King Saud University

College of Computer and Information Sciences
Computer Science Department

Course Code:	CSC 113
Course Title:	Computer Programming II
Semester:	Spring 2020
Exercises Cover Sheet:	**Midterm 1 Exam**

Student Name:	
Student ID:	
Student Section No.	

Tick the Relevant	Computer Science B.Sc. Program ABET Student Outcomes	Question No. Relevant Is Hyperlinked	Covering %
X	**a) Apply knowledge of computing and mathematics appropriate to the computer science;**		
	b) Analyze a problem, and identify and define the computing requirements appropriate to its solution		
X	**c) Design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;**		
X	**d) Function effectively on teams to accomplish a common goal;**		
	e) Understanding of professional, ethical, legal, security, and social issues and responsibilities;		
	f) Communicate effectively with a range of audiences;		
	g) Analyze the local and global impact of computing on individuals, organizations and society;		
	h) Recognition of the need for, and an ability to engage in, continuing professional development;		
X	**i) Use current techniques, skills, and tools necessary for computing practices.**		
	j) Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;		
	k) Apply design and development principles in the construction of software systems of varying complexity;		

Exercise 1:

Give the output of the following program.

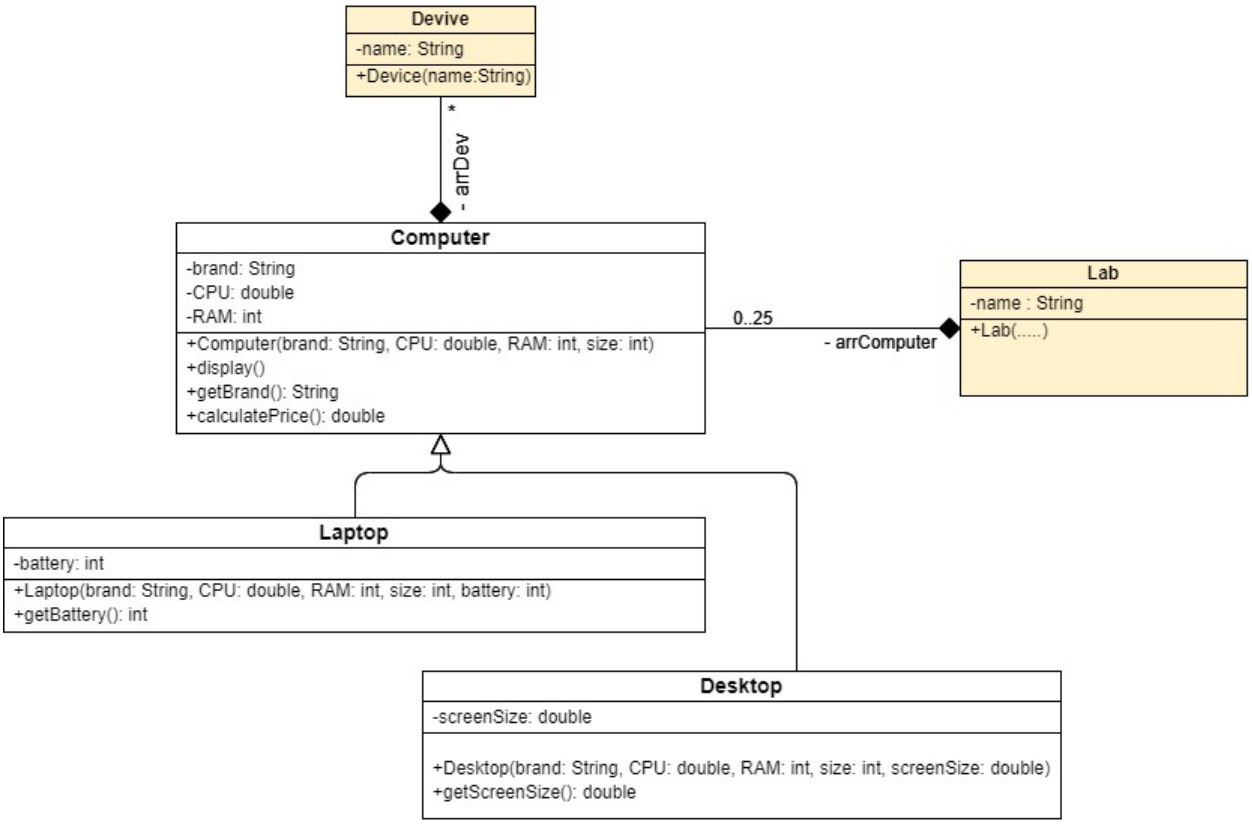
```
public class Flight {
	private String flightNum;
	protected int dist;

	public Flight() {
		flightNum = "Unknown";   dist = 500;
		System.out.println ("The Flight is Created");
	}
	public Flight (String flightNum, int dist) {
		this.flightNum = flightNum;
		this.dist=dist;
	}
	public void display() {
	     System.out.println ("Flight number: " +  flightNum + " distance: " +  dist );
	}
	public int cost (){  return 200;  }
 }
public class LongDistanceFlight extends Flight {
	protected int rate;

	public LongDistanceFlight () { rate = 2;  }

	public LongDistanceFlight (String flightNum, int dist, int r) {
		super(flightNum, dist);
		rate = r;
	}
	public void display () {
		System.out.println ("Long Distance Flight ");
		super.display();
	}
	public int cost(){
		if (dist < 1000 )
			System.out.println("Warning: Distance Less Than 1000 Km");
		return (super.cost()+ dist*rate);
	}
 }
public class InternationalFlight extends LongDistanceFlight{
	protected int airportFee;

	public InternationalFlight(String s, int d, int r, int f) {
		super(s,d,r);
		airportFee = f;
	}
	public InternationalFlight(int f) { airportFee = f;   }

	public void display()  {
		System.out.println ("International Flight ");
		super.display();
		System.out.println (cost());
	}

	public int cost(){ return (super.cost()+airportFee); }
 }
```

```
public class TestFlights {
      public static void main(String[] args) {
            int i;
            Flight [] flightList = new Flight[2];

            flightList[0]  = new InternationalFlight("SV3875", 1000, 3, 100);
            flightList[1]  = new InternationalFlight(500);

            for (i=0; i< 2; i++) {
                  System.out.println("Iteration " + (i+1));
                  flightList[i].display();
            }// end for
      }// end main
 }
```

Answer: ……../5

```
The Flight is Created      ……../0.5
Iteration 1
International Flight ……../0.5
Long Distance Flight ……../0.5
Flight number: SV3875 distance: 1000 ……../0.5
3300 ……../0.5
Iteration 2
International Flight ……../0.5
Long Distance Flight ……../0.5
Flight number: Unknown distance: 500 ……../0.5
Warning: Distance Less Than 1000 Km ……../0.5
1700 ……../0.5
```

Exercise 2:



The class Device:

- Attributes:
  - ***name:*** the name of the Device.
- Methods:
  - ***Device (…)***: constructor.

The class Computer:

- Attributes:
  - ***brand:*** the brand name of the Computer.
  - ***CPU***: the CPU speed (in GHz) of the Computer.
  - ***RAM***: the RAM size (in GB) of the Computer.
- Methods:
  - ***Computer (…)***: constructor.
  - ***display( )*:** displays all the attributes of the Computer, Laptop or Desktop.
  - ***getBrand()*: It** returns the brand name of the Computer .
  - ***calculatePrice()*:** It returns the price of the computer that is calculated as following:
    - For a ***Laptop***: the price = RAM * 400 + battery capacity * 100.
    - For a ***Desktop***: the price = CPU * 1200 + size of the monitor * 150.

The class Laptop:

- Attributes:

- ***battery:*** the capacity of the battery (in **mAh**) of the Laptop.

- Methods:
  - ***Laptop (…)***: constructor.
  - ***getBattery( ):*** returns the capacity of the battery of the Laptop.

The class Desktop:

- Attributes:
  - ***screenSize:*** the size of the monitor of the Desktop.
- Methods:
  - ***Desktop (…)***: constructor.
  - ***getScreeSize( ):*** returns the size of the monitor of the Desktop.

QUESTION: Implement using Java:

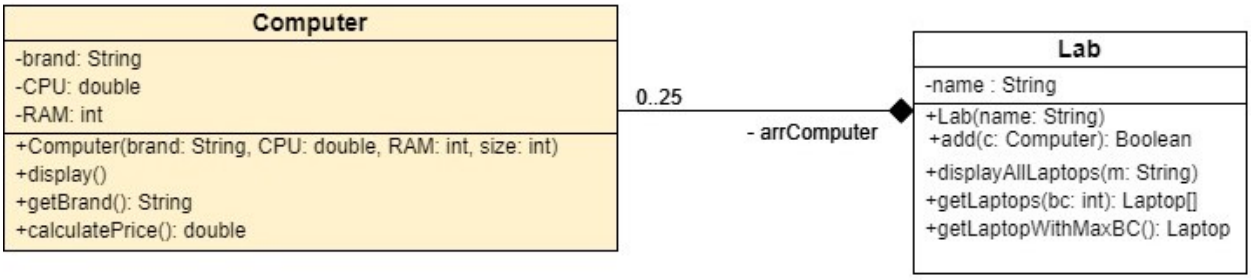
1. The class ***Computer***,
2. The class ***Laptop.***

```
1.  public abstract class Computer { ……../1      ……../12
2.        private String brand; //: the brand name of the Computer.
3.        protected double CPU; //: the CPU speed (in GHz) of the Computer.
4.        protected int RAM;
5.
6.        private Device arrDev[];  ……../0.5
7.        private int nbDev;        ……../0.5
8.
9.        public Computer(String b, double cpu, int ram, int size) { ……../2
10.             brand = b;
11.             CPU = cpu;
12.             RAM = ram;
13.
14.             arrDev = new Device[size];       ……../1
15.             nbDev = 0;                       ……../1
16.       }
17.       public Computer(Computer c) { ……../5
18.             brand = c.brand;
19.             CPU = c.CPU;
20.             RAM = c.RAM;
21.
22.             arrDev = new Device[c.arrDev.length];  ……../1
23.             for (int i = 0; i < c.nbDev; i++) {    ……../1
24.                   arrDev[i] = new Device(c.arrDev[i]);   ……../1+1
25.             }
26.             nbDev = c.nbDev;    ……../1
27.       }
28.
29.
```

```
30.         /* Other implementation of the copy constructor
31.          * public Computer(Computer c) {
32.          *           this(c.brand, c.CPU, c.RAM, c.arrDev.length);
33.          *
34.          *           for (int i = 0; i < c.nbDev; i++) {
35.          *                 arrDev[i] = new Device(c.arrDev[i]);
36.          *           }
37.          *           nbDev = c.nbDev;
38.          * }
39.          */
40.
41.         public void display() {  ……../1
42.               System.out.println(brand + " --- " + CPU + " --- " + RAM);
43.         }
44.
45.         public String getBrand() { ……../1
46.               return brand;
47.         }
48.
49.         public abstract double calculatePrice() ;  ……../1
50. }



public class Laptop extends Computer { ……../1     ……../9
            private int battery;

            public Laptop(String brand, double cpu, int ram, int size, int b) { …/2
                  super(brand, cpu, ram, size); ……../1
                  battery = b; ……../1
            }

            public Laptop(Laptop x) { ……../2
                  super(x); ……../1
                  battery = x.battery; ……../1
            }

            public int getBattery() { ……../1
                  return battery;
            }

            public double calculatePrice() { ……../1
                  return (RAM * 400 + battery * 100);
            }

            public void display() { ……../2
                  super.display(); ……../1
                  System.out.println(battery); ……../1
            }
}
```

Exercise 3:

Let's consider the same class ***Computer*** and its subclasses as described in Exercise 2.



The class Lab:

- Attributes:
  - ***name:*** name of the Lab.
- Methods:
  - ***Lab (…)***: constructor.
  - ***add(c: Computer)***this method adds the Computer ***c*** to the Lab. It returns ***true*** if the Computer ***c*** is inserted successfully. Otherwise, it returns ***false***;
  - ***displayAllLaptops(m: String)*:** It displays all Laptops with brand name ***m***.
  - ***getLaptops(bc:*** int ***):*** It returns an array containing all Laptops with a battery capacity greater than or equal to ***bc***.
  - ***getLaptopWithMaxBC(***): It returns the Laptop with brand name "***Dell***" that has the maximum battery capacity. In case of several Laptops, it returns the first one met.

QUESTION: Implement using Java the class ***Lab.***

```
public class Lab {  ……../34
	private String name;

	private Computer arrComputer[];   ……../0.5
	private int nbComp;               ……../0.5

	public Lab(String s, int size) {   ……../2
		name = s;

		arrComputer = new Computer[size];   ……../1
		nbComp = 0;          ……../1
	}
```

```
	public boolean add(Computer c) {    ……../8
		if (nbComp < arrComputer.length) {  ……../1
			if (c instanceof Laptop)  ……../1
				arrComputer[nbComp] = new Laptop( (Laptop) c ); ……../1+1
			else
				arrComputer[nbComp] = new Desktop( (Desktop) c ); ……../1+1

			nbComp++;            ……../1
			return true;         ……../0.5
		}
		else
			return false;        ……../0.5
	}

	public void displayAllLaptops(String m) { ……../4
		for (int i = 0; i < nbComp; i++) { ……../1
			if (   arrComputer[i] instanceof Laptop && ……../1
				arrComputer[i].getBrand().equals(m))    ……../1
					arrComputer[i].display(); ……../1
		}
	}

	public Laptop[] getLaptops(int bc) { ……../9
		Laptop[] res = new Laptop[nbComp]; ……../1
		int count = 0;       ……../1

		for (int i = 0 ; i < nbComp ; i++) { ……../1
			if (arrComputer[i] instanceof Laptop && ……../1
				((Laptop)arrComputer[i]).getBattery() >= bc) { ……../1+1
					res[count] = (Laptop) arrComputer[i]; ……../1+1
					count ++;     ……../1
			}
		}

		return res;
	}

	public Laptop getLaptopWithMaxBC() { ……../10
		Laptop max = null;  ……../1

		for (int i = 0; i < nbComp; i++) { ……../1
			if (   arrComputer[i] instanceof Laptop && ……../1
				arrComputer[i].getBrand().equals("Dell")) {  ……../1
				if (max == null ||    ……../1
				((Laptop) arrComputer[i]).getBattery() > max.getBattery())
				……../1+1
				{
					max = (Laptop) arrComputer[i]; ……../1+1
				}
			}
		}
		return max; ……../1
	}
}
```