CS526 O2 – Fall 2020
Homework Assignment 1

This assignment is practice of Java basics.

Problem 1 (50 points). An incomplete Java program named *Hw1_P1_incomplete.java* is posted on the course website. You must complete the program by implementing the following requirements. Don't forget to delete "*_incomplete*" from the file name.

This program has three methods. The first method receives an array of integers and calculates the average, the minimum, and the maximum of the integers and prints them on the screen.

The signature of this method is:

public static void stats(int[ ] numbers)

The second method creates and prints a subarray of a given array. The specification of the method is:

- Signature of method:

  public static void subarray(int[ ] a, int from, int to)

- Input arguments:
  a: An array of integers
  from: The index of an element in *a* which becomes the first element in the subarray
  to: The index of an element in *a* which becomes the last element in the subarray
- Behavior:
  - A new integer array is created, which is a subarray of *a*, and it includes elements *a*[*from*] to *a*[*to*], inclusively.
  - Prints the subarray.
- There is no return value.

The third method is a main method. If you run this program with the following main method:

```
public static void main(String[] args) {

      int[] a = {15, 25, 10, 65, 30, 55, 65};
      System.out.print("\nGiven array is: ");
      for (int i=0; i<a.length-1; i++) {
            System.out.print(a[i] + ", ");
      }
      System.out.print(a[a.length - 1]);
      System.out.println();

      stats(a);
      subarray(a, 1, 4);
```

Your output should be:

```
Given array is: 15, 25, 10, 65, 30, 55, 65

average = 37.86, min = 10, max = 65

The subarray, from index 1 to index 4, is: 25, 10, 65, 30
```

Problem 2 (50 points). For this problem, first you need to write a subclass of the *Employee* class. The definition of the *Employee* class is in the *Employee.java* file and it is briefly described below:

Class Employee

Instance variables
- empId:
  - Description: Employee id of an employee
  - Type: integer
- name:
  - Description: Name of an employee
  - Type: String

Instance methods:
- getEmpId
  - No input argument
  - Returns the employee id
- getName
  - No input argument
  - Returns the employee name
- setEmpId
  - Input argument: empId of integer type
  - Existing empId is replaced with the given empId
  - No return value
- setName
  - Input argument: name of String type
  - Existing name (which may be null) is replaced with the given name
  - No return value

You need to implement the following class, which is a subclass of the *Employee* class. Define this class in a separate file named *SalariedEmployee.java*.

Class *SalariedEmployee*

- A subclass of the *Employee* class

- Instance variables
  - salary: annual salary; double type
- Instance method
  - monthlyPayment
    - No input argument
    - Returns monthly payment; double type
    - Monthly payment is calculated as salary / 12
  - employeeInfo
    - No input argument
    - Displays the empId, name, salary, and monthly payment
    - No return value

In the subclass definition, you must implement all necessary *get methods* and *set methods*.

Then, write a program, named *Hw1_P2.java*, that reads employee information from a text file and stores *SalariedEmployee* objects in an array of size 10, named *employeeArray*, and displays employees satisfying a certain criterion. Follow the instruction below:

- Write a method, named *employeesAbove*, whose behavior is specified below:

  - Signature of method:

    public static void employeesAbove (SalariedEmployee[ ] empArray, double threshold)

  - Input arguments:
    empArray: An array of *SalariedEmployee* objects
    threshold: A salary threshold
  - Behavior: Selects from *empArray* only those employees who earn more than the given threshold amount, and displays their *empId*, *name*, *salary*, and monthly payment.
  - There is no return value.

- In the main method, do the following:

  - Read information of 10 salaried employees from an input file named *employee_input.txt* and create 10 *SalariedEmployee* objects and store them in *employeeArray*. A sample input file is shown below:

    ```
    1, John, 50000
    2, Susan, 120000
    3, Yapsiong, 80000
    4, Gomez, 90000
    5, Minsky, 60000
    6, Yorst, 30000
    ```

7, Govindranad, 40000
8, Kelsey, 60000
9, Jake, 110000
10, Guanyu, 70000

- Invoke the *employeesAbove* method a few times with different threshold values and check that your output is correct. If, for example, you invoke the method with the threshold 70000, then the output should be:

```
Employees earning more than $70000:

    Employee id = 2
    Name = Susan
    Salary = 120000.00
    Monthly pay = 10000.00

    Employee id = 3
    Name = Yapsiong
    Salary = 80000.00
    Monthly pay =  6666.67

    Employee id = 4
    Name = Gomez
    Salary = 90000.00
    Monthly pay =  7500.00

    Employee id = 9
    Name = Jake
    Salary = 110000.00
    Monthly pay =  9166.67
```

Deliverable

No separate documentation is needed for the program files. However, you must include the following in your source code:

- Include the following comments above each method:
  - Brief description of the method
  - Input arguments
  - Output arguments
- Include inline comments within your source code to increase readability of your code and to help readers better understand your code.

You must submit the following files:

- *Hw1_P1.java*
- *SalariedEmployee.java*
- *Hw1_P2.java*

Combine all files into a single archive file and name it *LastName_FirstName_hw1.EXT*, where *EXT* is an appropriate archive file extension, such as *zip* or *rar*.

Grading

Problem 1:

- If your *Hw1_P1.java* program does not compile, 30 points are deducted.
- If your program compiles but causes a runtime error, 20 points are deducted.
- If there is no output or output is completely wrong, 20 points are deducted.
- If your output is partly wrong, up to 20 points are deducted.

Problem 2:

- If there is an error in the class definition of *SalariedEmployee*, up to 10 points are deducted.
- If your program does not compile, 30 points are deducted.
- If your program compiles but causes a runtime error, 20 points are deducted.
- If there is no output or output is completely wrong, 20 points are deducted.
- If your output is partly wrong, up to 20 points are deducted.