

King Saud University
College of Computer and Information Sciences
CSC111 – Spring 2021
Lab Final Exam

- مدة الاختبار 4 ساعات. من 2 مساءً حتى 6 مساءً يوم السبت 10 أبريل 2021.
- لن يتم التسامح في الغش.
- لن يقبل أي تسليم متأخر بعد الساعة السادسة مساءً.
- التسليم يكون عن طريق البريد الإلكتروني.
- ترسل حلك على نفس الإيميل تبع الواجبات الفرق الوحيد بالعنوان بدال كلمة HW# تكتب FINAL
- للتذكير، الإيميل هو CSC111LAB@GMAIL.COM
- قم بتسمية الكلاسات حسب ماهو مذكور بالإمتحان. وإسم البروجكت إسم العائلة تبعك.
- الرجاء حل جميع الأسئلة في مشروع واحد.
- عند إنشاء الملف المضغوط قم بتسميته على النحو التالي.
- zip اسم العائلة_اسمك_رقمك الجامعي_Final
- zip_Final_123456789_FirstName_LastName
- اكتب اسمك ورقمك الجامعي ووقت شعبتك في بداية كل كلاس كملاحظه. خصم درجة في حال لم تقم بذلك.

```
// Firstname Lastname
// id
// Section time
```

- توزيع الدرجات: 25 درجة هي الدرجة الكاملة للاختبار:

○ 15 درجة للسؤال الأول.

○ 5 درجات للسؤال الثاني.

○ 5 درجات للسؤال الثالث.

Create one Project that will contain all classes in this exam. You are to submit one project only.

Question 1 (15 points):

Write a class **Employee** (12 Points) that stores the information of an employee.

Employee	Points
- salary: double - department: String - years: int - manager: Employee	0.5
+ Employee(salary: double, department: String, manager: Employee) + getSalary(): double + getDepartment(): String + getManager(): Employee + changeManager(newManager: Employee): boolean + isManager(): boolean - isValidManager(manager: Employee): boolean + yearlyRaise(performance: double): boolean + toString(): String	1 0.5 0.5 0.5 2 1 2 3 1

Attributes:

- **salary**: the employee's salary
- **department**: the name of the department the employee works in.
- **years**: the number of years the employee worked in the company.
- **manager**: represents the manager of the employee. If this attribute is NULL then it means that this object is a manager.

Notice that:

- The employee and his manager must have the same department.
- A department can have more than one manager.
- A manager can not have a manager.

Methods:

- **Employee(salary: double, department: String, manager: Employee)**: the constructor that initializes the attributes. Sets **years** to 0. If the **manager** is not valid, set **manager** to NULL.
- **getSalary()**: returns the employee's **salary**.
- **getDepartment()**: returns the employee's **department** name.
- **getManager()**: returns the employee's **manager** object.

- **changeManager(newManager: Employee):** replaces the employee's current **manager** with *newManager* only if the *newManager* object is a valid manager (use **isManagerValid** to check). If *newManager* is valid, it prints "Manager has been changed correctly" then returns true. Otherwise, it prints "Manager couldn't be changed" and returns false.
- **isManager():** returns true if the employee doesn't have a manager. Otherwise, it returns false.
- **isManagerValid(manager: Employee):** returns true if the employee's **department** name matches the *manager*'s **department** name and *manager* is a manager. Otherwise, it returns false. It returns false if *manager* is NULL.
- **yearlyRaise(performance: double):** change the employee's **salary** based of the following table (Apply the performance table first then years table). After that, it increases **years** by 1. A manager will have 1000 added to the **salary** as a bonus. Then return true. If *performance* is more than 100.0 or less than 0.0, return false.

<i>Performance</i>	Change in salary
100.0 to 90.0	+15%
less than 90.0 to 75.0	+10%
less than 75.0 to 50.0	+5%
less than 50.0 to 25.0	+0%
less than 25.0	-5%

Years	Change in salary
0 to 5	+100
6 to 10	+250
11 and more	+500

If an employee (salary = 10000 and years = 5) had a 80 in performance, his salary will be 11000 because of the first table (+10%). Then the salary will be 11100 because of the second table (+100). Once the salary calculations are done, increase the year by 1.

- **toString():** returns a String with the employee's information as follows:
 - If the employee is a manager:

Manager of **IT** department. Worked for **2** years. Salary: **11000**

- If the employee is not a manager.

Employee in **Sales** department. Worked for **6** years. Salary: **5000**

Write another class **TestEmployee** (3 Points) that does the following:

1. Ask the user to type in 3 employee's salaries and departments.

2. Create 3 Employee objects (use NULL as their manager).
3. Perform 3 **yearlyRaise()** on the first employee (ask the user to type the performance for each year).
4. Try to make the first employee the manager of the other 2 employees.
5. Perform 1 **yearlyRaise()** on the second and third employee (ask the user to type the performance for each employee).
6. Print the information of all 3 employees.

Sample run:

```
Enter 3 employees' salaries and departments:
15000 IT
10000 Sales
5000 IT
Enter the first employee yearly performance 1: 90
Enter the first employee yearly performance 2: 80
Enter the first employee yearly performance 3: 10
Manager couldn't be changed.
Manager has been changed correctly.
Enter the second employee yearly performance: 50
Enter the third employee yearly performance: 30
The first employee: Manager of IT department. Worked for 3
years. Salary: 21320.75
The second employee: Manager of Sales department. Worked
for 1 years. Salary: 11600.0
The third employee: Employee in IT department. Worked for
1 years. Salary: 5100.0
```

Question 2 (5 points):

Q2) Write a Java Program that will read the Heart Beat Rate (HB) of a patient (integer). Then decide if the patient at risk or not based on the following:

- If $HB > 120$ then patient have HIGH HB.
- $80 \leq HB \leq 120$ NORMAL HB.
- Below 80 is LOW HB.

Your program should:

1. Continuously read HB and print status of the patient.
2. Keep reading HB and printing status until you get two consecutive HIGH HB or two consecutive LOW HB, if you do then you should print a warning message and stop reading HB.
3. Print the average HB of all your readings.
4. Exit the program

Name your class **Heart**.

Sample run 1:

```
Please enter patient HB 100
HB is Normal
Please enter patient HB 120
HB is Normal
Please enter patient HB 121
HB is High
Please enter patient HB 70
HB is Low
Please enter patient HB 60
HB is Low
HB is low twice in a row
The average HB is 94.2
```

Sample run 2:

```
Please enter patient HB 200
HB is High
Please enter patient HB 100
HB is Normal
Please enter patient HB 200
HB is High
Please enter patient HB 90
HB is Normal
Please enter patient HB 20
```

```
HB is Low
Please enter patient HB 30
HB is Low
HB is low twice in a row
The average HB is 106.66666666666667
```

Sample run 3:

```
Please enter patient HB 40
HB is Low
Please enter patient HB 100
HB is Normal
Please enter patient HB 200
HB is High
Please enter patient HB 200
HB is High
HB is high twice in a row
The average HB is 135.0
```

Question 3 (5 points):

Q3) Write a Java class **Point** to represent a point with x and y coordinates. The class Point has the following:

Attributes:

- double x
- double y

Methods:

- Setters and getters for x and y.
- **distance(Point p): double**. This method returns the distance between the two points: *p* and the calling object.

hint: to compute distance between two points, use the equation:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Write another class **TestPoint** that does the following:

1. Declare two **Point** objects p1 and p2
2. Read the values of x and y for both points.
3. Compute the distance between the two points using the method distance, if the distance is less than or equal to 5.0 you should print the distance and exit. Otherwise keep reading two more points and printing the distance until you get two points with distance less than or equal to 5.0.

Sample run1:

```
Enter x and y for p1 1 2
Enter x and y for p2 6 8
Distance is 7.810249675906654 which is > 5. Try again.
Enter x and y for p1 2 6
Enter x and y for p2 9 18
Distance is 13.892443989449804 which is > 5. Try again.
Enter x and y for p1 1 1
Enter x and y for p2 4 3
The two points are close.
The distance between them is 3.605551275463989 which is
less than 5
```

Sample run2:

```
Enter x and y for p1 0 0
Enter x and y for p2 0 0
The two points are close.
The distance between them is 0.0 which is less than or
equal to 5
```

Sample run3:

```
Enter x and y for p1 10 5
Enter x and y for p2 15 5
The two points are close.
The distance between them is 5.0 which is less than or
equal to 5
```