بسم الله الرحمن الرحيم

Data Structures and Algorithms

Homework #1

EE-367 Date: 21/06/1443
Spring 2022 24/01/2022
Dr. A. M. Al-Qasimi Due: 31/01/2022

Objective:

To introduce the student to the subject of data structures using the Java programming language, by reviewing the introductory chapters from the textbook.

What to do:

Read chapters 1 & 2 from the text book and then solve the following problems:

1) R-1.3:
Write a short Java function, isMultiple, that takes two **long** values, *n* and *m*, and returns true if and only if *n* is a multiple of *m*, that is, $n = mi$ for some integer *i*.

2) R-1.5:
Write a short Java function that takes an integer *n* and returns the sum of all the positive integers less than or equal to *n*.

3) R-1.6:
Write a short Java function that takes an integer *n* and returns the sum of all the odd positive integers less than or equal to *n*.

4) R-1.10:
Write a Java class, Flower, that has three instance variables of type String, int, and float, which respectively represent the name of the flower, its number of petals, and price. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type, and getting the value of each type.

5) C-1.23:
Write a short Java program that takes two arrays *a* and *b* of length *n* storing **int** values, and returns the dot product of *a* and *b*. That is, it returns an array *c* of length *n* such that $c[i] = a[i] \cdot b[i]$, for $i = 0, \ldots, n-1$.

6) R-2.6:
Give a short fragment of Java code that uses the progression classes from Section 2.2.3 to find the 8th value of a Fibonacci progression that starts with 2 and 2 as its first two values.

7) R-2.11:
Consider the following code fragment, taken from some package:

```
Public class Maryland extends State {
    Maryland() { /* null constructor */ }
    public void printMe() { System.out.println("Read it."); }
    public static void main(String[] args) {
        Region mid = new State();
```

```
        State md = new Maryland();
        Object obj = new Place();
        Place usa = new Region();

        md.printMe();
        mid.printMe();
        ((Place) obj).printMe();
        obj = md;
        ((Maryland) obj).printMe();
        obj = usa;
        ((Place) obj).printMe();
        usa = md;
        ((Place) usa).printMe();
    }
}

class State extends Region {
    State() { /* null constructor */ }
    public void printMe() { System.out.println("Ship it."); }
}

class Region extends Place {
    Region() { /* null constructor */ }
    public void printMe() { System.out.println("Box it."); }
}

class Place extends Object {
    Place() { /* null constructor */ }
    public void printMe() { System.out.println("Buy it."); }
}
```

What is the output from calling the main() method of the Maryland class?

8) R-2.12:

Draw a class inheritance diagram for the following set of classes:

- Class Goat extends Object and adds an instance variable tail and methods milk() and jump().
- Class Pig extends Object and adds an instance variable nose and methods eat() and wallow().
- Class Horse extends Object and adds instance variables height and color, and methods run() and jump().
- Class Racer extends Horse and adds a method race().
- Class Equestrian extends Horse and adds an instance variable weight and methods trot() and isTrained().

9) R-2.13:

Consider the inheritance of classes from Exercise R-2.12, and let *d* be an object variable of type Horse. If *d* refers to an actual object of type Equestrian, can it be cast to the class Racer? Why or why not?

10) C-2.24:

Write a Java class that extends the Progression class so that each value in the progression is the absolute value of the difference between the previous two values. You should include a default constructor that starts with 2 and 200 as the first two values and a parametric constructor that starts with a specified pair of numbers as the first two values.