North South University

Department of Electrical and Computer Engineering

CSE 215L: Programming Language II Lab

Lab Manual - 8

Lab Instructor: Taif Al Musabe

Objective:

- To explore the differences between the procedural paradigm and object-oriented paradigm
- To discover the relationships between classes
- To design programs using the object-oriented paradigm
- To simplify programming using automatic conversion between primitive types and wrapper class types

Task – 1

(*The **Fan** class*) Design a class named **Fan** to represent a fan. The class contains:

■ Three constants named **SLOW**, **MEDIUM**, and **FAST** with the values **1**, **2**, and **3** to denote the fan speed.

■ A private **int** data field named **speed** that specifies the speed of the fan (the default is **SLOW**).

■ A private **boolean** data field named **on** that specifies whether the fan is on (the default is **false**).

■ A private **double** data field named **radius** that specifies the radius of the fan (the default is **5**).

■ A string data field named **color** that specifies the color of the fan (the default is **blue**).

■ The accessor and mutator methods for all four data fields.

■ A no-arg constructor that creates a default fan.

■ A method named **toString()** that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string "fan is off" in one combined string.

Write a test program that creates two **Fan** objects. Assign arbitrarily maximum speed, radius, color, and on to the first object and second object. Display the objects by invoking their **toString** method.

Task – 2

(*The* **Time** *class*) Design a class named **Time**. The class contains:

■ The data fields **hour**, **minute**, and **second** that represent a time.

■ A no-arg constructor that creates a **Time** object for the current time. (The values of the data fields will represent the current time.)

■ A constructor that constructs a **Time** object with a specified elapsed time since midnight, January 1, 1970, in milliseconds. (The values of the data fields will represent this time.)

■ A constructor that constructs a **Time** object with the specified hour, minute, and second.

■ Three getter methods for the data fields **hour**, **minute**, and **second**, respectively.

■ A method named **setTime(long elapseTime)** that sets a new time for the object using the elapsed time. For example, if the elapsed time is **555550000** milliseconds, the hour is **10**, the minute is **19**, and the second is **10**.

Write a test program that creates two **Time** objects and displays their hour, minute, and second in the format hour:minute:second.

(*Hint*: The first two constructors will extract the hour, minute, and second from the elapsed time. For the no-arg constructor, the current time can be obtained using **System.currentTimeMillis**).

Homework – 1

(*The* **MyInteger** *class*) Design a class named **MyInteger**. The class contains:

■ An **int** data field named **value** that stores the **int** value represented by this object.

■ A constructor that creates a **MyInteger** object for the specified **int** value. ■
A getter method that returns the **int** value.

■ The methods **isEven()**, **isOdd()**, and **isPrime()** that return **true** if the value in this object is even, odd, or prime, respectively.

■ The static methods **isEven(int)**, **isOdd(int)**, and **isPrime(int)** that return **true** if the specified value is even, odd, or prime, respectively.

■ The static methods **isEven(MyInteger)**, **isOdd(MyInteger)**, and **isPrime(MyInteger)** that return **true** if the specified value is even, odd, or prime, respectively.

■ The methods **equals(int)** and **equals(MyInteger)** that return **true** if the value in this object is equal to the specified value.

■ A static method **parseInt(char[])** that converts an array of numeric characters to an **int** value.

■ A static method **parseInt(String)** that converts a string into an **int** value.

Write a client program that tests all methods in the class.

Homework – 2

(*The* **Queue** *class*) Design a class named **Queue** for storing integers. Like a stack, a queue holds elements. In a stack, the elements are retrieved in a last-in first-out fashion. In a queue, the elements are retrieved in a first-in first-out fashion. The class contains:

■ An **int[]** data field named **elements** that stores the **int** values in the queue.

■ A data field named **size** that stores the number of elements in the queue. ■
A constructor that creates a **Queue** object with default capacity **8**.

■ The method **enqueue(int v)** that adds **v** into the queue.

■ The method **dequeue()** that removes and returns the element from the queue.

■ The method **empty()** that returns true if the queue is empty.

■ The method **getSize()** that returns the size of the queue.

Implement the class with setting a initial array size. The array size will be doubled once the number of the elements exceeds the size. After an element is removed from the beginning of the array, you need to shift all elements in the array one position the left. Write a test program that adds some arbitrary numbers into the queue and removes these numbers and displays them