Computer Science COMP-2120 - Fall 2020

Assignment 2

Due: End of Sunday, November 1, 2020

****

Note: Submit your assignment as a **single zip file** on Blackboard. The zip file should be named as Assignment1_*StudentId*.zip, in which replace *StudentId* with your university student number. The zip file should include only several Java files, i.e. files with the extension name ".java". DO NOT include any compiled class file. For every problem, follow the name of the files as instructed.

****

Problem 1. (25 points)

Alice is going to create a notebook including the profile of all her friends. For each of her friends, she would like to keep first-name, last-name, cell number, and birthdate (month and day). At the beginning, she doesn't know how many spaces she should reserve for recording her friends, so she gradually inputs her friends' information into it.

- Please help her by first creating a Java class, called **Person**, with the required information from description above, along with any required setter/getter methods. (5 points)

- Then create, a Java class, called **FriendsList**, that keeps the list of some persons as someone's friends, using the Person class above, and performs the required operations, such as add new friends, delete/ modify an existing one, return the list of her friends sorted by last-names, report the number of her friends, report the list of her friends who born in a given month sorted by their day of birth, and report the list of her friends who born in a given day of a month, sorted by their last-names. She also needs to get the cell number of a friend by giving her/his first-name and last-name. (10 points)

- Create a tester class, called **MyFriends**, to test the FriendList class by creating a new object of this class, adding some friends, modifying and deleting some existing friends, retrieving and printing all friends, retrieving and printing the list of friends who born in a given month, retrieving and printing the list of friends who born in a given day of a month, and finding the cell number of an existing friend by giving her/his first-name and last-name. Also, create a separate copy of the FriendList object you have already created, and remove all friends with first-name "Shane" from the new object. (10 points)

Java file names: Person.Java, FriendList.java, and MyFriends.java.

Problem 2. (40 points)

In this problem, we are going to reuse the Person class created in the previous problem. Therefore, first make sure that you have already created Person.java based on the description provided above.

- Create a subclass of Person, called **Employee**, such that it can keep extra information of a person including year of hiring, annual salary, vacation days, and unused vacation days. Provide required setter/getter methods for this class. (10 points)

- Create a subclass of Employee, called **Manager**, which also has monthly bonus, and the list of all her/his employees. Provide required setter/getter methods for this class. (10 points)

- Create a class, called **Company**, with the following information: name of the company, starting year, and the list of all its employees/managers. Provide required setter/getter methods for this class. (10 points)

- Create a tester class, called **MyCompany**, in which you should test all the above classes by creating a company, adding some employees/managers into it. (10 points)

- Bonus: Provide two listings of the company's employees/managers, one sorted by last-names, and one sorted by each manager and then her/his employees. (5 points)

Java file names: Person.Java, Employee.java, Manager.java, Company.java, and MyCompany.java.

Problem 3. (15 points)

Complete the two Java programs below. The first one, PrimeGenerator, generates prime numbers, and the second one, prompts the user for an integer, and then prints out all prime numbers up to that integer. For example, when the user enters 10, the program should print:

2
3
5
7

```
/**
This class generates all prime numbers.
*/
public class PrimeGenerator
{
private int current;
public PrimeGenerator()
{
current = 1;
}
/**
Calculates the next prime number.
@return the next prime number
*/
public int nextPrime()
{
// Complete this part
...
}
/**
Check if n is a prime number.
@param n to check whether it is prime or not
@return true if n is prime
*/
public static boolean isPrime(int n)
{
// Complete this part
...
}
}
```

```
-------------------
import java.util.Scanner;
/**
This class prints prime numbers.
*/
public class PrimePrinter
{
public static void main (String[] args)
{
Scanner in = new Scanner(System.in);
System.out.print("Enter upper limit: ");
int limit = in.nextInt();
// Complete this part
...
}
}
```

Java file names: Same as the ones above.

Problem 4. (20 points)

Write a Java class `ComboLock` that works like the combination lock in a gym locker. The lock is constructed with a combination, three numbers between `0` and `39`. The `reset` method resets the dial so that it points to 0. The `turnLeft` and `turnRight` methods turn by a given number of ticks to the left or right. The `open` method attempts to open the lock. The lock opens if the user first turned it right to the first number in the combination, then left to the second, and then right to the third. The class and its methods' signatures are as follows:

```
public class ComboLock {
	. . .
	public ComboLock(int secret1, int secret2, int secret3) { . . . }
	public void reset() { . . . }
	public void turnLeft(int ticks) { . . . }
	public void turnRight(int ticks) { . . . }
	public boolean open() { . . . }
}
```

Also, provide a tester class, `ComboLockTester`, that tests the above class using various combinations entered by a user inside a loop. User will exit from the loop by entering three 0s as the combination.

Java file names: ComboLock.java and ComboLockTester.java