Lab Assignment 08

Course Code:	CSE111
Course Title:	Programming Language II
Topic:	Review and Polymorphism
Number of Tasks:	11

[You are not allowed to change the driver codes of any of the tasks]

Task 1

Design the **Vaccine** and **Person** class so that the following expected output is generated.
[N.B: Students will get vaccines on a priority basis. So, age doesn't matter for students. All **attributes of Vaccine** class should be **Private**.**]**

Driver Code	Output
| ```
public class VaccineTester {
  public static void main(String[] args) {
    Vaccine astra = new Vaccine("AstraZeneca", "UK", 60);
    Vaccine modr = new Vaccine("Moderna", "UK", 30);
    Vaccine sin = new Vaccine("Sinopharm", "China", 30);

    Person p1 = new Person("Bob", 21, "Student");
    System.out.println("================");
    p1.pushVaccine(astra);
    System.out.println("================");
    p1.showDetail();
    System.out.println("================");
    p1.pushVaccine(sin, "2nd Dose");
    System.out.println("================");
    p1.pushVaccine(astra, "2nd Dose");
    System.out.println("================");
    p1.showDetail();
    System.out.println("================");
    p1.pushVaccine(astra, "2nd Dose");
    System.out.println("================");
    p1.showDetail();
    System.out.println("================");

    Person p2 = new Person("Carol", 23, "Actor");
    System.out.println("================");
    p2.pushVaccine(sin);
    System.out.println("================");

    Person p3 = new Person("David", 34);
    System.out.println("================");
    p3.pushVaccine(modr, "2nd Dose");
    System.out.println("================");
    p3.pushVaccine(modr, "1st Dose");
    System.out.println("================");
    p3.showDetail();
    System.out.println("================");
    p3.pushVaccine(modr, "2nd Dose");
  }
}
``` | ```
================
1st dose done for Bob
================
Name: Bob Age: 21 Type: Student
Vaccine name: AstraZeneca
1st dose: Given
2nd dose: Please come after 60 days
================
Sorry Bob, you can't take 2 different
vaccines
================
2nd dose done for Bob
================
Name: Bob Age: 21 Type: Student
Vaccine name: AstraZeneca
1st dose: Given
2nd dose: Given
================
Sorry Bob, you already received both doses.
================
Name: Bob Age: 21 Type: Student
Vaccine name: AstraZeneca
1st dose: Given
2nd dose: Given
================
================
Sorry Carol, Minimum age for taking vaccines
is 25 years now.
================
================
Sorry David, invalid dose request.
================
1st dose done for David
================
Name: David Age: 34 Type: General Citizen
Vaccine name: Moderna
1st dose: Given
2nd dose: Please come after 30 days
================
2nd dose done for David
``` |

Task 2

We know that Nike has opened their official outlets in Bangladesh. So let's construct a **NikeBD** class so that they can keep track of their inventory and sales here.
[**Hint:** Only 3 types of products are available: "Jordan", "Cortez" and "Kobe"]

Driver Code	Output
```public class NikeTester {``` ...	...

```
public class NikeTester {
  public static void main(String[] args) {
    System.out.println("==========1==========");
    NikeBD.status();
    NikeBD dhaka = new NikeBD("Dhaka Banani");
    NikeBD chittagong = new NikeBD("Chittagong GEC");
    System.out.println("==========2==========");
    dhaka.details();
    System.out.println("==========3==========");
    chittagong.details();
    System.out.println("==========4==========");
    dhaka.restockProducts("Jordan", 200);
    System.out.println("==========5==========");
    String [] products = {"Jordan", "Cortez", "Kobe"};
    int [] qty = {1200, 200, 200};
    String [] products2 = {"Jordan", "Cortez", "Kobe"};
    int [] qty2 = {1200, 250, 100};
    dhaka.restockProducts(products, qty);
    System.out.println("==========6==========");
    chittagong.restockProducts(products2, qty2);
    System.out.println("==========7==========");
    NikeBD.status();
    System.out.println("==========8==========");
    dhaka.details();
    System.out.println("==========9==========");
    chittagong.details();
    dhaka.productSold("Jordan", 760, "Cortez", 90);
    chittagong.productSold("Jordan", 520, "Kobe", 70);
    System.out.println("==========10==========");
    NikeBD.status();
    System.out.println("==========11==========");
    chittagong.details();
  }
}
```

Output:

```
==========1==========
Nike Bangladesh Status:
Branches Opened: 0
Currently Stocked: Air Jordan: 0, Cortez: 0,
Zoom Kobe: 0
Sold: 0
==========2==========
Nike Dhaka Banani outlet:
Products Currently Stocked: Air Jordan: 0,
Cortez: 0, Zoom Kobe: 0
Sold: 0
==========3==========
Nike Chittagong GEC outlet:
Products Currently Stocked: Air Jordan: 0,
Cortez: 0, Zoom Kobe: 0
Sold: 0
==========4==========
==========5==========
==========6==========
==========7==========
Nike Bangladesh Status:
Branches Opened: 2
Currently Stocked: Air Jordan: 0, Cortez:
450, Zoom Kobe: 0
Sold: 0
==========8==========
Nike Dhaka Banani outlet:
Products Currently Stocked: Air Jordan: 0,
Cortez: 200, Zoom Kobe: 0
Sold: 0
==========9==========
Nike Chittagong GEC outlet:
Products Currently Stocked: Air Jordan: 0,
Cortez: 250, Zoom Kobe: 0
Sold: 0
==========10==========
Nike Bangladesh Status:
Branches Opened: 2
Currently Stocked: Air Jordan: 0, Cortez:
360, Zoom Kobe: 0
Sold: 90
==========11==========
Nike Chittagong GEC outlet:
Products Currently Stocked: Air Jordan: 0,
Cortez: 250, Zoom Kobe: 0
Sold: 0
```

Task 3

Design the child class **Striker** and **Defender** that inherits from the Football class so that the given output matches with the output generated by the driver code.

Parent Class

```java
public class Football {
  public String name;
  public int age;
  public int stamina;

  public Football(String name, int age, int stamina) {
    this.name = name;
    this.age = age;
    this.stamina = stamina;
  }

  public void display() {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Stamina: " + stamina);
  }

  public void calculatePerformance() {
    System.out.println("Performance is not defined yet");
  }
}
```

Driver Code	Output

<pre>public class FootballTester {
  public static void main(String[] args) {
    Striker ronaldo = new Striker("Ronaldo", 39, 90, 901, 1000);
    Defender ramos = new Defender("Ramos", 38, 85, 1000, 100);
 	
    System.out.println("1========");
    ronaldo.display();
    System.out.println("2========");
    ronaldo.calculatePerformance();
    System.out.println("3========");
    ramos.display();
    System.out.println("4========");
    ramos.calculatePerformance();
  }
}</pre>
	<pre>1========
	Name: Ronaldo
	Age: 39
	Stamina: 90
	Goals: 901
	Shots on target: 1000
	2========
	Performance: 0.901
	3========
	Name: Ramos
	Age: 38
	Stamina: 85
	Tackles: 1000
	Interceptions: 100
	4========
	Performance: 0.1</pre>

Task 4

Design the **Nokia** class derived from the Mobile class so that the following output is produced.

Parent Class

```java
class Mobile {
  public String model;
  public String IMEICode;
  public boolean simCardStatus;

  public Mobile(String model, String IMEICode, boolean simCardStatus) {
    this.model = model;
    this.IMEICode = IMEICode;
    this.simCardStatus = simCardStatus;
    System.out.println("Model " + model + " is manufactured.");
  }

  public String getCountryName(String countryCode) {
    if (countryCode.equals("880")) {
      return "Bangladesh";
    } else if (countryCode.equals("455")) {
      return "USA";
    }
    return null;
  }

  public void activateSimCard() {
    if (!simCardStatus) {
      simCardStatus = true;
      System.out.println("SIM card is activated successfully.");
    }
  }

  @Override
  public String toString() {
    return "Mobile Phone Detail:\nModel: " + model + "\nIMEICode: " + IMEICode + "\nSIM Card Status: " +
simCardStatus;
  }
}




//Driver code below
```

Driver Code	Output
| ```java
public class MobileTester {
  public static void main(String[] args) {
    Nokia N3110 = new Nokia("N3110", true, "IMEI-102", 0);
    System.out.println(N3110);
    System.out.println("1=======================");
    Nokia N1100 = new Nokia("N1100", false, "IMEI-124", 100);
    System.out.println(N1100);
    System.out.println("2=======================");
    System.out.println(N3110.dialCall("88017196xxxx"));
    System.out.println("3=======================");
    N3110.rechargeSIMCard(200);
    N1100.rechargeSIMCard(300);
    System.out.println("4=======================");
    System.out.println(N3110.dialCall("88017196xxxx"));
    System.out.println("5=======================");
    System.out.println(N1100.dialCall("45517196xxxx"));
    System.out.println("6=======================");
    N1100.activateSimCard();
    System.out.println("7=======================");
    System.out.println(N1100.dialCall("45517196xxxx"));
    System.out.println("8=======================");
    System.out.println(N1100.dialCall("96617196xxxx"));
  }
}
``` | Model N3110 is manufactured.<br>Mobile Phone Detail:<br>Model: N3110<br>IMEICode: IMEI-102<br>SIM Card Status: true<br>Balance: 0.0 TK<br>1=======================<br>Model N1100 is manufactured.<br>Mobile Phone Detail:<br>Model: N1100<br>IMEICode: IMEI-124<br>SIM Card Status: false<br>Balance: 100.0 TK<br>2=======================<br>Insufficient balance! Please<br>recharge.<br>3=======================<br>Recharge successful! Current<br>balance 200.0 TK.<br>Recharge successful! Current<br>balance 400.0 TK.<br>4=======================<br>Dialing the number 88017196xxxx to<br>Bangladesh region.<br>5=======================<br>No SIM card available! Please<br>check the SIM card connectivity.<br>6=======================<br>SIM card is activated<br>successfully.<br>7=======================<br>Dialing the number 45517196xxxx to<br>USA region.<br>8=======================<br>Dialing is not allowed in this<br>region. |

Task 5

Design the **Dragon** class and **Phoenix** class derived from the MagicalCreature class so that the following output is produced.

Parent Class

```java
public class MagicalCreature {
  public String name;
  public int age;
  public MagicalCreature(String name, int age) {
    this.name = name;
    this.age = age;
  }
  public void makeSound() {
    System.out.println(name + " makes a magical sound.");
  }
  public void displayInfo() {
    System.out.println("Name: " + name + "\nAge: " + age);
  }
  public void performMagic() {
    System.out.println(name + " performs a generic magic.");
  }
}
```

Driver Code	Output
| ```java
public class MagicalTester {
  public static void main(String[] args) {
    Dragon drake = new Dragon("Drake", 500, 75);
    Phoenix fawkes = new Phoenix("Fawkes", 200, 5);
    drake.displayInfo();
    drake.makeSound();
    drake.performMagic();
    drake.fly();
    System.out.println("=====================");
    fawkes.displayInfo();
    fawkes.makeSound();
    fawkes.performMagic();
    fawkes.regenerate();
  }
}
``` | Name: Drake<br>Age: 500<br>Drake roars with a fiery breath!<br>Drake breathes fire with power level: 75<br>Drake flies through the sky.<br>=====================<br>Name: Fawkes<br>Age: 200<br>Fawkes sings an enchanting song.<br>Fawkes is reborn with 5 rebirth cycles.<br>Fawkes regenerates its body in a burst of flames. |

Task 6

Design the **Bondhus** class derived from the SocialMedia class so that the following output is produced.

<table>
<tr><td colspan="2" align="center">Parent Class</td></tr>
<tr><td colspan="2">

```java
public class SocialMedia{
  public String userName;
  public String email;

  public SocialMedia(String name, String mail){
    userName = name;
    email = mail;
  }
  @Override
  public String toString() {
    return userName + "'s profile:"+ "\nUser Name: " + userName + "\nEmail:" + email;
  }
}
```

</td></tr>
<tr><td align="center">Driver Code</td><td align="center">Output</td></tr>
<tr><td>

```java
public class SocialMediaTester{
  public static void main(String []args){
    Bondhus f1 = new Bondhus("Sheldon", "sheldon@qmail.com");
    Bondhus f2 = new Bondhus("Penny", "penny@qmail.com");
    Bondhus f3 = new Bondhus("Leonard", "leonard@qmail.com");
    System.out.println("1--------------------------");
    f1.showSentbox();
    System.out.println("2--------------------------");
    f2.showSentbox();
    System.out.println("3--------------------------");
    f2.sendMessage("Hi");
    f2.sendMessage("Hello");
    f2.sendMessage("NiHao");
    f3.sendMessage("Hola");
    f3.sendMessage("Sheldon, please.");
    System.out.println("4--------------------------");
    f2.showSentbox();
    System.out.println("5--------------------------");
    f1.showSentbox();
    System.out.println("6--------------------------");
    f1.sendMessage("Bazinga!");
    f2.sendMessage("Well, duh!");
    f3.showSentbox();
    System.out.println("7--------------------------");
    f2.showSentbox();
    f2.sendMessage("Bye.");
    f2.sendMessage("Oh! No");
    System.out.println("8--------------------------");
    f1.showSentbox();
    System.out.println("9--------------------------");
    System.out.println(f1);
    System.out.println("10--------------------------");
    System.out.println(f2);
  }
```

</td><td>

```
1--------------------------
Sheldon's Sentbox:
No sent messages.
2--------------------------
Penny's Sentbox:
No sent messages.
3--------------------------
4--------------------------
Penny's Sentbox:
Hi
Hello
NiHao
5--------------------------
Sheldon's Sentbox:
No sent messages.
6--------------------------
Leonard's Sentbox:
Hola
Sheldon, please.
7--------------------------
Penny's Sentbox:
Hi
Hello
NiHao
Well, duh!
Sentbox is full.
8--------------------------
Sheldon's Sentbox:
Bazinga!
9--------------------------
Sheldon's profile:
User Name: Sheldon
Email:sheldon@qmail.com
Messages Sent: 1
```

</td></tr>
</table>

Task 7

Write the **Mango** and the **Jackfruit** classes derived from Fruit class so that the following code generates the output below:

Parent Class

```java
public class Fruit{
  private boolean formalin = false;
  private String name = "";
  public Fruit(boolean formalin, String name){
    this.formalin = formalin;
    this.name = name;
  }
  public String getName(){
    return name;
  }
  public boolean hasFormalin(){
    return formalin;
  }
}
```

Driver Code	Output
| ```java
public class FruitTester{
  public static void testFruit(Fruit f){
    System.out.println("----Printing Detail-----");
    if(f.hasFormalin()){
      System.out.println("Do not eat the "+f.getName()+".");
      System.out.println(f);
    }else{
      System.out.println("Eat the "+f.getName()+".");
      System.out.println(f);
    }
  }
  public static void main(String [] args){
    Mango m = new Mango();
    testFruit(m);
    Jackfruit j = new Jackfruit();
    testFruit(j);
  }
}
``` | ----Printing Detail-----<br><br>Do not eat the Mango.<br><br>Mangos are bad for you<br><br>----Printing Detail-----<br><br>Eat the Jackfruit.<br><br>Jackfruits are good for you |

Task 8

Write the **CSEStudent** and **CSE111Student** classes derived from Student class so that the following code generates the output below

Parent Class

```java
public class Student{
  public String msg = "I love BU";
  public String shout(){
    return msg;
  }
}
```

Driver Code	Output

<pre>public class StudentTester{
  public static void printShout(Student s){
    System.out.println("------------------");
    System.out.println(s.msg);
    System.out.println(s.shout());
  }
  public static void main(String [] args){
    Student s = new Student();
    CSEStudent cs = new CSEStudent();
    CSE111Student cs111 = new CSE111Student();
    System.out.println(s.msg);
    System.out.println(cs.msg);
    System.out.println(cs111.msg);
    printShout(s);
    printShout(cs);
    printShout(cs111);
  }
}</pre>
	<pre>I love BU
	I want to transfer to CSE
	I love Java Programming
	------------------ 	
	I love BU
	I love BU
	------------------ 	
	I love BU
	I want to transfer to CSE
	------------------ 	
	I love BU
	I love Java Programming</pre>

Task 9

1	`public class Trace {`
2	`  public int sum, temp;`
3	`  public Trace(int sum, int temp){`
4	`    this.sum = sum;`
5	`    this.temp = temp;`
6	`  }`
7	`}`
8	`class Quiz5{`
9	`  public int sum = 12, x = 2, y = 6;`
10	`  public Trace trace;`
11	`  public Quiz5(Trace t){`
12	`    trace = t;`
13	`    int x = trace.temp + y;`
14	`    sum = sum + (t.sum--) + y;`
15	`    System.out.println(trace.sum + " " + sum + " " + x);`
16	`    sum -= 10;`
17	`  }`
18	`  public void methodA(int sum, int temp){`
19	`    sum = 3 + sum - trace.sum;`
20	`    x = sum + 12 + y;`
21	`    y = trace.temp + temp + sum;`
22	`    this.sum = y + methodB(trace.temp, trace) + trace.temp;`
23	`    System.out.println(sum + " " + y + " " + this.sum);`
24	`  }`
25	`  public int methodB(int x, Trace temp){`
26	`    int sum = x + temp.sum + this.x;`
27	`    temp.temp = sum + this.sum;`
28	`    System.out.println(x + " " + temp.temp + " " + sum);`
29	`    return sum;`
30	`  }`
31	`}`

`Trace p = new Trace(3, 4);`
`Quiz5 q = new Quiz5(p);`
`q.methodA(4, 8);`
`q.methodA(5, 10);`
	Output		

Task 10

1	`public class Test {`
2	`  public static int a=3;`
3	`  public int b=7, c;`
4	`  public Test(){`
5	`    methodA(a+4);`
6	`  }`
7	`  public void methodA(int a){`
8	`    Tracing t = new Tracing(2,7);`
9	`    a = Tracing.a+ Test.a;`
10	`    c = b + a + t.methodB();`
11	`    System.out.println(this.a+" "+this.b+" "+c);`
12	`  }`
13	`}`
14	`class Tracing {`
15	`  public static int a = 9, y = 5;`
16	`  public int x, b;`
17	`  public Tracing(int a, int b){`
18	`    x += a;`
19	`    y += b;`
20	`    this.a = this.x;`
21	`    this.b = this.y;`
22	`  }`
23	`  public int methodB(){`
24	`    System.out.println(this.a+" "+this.b+" "+x);`
25	`    b = y - this.b + Test.a;`
26	`    x += this.b;`
27	`    return this.b;`
28	`  }`
29	`}`

	Output		
`Tracing t2 = new Tracing(4, 3);`			
`Test ex = new Test();`			
`t2.methodB();`			
`ex.methodA(Test.a);`			

Task 11

```
1     public class A {
2         public static int temp = 3;
3         public int sum;
4         public int y;
5         public A(int x) {
6             y = A.temp - 1 + x;
7             sum = this.temp + 2;
8             A.temp -= 2;
9         }
10        public void methodA(int y, int[] n) {
11            int x = 0;
12            n[0] += 1;
13            this.y = this.y + y + temp;
14            A.temp += 1;
15            x = x + 2 + n[0];
16            n[0] = sum + 2;
17            System.out.println(x + " " + this.y + " " + this.sum);
18        }
19    }
20    public class B extends A {
21        public static int x = 1;
22        public B() {
23            super(5);
24            sum = 2;
25            y = A.temp + 1;
26            B.x = 3 + temp + B.x;
27            A.temp -= 2;
28        }
29        public B(B b) {
30            super(2);
31            sum = 3;
32            this.sum = sum + this.sum%2 + 2;
33            B.x = b.x + B.x;
34        }
35        public void methodB(int m, int n) {
36            int[] y = {2, 3};
37            this.y = y[0] + this.y + m;
38            B.x = this.y + 2 + A.temp - n;
39            methodA(B.x, y);
40            this.sum = B.x + y[1] + this.sum;
41            System.out.println(B.x + " " + (y[0]+y[1]) + " " + this.sum);
42        }
43    }
```

Write the output of the following tester code:

int[] n = {23};	Output		
A a1 = new A(3);	x	y	sum
B b1 = new B();			
B b2 = new B(b1);			
a1.methodA(1, n);			
b2.methodB(3, 2);			
a1.methodA(1, n);			