testSetDegrees() {
    Coordinate aLatitude = new Latitude(-1, 0, 0, 'S');
    assertEquals(-1, aLatitude.getDegrees());
    assertEquals(0, aLatitude.getMinutes());
    assertEquals(0.00, aLatitude.getSeconds(), 0.00);
    assertEquals('S', aLatitude.getDirection());
}
```

Note: This exercise isn't about understanding how GPS coordinates are defined. It is about knowing what the Java testing code is telling you about how the Java source code should be written.

Based on this code

(i) Write the `interface` for `Coordinate` (Don't include JavaDoc or method code) [5%]

(ii) You have written `Coordinate` using the Java `interface` construct. Describe an alternative approach to writing the skeleton code for `Coordinate` and the reasons why you might use the approach ahead of using an `interface`. [4%]

(iii) Write the class header for `Latitude` [1%]

(iv) If you had used the alternative approach to implementing `Coordinate` how would the class header for the `Latitude` differ from the solution used in (iii)? [1%]

3. You have been asked to write a class that would convert between number bases. The class needs to have the following characteristics:

- The constructor should accept two parameters. These are the 'from' number base and the 'to' number base. (e.g. to create an object to convert between base 4 and base 16, the 'from' number base would be 4 and the 'to' number base 16).
- A method that takes a string as input and returns a string as output. The input string has a number in the 'from' number system and the resulting string will be either null if the input number is not a valid number in the 'from' number system or a string in the 'to' number system. (e.g. the string "3333" in base 4 as input would return the string "FF" in base 16.

(i) Describe in English a sequence of tests that you might use to test code to implement the number conversion class. Your tests should cover all of the conditions described in the above definition.

[10%]

(ii) A programmer plans to write the class using the following class skeleton.

```
public class NumberConvert {

    public NumberConvert(int from, int to) {}
    public String convert(String input) {}
}
```

Write at least two JUnit test methods for the tests that you have described in part (i).

[6%]

(iii) The **Integer** class in the Java API provides a function **valueOf**.

```
public static Integer valueOf(String s,
                              int radix)
                       throws NumberFormatException
```

The string argument is a signed integer in the number base (**radix**) specified by the second argument. The function throws a **NumberFormatException** if the number base is outside the allowable range or the string is not a signed integer in the number base.

The **Integer** class also provides a function **toString**.

```
public static String toString(int i, int radix)
```

This creates a string from the first argument in the number base (**radix**) specified by the second argument.

Write code for the **`NumberConvert`** class. The **`convert`** function should use the **`valueOf`** and **`toString`** functions described above. If the **`NumberFormatException`** exception is thrown, then the **`convert`** function is to return the empty string (**`""`**).

[10%]

4. A project requires a **`Date`** class. The class performs a number of calculations on dates.

A Gregorian date is represented by three numbers, the year, month, and day. The month is represented by a number between 1 and 12 inclusive and the day by a number between 1 and the maximum number of days in the month.

ISO 8601 defines an alternative representation that is the year and day number within the year. The day number ranges from 001 for 1st January to 365 (366 in leap years) for 31st December.

In the **`Date`** class, each instance stores a valid *date*, with integer fields for day, month, and year. Each instance is to be *immutable*, which means once it is created if field values cannot be changed.

The class will include the following:

Getter methods **`getDay`**, **`getMonth`**, and **`getYear`**.

An array **`daysInMonth`** that stores the lengths of the months in a normal (non-leap) year. They are 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31.

A method **`leapYear`** that returns a boolean value true if the year is a leap year. A year is a leap year if either (i) it is divisible by 4 but not by 100 or (ii) it is divisible by 400.

A method **`getGregorianDateNumber`** that returns an integer number that is calculated using the formula: year **`x`** 384 + month **`x`** 32 + day.

A method **`getISO8601Date`** that returns an integer number that is calculated using the formula: year **`x`** 367 + day of year.

A constructor with three parameters for day, month, and year. It should throw an **`IllegalArgumentException`** if the parameters are an invalid combination.

Write

(i) the **`leapYear`** function [6%]
(ii) the **`getGregorianDateNumber`** function [4%]
(iii) the **`getIS08601Date`** function [8%]

SECTION B

[Use a Separate Answer Book for THIS Section]

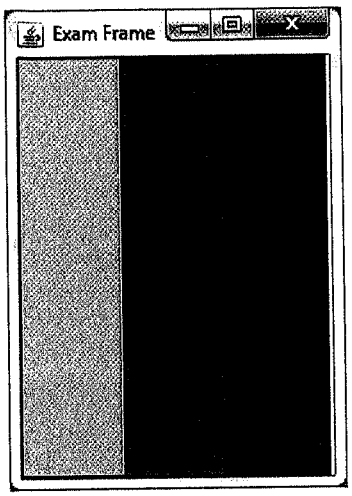
5. The window illustrated contains a Panel described by the following Java class:

```
import java.awt.*;

public class ExamPanel extends Panel {

   private ExamCanvas canv1, canv2, canv3;

   public ExamPanel () {
      setLayout(new GridLayout(1,3));
      canv1 =
         new ExamCanvas(Color.LIGHT_GRAY);
      canv2 = new ExamCanvas(Color.GRAY);
      canv3 =
         new ExamCanvas(Color.DARK_GRAY);
      add(canv1); add(canv2); add(canv3);
   }
}
```



(a) An `ExamPanel` object contains three `ExamCanvas` objects. Write a class `ExamCanvas` that can be used as illustrated in the `ExamPanel` class. [4%]

(b) Write a Java class `ExamFrame` that can be used to create windows like the one illustrated above (of size 200x300 pixels), and write a main method to display one such window. [4%]

Question 5 continues over the page

Question 5 continues

(c) Now modify the `ExamCanvas` class so that it has an attribute called `highlight` of type `java.awt.Color` with the value given as a parameter in the `ExamCanvas` constructor and with the behaviour described below. (`setHighlight` and `getHighlight` methods should also be defined.)

The initial behaviour should be such that each time the mouse pointer enters the region on the screen defined by a canvas, the background colour of the canvas should change from white to the current highlight colour and when the mouse exits the region, the background colour should change back to white. *However this behaviour should only occur when it is switched on.*

A mouse click on a canvas should switch the highlighting behaviour for that canvas from on to off or from off to on. Implement this behaviour with the `MouseListener` interface, from the Java API, which is defined as:

```
package java.awt.event;
import java.util.EventListener;

public interface MouseListener extends EventListener
{
   public void mouseClicked(MouseEvent e);
   public void mousePressed(MouseEvent e);
   public void mouseReleased(MouseEvent e);
   public void mouseEntered(MouseEvent e);
   public void mouseExited(MouseEvent e);
}
```

[8%]

6. A Java class is required that describes a data structure that can be used to store a collection of a large number of strings. The collection will be a set in the sense that it will not contain any duplicate entries. The following interface string set, specifies the functionality required of our and any class we define must implement this interface:

```
package q7;

/** The functionality required to store a set of strings. */
public interface StringSet
{
   /** Tests whether the set contains a given string **/
   public boolean contains(String string);

   /** Adds a new string to the set, ONLY if not there already
    * @return true if the string was added, false if not.
    */
   public boolean add(String string);

   /** Reports the number of strings stored in the set */
   public int size();
}
```

A class called `LinkedListStringSet` that implements this interface has already been written. A `LinkedListStringSet` object uses a linked list of Strings to store the set. The class has the heading:

```
public class LinkedListStringSet implements StringSet
```

(a) Write a class `HashStringSet` that implements the `StringSet` interface and uses a hash table of 1000 linked lists to store the strings. Do not use any of the Java Collection classes in your answer to this part. [10%]

(b) The heading for the class `HashSet` in the Java library package `java.util` includes the following elements:

```
public class HashSet<E> ... implements Set<E>, ...
```

The method `add` defined in the class `HashSet` starts with:

```
public boolean add(E e) {...}
```

Explain the use of the type `E` in the above code fragments. [4%]

Question 6 continues over the page

Question 6 continued

(c) Your `HashStringSet` defined in (a) might be used in a statement such as:

```
StringSet words = new HashStringSet();
```

Show how this could be replaced by an equivalent statement that uses the class `HashSet<E>` from the `java.util` package to create a variable `words` with exactly the same functionality.

[3%]