CSC1022 Exercises 2015-16 12 Marks

Deadlines:

Exercises 1-5 Friday 26th February 2016,

Exercises 6-10 Friday 11th March 2016

1. Write a class called Exercise1 which contains a static method `min` that finds the minimum value in an array of integers. Make sure that the method works even if the array contains only one element. Write an Exception class that should be thrown if the array does not contain any elements. Note that the `min` method is static so that it can be called by other classes as follows:

```
public static void main(String[] args) {
    int[] theArray = new int[10];
    // initialise array
    int result = Exercise1.min(theArray);

    // Where the class Exercise1
    // contains the min method
}
```

Complete the main method above with appropriate try and catch blocks to test that the min method works correctly in all scenarios. [1 Mark]

2. Complete the following Java Class that may be used to represent a Rational number (i.e. a number of the form a /b where a and b are ints. Hint: b should be positive). Implement the constructors (the default should set the number to 0/1) and methods to `subtract, multiply` and `divide` two Rational numbers. Also implement a `toString` method. Do not worry about simplifying your fractions (an answer of 2/4 is just as good as 1/2). Remember that fractions must not have denominators of zero and so throw appropriate exceptions if they do. Demonstrate that your class works by writing a main method that uses all the operations of your type e.g.

```
public class Rational {
   private int num;
   private int denom;

   public Rational() {...}
   public Rational(int num, int denom) {...}

   int getNum() {...}
   int getDenom() {...}

   public Rational add(Rational rhs) {
         return new Rational(num*rhs.denom+rhs.num*denom,
                                        denom*rhs.denom);
   }
   public Rational subtract(Rational rhs) {...}
   public Rational multiply(Rational rhs) {...}
   public Rational divide(Rational rhs) {...}
   public String toString() {...}
```

```
    public static void main(String[] args) {
        Rational r1 = new Rational(1, 2)  // 1/2
        Rational r2 = new Rational(3, 4)  // 3/4
        Rational result = new Rational();
        result = r1.add(r2);
        System.out.println(result);
        // etc
    }
}
```

[2 Marks]

3. Design and implement a class `Rectangle` to represent a rectangle. You should provide two Constructors for the class, the first being the default constructor and the second which takes the basic dimensions and sets up the private member variables with the appropriate initial values. Methods should be provided that allow a user of the class to find out the length, width, area and perimeter of the shape plus a `toString()` method to print the values of the basic dimensions. Now implement a class called Square that represents a square. Class `Square` must be derived from `Rectangle`. Make sure you override `toString()`.

[2 Marks]

4. Consider the following Java Library interface

```
public interface Comparable<T> {
      int compareTo(T rhs);
}
```

Complete the implementation of the class below that implements the above interface (**note this interface is automatically imported by java – do NOT re-type it in your project**). The `compareTo` method should return -1 if `value` is less than `rhs.value`, 0 if both sides are equal and +1 if `value` is greater than `rhs.value`.

```
public class MyInt implements Comparable<MyInt> {
      private int value;

      MyInt(int x) {...}
      public String toString() {...}
      public int intValue() {...}
      public int compareTo(MyInt rhs){...}
}
```

[1 Mark]

5. Adapt your Rational class of Exercise 2 to implement the `Comparable<T>` interface. [1 Mark]

6. Implement a *recursive* function to evaluate binomial co-efficients C(n,k) (i.e. n!/(n-k)!* k!). You may use the following information

c(n, k) = 1 if k = 0

c(n, k) = 1 if k = n

c(n, k) = 0 if k> n

c(n, k) = c(n-1, k-1) + c(n-1, k) if 0<k<n

[1 Mark]

7. Implement a recursive Greatest Common Denominator method that find the GCD of two integers. Make sure they are both positive and use the observations that gcd(x, x) = x and gcd(x, y) = gcd(x-y, y) if x > y. Add this method to your Rational Numbers class and call it at appropriate times to simplify your fractions. [1 Mark]

8. Write a program that creates a `List` of Rationals and sorts them into increasing order. Use appropriate methods from the Collections Framework classes to sort the elements into increasing order. [1 Mark]

9. Write a `Comparator` to compare two Rationals so that a/b>c/d if a>c i.e. sort by increasing order of numerators only (I know that in practice nobody would ever want you to do this but, for the sake of this exercise, I do!). Test your program by modifying your answer to question 8. Note **do NOT change** the Rational class in order to solve this problem. [1 Mark]

10. Override the `equals()` and `hashCode()` methods appropriately in your Rational Numbers class. [1 Mark]