1 of 4 Paper No: 1

Question 1:

(3 marks) You have main function which is saved on folder named Q1 already. Use these classes and functions to test all your operations in this question, DO NOT EDIT ANY CODE in main function. You can add more operations/classes/interfaces to be able to answer the below question.

Design and code a class named Person that holds information about a Person. Information of a Person includes:

A string describes the name of a Person.

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class.

Design and code a deriving class named worker from Person class that holds information about a Worker. Information of a Worker includes:

- A double-value holding working hours of a Worker.
- A double-value holding the payment rate for a Worker.

Include the following member functions in your design:

- Constructors to set values for instance variables.
- Add needed operations to the class so that the main function can be run and complete the below method which is declared in Worker class, the function will be used in second test case.
 - double getIncome() use to determine salary of a person income = salary+
 income tax, where salary = payment rate * working hours and
 - income tax = 10 percent out of salary if salary > 5000
 - otherwise incometax = 5 percent out of salary

The program output might look something like:

No	Correct output	Correct output
of		_
test		
case		

about:blank 1/121

2 of 4 Paper No: 1

2	Enter name of a worker: An An Enter payment rate of a worker: 250 Enter working hours of a worker: 150 Enter TC: 2 OUTPUT: 41250.0	Enter name of a worker: An An Enter payment rate of a worker: 80 Enter working hours of a worker: 41 Enter TC: 2 OUTPUT: 3444.0
1	Enter name of a worker: An An Enter payment rate of a worker: 250 Enter working hours of a worker: 100 Enter TC: 1 OUTPUT: An An An An 100.0 250.0	

Question 2:

(4 marks) You have main function which is saved on folder named Q2 already. Use these classes and function to test all your operations in this question, DO NOT EDIT ANY CODE in main. You can add more operations/classes/interfaces to be able to answer the below questions. We had provided you:

1. Interface – IStudent which will declare some operations for an Student – DO NOT EDIT this one

2. You only need to complete the code in class MyStudent and Student.

Design and code a class named Student that holds information about an Student. Information of a Student includes:

- A string describing name of a Student.
- A string describing major of a Student

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class so that the main function can be run. Complete the method
 - o public String getName() return the name of a Student

Design and code a class named MyStudent which will implement interface IStudent and complete 2 methods which were declared in IStudent (assume that number of Students in

list "s" is always greater than 2).

- List<Student> f1(List<Student> s) -. Firstly, sort the list of Student "s" ascending by student name (alphabet order), finally return the list of 2 first students in the list "s". The sorting must ignores the case during comparison.
- int f2(ArrayList<Student> s) count and return the number of students in the list "s" who are in major of "SE".

about:blank 2/121

3 of 4 Paper No: 1

The program output might look something like:

```
Enter number of Student: 3
                                   Enter number of Student: 3
Student name: Hoa Hoa
Student major: SE
                                   Student name: Hoa Hoa
                                   Student major: SE
Student name: An An
Student major: IA
                                   Student name: An An
                                   Student major: IA
Student name: Binh Binh
Student major: SE
                                   Student name: Binh Binh
                                   Student major: SE
Enter test function (1-f1;2-f2): 1
                                   Enter test function (1-f1;2-f2): 2
OUTPUT:
An An
                                   OUTPUT:
Binh Binh
```

Question 3:

(3 marks) You have main function which is saved on folder named Q3 already. Use these classes and functions to test all your operations in this question, DO NOT EDIT ANY CODE in main function. You can add more operations/classes/interfaces to be able to answer the below question.

Design and code a class named Flight, that holds information of a Flight, including:

- A string describes route of a flight as the format of "departure-destination", for example HAN-MNL or CDG-SGN...
- · A whole number describes fare of a flight.

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class and complete below method which is declared in Flight class, the function will be used in second test case.
 - String getDeparture() this function return value of departure of a route.

The program output might look something like:

about:blank 3/121

4 of 4 Paper No: 1

No of test	Correct output	
case		
2	Enter flight route: ICN-HAN	
	Enter flight fare: 250	
	Enter TC: 2	
	OUTPUT:	
	ICN	
1	Enter flight route: ICN-HAN	
-	Enter flight fare: 250	
	Enter TC: 1	
	OUTPUT:	
	ICN-HAN 250	

about:blank 4/121

1 of 4 Paper No: 2

Question 1:

(3 marks) You have main function which is saved on folder named Q1 already. Use these classes and functions to test all your operations in this question, DO NOT EDIT ANY CODE in main function. You can add more operations/classes/interfaces to be able to answer the below question.

Design and code a class named Person that holds information about a Person. Information of a Person includes:

· A string describes the name of a Person.

Include the following member functions in your design:

- Constructors to set values for instance variables.
- Add needed operations to the class.

Design and code a deriving class named worker from Person class that holds information about a worker. Information of a worker includes:

- A double-value holding working hours of a Worker.
- A double-value holding the payment rate for a Worker.

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class so that the main function can be run and complete the below method which is declared in Worker class, the function will be used in second test case
 - double getIncome() use to determine salary of a person income = salary+
 income tax, where salary = payment rate * working hours and
 - income tax = 10 percent out of salary if salary > 5000
 - otherwise incometax = 5 percent out of salary

The program output might look something like:

No	Correct output	Correct output
of		_
test		
case		

about:blank 5/121

2 of 4 Paper No: 2

2	Enter name of a worker: An An Enter payment rate of a worker: 250 Enter working hours of a worker: 150 Enter TC: 2 OUTPUT: 41250.0	Enter name of a worker: An An Enter payment rate of a worker: 80 Enter working hours of a worker: 41 Enter TC: 2 OUTPUT: 3444.0
1	Enter name of a worker: An An Enter payment rate of a worker: 250 Enter working hours of a worker: 100 Enter TC: 1 OUTPUT: An An An An 100.0 250.0	

Question 2:

(4 marks) You have main function which is saved on folder named Q2 already. Use these classes and function to test all your operations in this question, DO NOT EDIT ANY CODE in main. You can add more operations/classes/interfaces to be able to answer the below questions. We had provided you:

1. Interface – IStudent which will declare some operations for an Student – DO NOT EDIT this one

2. You only need to complete the code in class MyStudent and Student.

Design and code a class named Student that holds information about an Student. Information of a Student includes:

- A string describing name of a Student.
- A string describing major of a Student

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class so that the main function can be run. Complete the method
 - o public String getName() return the name of a Student

Design and code a class named MyStudent which will implement interface IStudent and complete 2 methods which were declared in IStudent (assume that number of Students in

list "s" is always greater than 2).

- List<Student> f1(List<Student> s) -. Firstly, sort the list of Student "s" ascending by student name (alphabet order), finally return the list of 2 first students in the list "s". The sorting must ignores the case during comparison.
- int f2(ArrayList<Student> s) count and return the number of students in the list "s" who are in major of "SE".

about:blank 6/121

```
3 of 4 Paper No: 2
```

The program output might look something like:

```
Enter number of Student: 3
                                   Enter number of Student: 3
Student name: Hoa Hoa
Student major: SE
                                   Student name: Hoa Hoa
                                   Student major: SE
Student name: An An
Student major: IA
                                   Student name: An An
                                   Student major: IA
Student name: Binh Binh
Student major: SE
                                   Student name: Binh Binh
                                   Student major: SE
Enter test function (1-f1;2-f2): 1
                                   Enter test function (1-f1;2-f2): 2
OUTPUT:
An An
                                   OUTPUT:
Binh Binh
```

Question 3:

(3 marks) You have main function which is saved on folder named Q3 already. Use these classes and functions to test all your operations in this question, DO NOT EDIT ANY CODE in main function. You can add more operations/classes/interfaces to be able to answer the below question.

Design and code a class named Flight, that holds information of a Flight, including:

- A string describes route of a flight as the format of "departure-destination", for example HAN-MNL or CDG-SGN...
- A whole number describes fare of a flight.

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class and complete below method which is declared in Flight class, the function will be used in second test case.
 - String getDeparture() this function return value of departure of a route.

The program output might look something like:

about:blank 7/121

4 of 4 Paper No: 2

No of test case	Correct output	
2	Enter flight route: ICN-HAN	
	Enter flight fare: 250	
	Enter TC: 2	
	OUTPUT:	
	ICN	
1	Enter flight route: ICN-HAN	
-	Enter flight fare: 250	
	Enter TC: 1	
	OUTPUT:	
	ICN-HAN 250	

about:blank 8/121

1 of 4 Paper No: 3

Question 1:

(3 marks) You have main function which is saved on folder named Q1 already. Use these classes and functions to test all your operations in this question, DO NOT EDIT ANY CODE in main function. You can add more operations/classes/interfaces to be able to answer the below question.

Design and code a class named Person that holds information about a Person. Information of a Person includes:

· A string describes the name of a Person.

Include the following member functions in your design:

- Constructors to set values for instance variables.
- Add needed operations to the class.

Design and code a deriving class named worker from Person class that holds information about a worker. Information of a worker includes:

- A double-value holding working hours of a Worker.
- A double-value holding the payment rate for a Worker.

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class so that the main function can be run and complete the below method which is declared in Worker class, the function will be used in second test case
 - double getIncome() use to determine salary of a person income = salary+
 income tax, where salary = payment rate * working hours and
 - income tax = 10 percent out of salary if salary > 5000
 - otherwise incometax = 5 percent out of salary

The program output might look something like:

No	Correct output	Correct output
of		_
test		
case		

about:blank 9/121

2 of 4 Paper No: 3

2	Enter name of a worker: An An Enter payment rate of a worker: 250 Enter working hours of a worker: 150 Enter TC: 2 OUTPUT: 41250.0	Enter name of a worker: An An Enter payment rate of a worker: 80 Enter working hours of a worker: 41 Enter TC: 2 OUTPUT: 3444.0
1	Enter name of a worker: An An Enter payment rate of a worker: 250 Enter working hours of a worker: 100 Enter TC: 1 OUTPUT: An An An An 100.0 250.0	

Question 2:

(4 marks) You have main function which is saved on folder named Q2 already. Use these classes and function to test all your operations in this question, DO NOT EDIT ANY CODE in main. You can add more operations/classes/interfaces to be able to answer the below questions. We had provided you:

1. Interface – IStudent which will declare some operations for an Student – DO NOT EDIT this one

2. You only need to complete the code in class MyStudent and Student.

Design and code a class named Student that holds information about an Student. Information of a Student includes:

- A string describing name of a Student.
- A string describing major of a Student

Include the following member functions in your design:

- · Constructors to set values for instance variables.
- Add needed operations to the class so that the main function can be run. Complete the method
 - o public String getName() return the name of a Student

Design and code a class named MyStudent which will implement interface IStudent and complete 2 methods which were declared in IStudent (assume that number of Students in

list "s" is always greater than 2).

- List<Student> f1(List<Student> s) -. Firstly, sort the list of Student "s" ascending by student name (alphabet order), finally return the list of 2 first students in the list "s". The sorting must ignores the case during comparison.
- int f2(ArrayList<Student> s) count and return the number of students in the list "s" who are in major of "SE".

about:blank 10/121

about:blank 11/121

about:blank 12/121

about:blank 13/121

about:blank 14/121

about:blank 15/121

about:blank 16/121

about:blank 17/121

about:blank 18/121

about:blank 19/121

about:blank 20/121

about:blank 21/121

about:blank 22/121

about:blank 23/121

about:blank 24/121

about:blank 25/121

about:blank 26/121

about:blank 27/121

about:blank 28/121

about:blank 29/121

about:blank 30/121

about:blank 31/121

about:blank 32/121

about:blank 33/121

about:blank 34/121

about:blank 35/121

about:blank 36/121

about:blank 37/121

about:blank 38/121

about:blank 39/121

about:blank 40/121

about:blank 41/121

about:blank 42/121

about:blank 43/121

about:blank 44/121

about:blank 45/121

about:blank 46/121

about:blank 47/121

about:blank 48/121

about:blank 49/121

about:blank 50/121

about:blank 51/121

about:blank 52/121

about:blank 53/121

about:blank 54/121

about:blank 55/121

about:blank 56/121

about:blank 57/121

about:blank 58/121

about:blank 59/121

about:blank 60/121

about:blank 61/121

about:blank 62/121

about:blank 63/121

about:blank 64/121

about:blank 65/121

about:blank 66/121

about:blank 67/121

about:blank 68/121

about:blank 69/121

about:blank 70/121

about:blank 71/121

about:blank 72/121

about:blank 73/121

about:blank 74/121

about:blank 75/121

about:blank 76/121

about:blank 77/121

about:blank 78/121

about:blank 79/121

about:blank 80/121

about:blank 81/121

about:blank 82/121

about:blank 83/121

about:blank 84/121

about:blank 85/121

about:blank 86/121

about:blank 87/121

about:blank 88/121

about:blank 89/121

about:blank 90/121

about:blank 91/121

about:blank 92/121

about:blank 93/121

about:blank 94/121

about:blank 95/121

about:blank 96/121

about:blank 97/121

about:blank 98/121

about:blank 99/121

about:blank 100/121

about:blank 101/121

about:blank 102/121

about:blank 103/121

about:blank 104/121

about:blank 105/121

about:blank 106/121

about:blank 107/121

about:blank 108/121

about:blank 109/121

about:blank 110/121

about:blank 111/121

about:blank 112/121

about:blank 113/121

about:blank 114/121

about:blank 115/121

about:blank 116/121

about:blank 117/121

about:blank 118/121

about:blank 119/121

about:blank 120/121

about:blank 121/121