Lab Assignment 10

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Topic:	**Review**
Number of Tasks:	**12**

[NO SUBMISSION]

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

Task 1

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 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();
 	
  }
}
	==========1==========
	Nike Bangladesh Status:
	Branches Opened: 0
	Currently Stocked: Jordan: 0, Cortez: 0, Kobe: 0
	Sold: 0
	==========2==========
	Nike Dhaka Banani outlet:
	Products Currently Stocked: Jordan: 0, Cortez: 0, Kobe: 0
	Sold: 0
	==========3==========
	Nike Chittagong GEC outlet:
	Products Currently Stocked: Jordan: 0, Cortez: 0, Kobe: 0
	Sold: 0
	==========4==========
	==========5==========
	==========6==========
	==========7==========
	Nike Bangladesh Status:
	Branches Opened: 2
	Currently Stocked: Jordan: 2600, Cortez: 450, Kobe: 300
	Sold: 0
	==========8==========
	Nike Dhaka Banani outlet:
	Products Currently Stocked: Jordan: 1400, Cortez: 200, Kobe: 200
	Sold: 0
	==========9==========
	Nike Chittagong GEC outlet:
	Products Currently Stocked: Jordan: 1200, Cortez: 250, Kobe: 100
	Sold: 0
	==========10==========
	Nike Bangladesh Status:
	Branches Opened: 2
	Currently Stocked: Jordan: 1320, Cortez: 360, Kobe: 230
	Sold: 1440
	==========11==========
	Nike Chittagong GEC outlet:
	Products Currently Stocked: Jordan: 680, Cortez: 250, Kobe: 30
	Sold: 590

Task 2

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

```
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

```
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();
  }
}
```

Output

```
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
```

Task 3

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

Parent Class

```
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

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.
	Mobile Phone Detail:
	Model: N3110
	IMEICode: IMEI-102
	SIM Card Status: true
	Balance: 0.0 TK
	1=======================
	Model N1100 is manufactured.
	Mobile Phone Detail:
	Model: N1100
	IMEICode: IMEI-124
	SIM Card Status: false
	Balance: 100.0 TK
	2=======================
	Insufficient balance! Please recharge.
	3=======================
	Recharge successful! Current balance 200.0 TK.
	Recharge successful! Current balance 400.0 TK.
	4=======================
	Dialing the number 88017196xxxx to Bangladesh region.
	5=======================
	No SIM card available! Please check the SIM card connectivity.
	6=======================
	SIM card is activated successfully.
	7=======================
	Dialing the number 45517196xxxx to USA region.
	8=======================
	Dialing is not allowed in this region.

Task 4

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

Parent Class

```
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

```
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();
  }
}
```

Output

```
Name: Drake
Age: 500
Drake roars with a fiery breath!
Drake breathes fire with power level: 75
Drake flies through the sky.
====================
Name: Fawkes
Age: 200
Fawkes sings an enchanting song.
Fawkes is reborn with 5 rebirth cycles.
Fawkes regenerates its body in a burst of
flames.
```

Task 5

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

Parent Class

```
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;
  }
}
```

Driver Code

```
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);
  }
}
```

Output

```
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
10----------------------------
Penny's profile:
User Name: Penny
Email:penny@qmail.com
Messages Sent: 5
```

Task 6

Observe the following Tester class and outputs to design both the **Patient** class and **InPatient** class with appropriate elements.

Tester Class	Expected Output

public class PatientTester {
public static void main(String[] args) {
Patient p1 = new Patient("Robert", "Dr. Thomas");
System.out.println(p1);
System.out.println("1.................");
Patient.details();
System.out.println("2.................");
InPatient p2 = new InPatient("Christina", "Dr. Alex", "Oncology");
System.out.println(p2);
System.out.println("3.................");
InPatient.details();
System.out.println("4.................");
Patient p3 = new InPatient("Sofia", "Dr. Brawn", "Pediatrics");
Patient p4 = new Patient("Patrick", "Dr. Alex");
Patient.details();
System.out.println("5.................");
Patient[] allPatients = { p1, p2, p3, p4 };
Patient.details(allPatients);
}
}
	Patient ID: P01, Name: Robert
	Doctor: Dr. Thomas
	1.................
	Total patients: 1.
	2.................
	New patient admitted in Oncology.
	Patient ID: P02, Name: Christina
	Doctor: Dr. Alex
	Department: Oncology
	3.................
	Total patients: 2.
	Admitted In-Patients: 1.
	Out-Patients: 1.
	4.................
	New patient admitted in Pediatrics.
	Total patients: 4.
	5.................
	Details of 4 selected patients:
	== == == == ==
	Patient ID: P01, Name: Robert
	Doctor: Dr. Thomas
	== == == == ==
	Patient ID: P02, Name: Christina
	Doctor: Dr. Alex
	Department: Oncology
	== == == == ==
	Patient ID: P03, Name: Sofia
	Doctor: Dr. Brawn
	Department: Pediatrics
	== == == == ==
	Patient ID: P04, Name: Patrick
	Doctor: Dr. Alex

Task 7

```
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 8

```
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 }
```

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

Output		

Task 9

```
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};
A a1 = new A(3);
B b1 = new B();
B b2 = new B(b1);
a1.methodA(1, n);
b2.methodB(3, 2);
a1.methodA(1, n);
	Output		

	x	y	sum

Task 10

```
1  public class Bank {
2   public static int rs = 3;
3   public static int cd = -8;
4   public int bl = 0;
5   public int br = 0;
6   public Bank() {
7     br = rs - 2;
8     bl = rs + 1;
9     rs -= 2;
10  }
11  public void deposit(int a, int b) {
12    int cd = 0;
13    br = br + a + (rs++);
14    cd = cd + 1 + b;
15    bl = bl + cd + br;
16    System.out.println(cd + " " + br + " " + bl);
17  }
18 }
19 public class Account extends Bank {
20  public static int cd = 1;
21  public int bl = -4;
22  public Account() {
23    bl = 0;
24    br = rs + 3;
25    super.bl = 2 + rs + 3;
26    rs -= 2;
27  }
28  public Account(Account acc) {
29    bl = acc.bl + super.bl;
30    cd = acc.cd;
31    acc.withdraw(2, 3);
32  }
```

```
33  public void withdraw(int a, int b) {
34    int br = 0;
35    br = br + this.br;
36    cd = br + 2 + (++rs);
37    deposit(cd, br);
38    bl = cd + br + bl;
39    System.out.println(cd + " " + br + " " + bl);
40  }
41 }
```

Write the output of the following code:

```
public class Tester {
   public static void main(String[] args) {
       Bank b1 = new Bank();
       Account a1 = new Account();
       Account a2 = new Account(a1);
       a1.deposit(3, 4);
       a2.withdraw(1, 6);
   }
}
```

Output:-		

Task 11

```
1  public class Device {
2      public void start() {
3        System.out.println("Device starting");
4      }
5      public void shutdown() {
6        System.out.println("Device shutting down");
7        start();
8      }
9  }
10 public class Laptop extends Device {
11     public void start() {
12       System.out.println("Laptop booting up");
13     }
14 }
15 public class Smartphone extends Device {
16     public void notifyUser() {
17       System.out.println("Smartphone notification");
18     }
19 }
20 public class GamingLaptop extends Laptop {
21     public void start() {
```

```
22          System.out.println("GamingLaptop powering on");
23          super.start();
24      }
26      public void notifyUser() {
27          System.out.println("GamingLaptop notification alert");
28      }
29  }
```

Assuming the following variables have been defined:

```
Device d1 = new GamingLaptop();
Device d2 = new Laptop();
Device d3 = new Device();
Object d4 = new Laptop();
Laptop d5 = new GamingLaptop();
Object d6 = new Smartphone();
```

In the table below,

- The output produced by the statement in the left-hand column, should be written in the right-hand column
- If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c" to indicate three lines of output with "a" followed by "b" followed by "c".
- If the statement causes an error, fill in the right-hand column with either the phrase "compiler error" or "runtime error" to indicate when the error would be Detected.

	Statement	**Output**
1	d1.start();	
2	d2.start();	
3	d4.start();	
4	d6.start();	
5	d1.shutdown();	
6	d3.shutdown();	
7	d4.shutdown();	
8	d5.shutdown();	
9	d6.shutdown();	
10	((GamingLaptop)d4).notifyUser();	
11	((GamingLaptop)d6).shutdown();	
12	((Smartphone)d1).notifyUser();	
13	((Smartphone) d6).start();	
14	((GamingLaptop) d5).start();	
15	((GamingLaptop)d5).notifyUser();	

Task 12

```
1  public class Department{
2    public String chant = "I love my Department!";
3    public void task1(){
4      System.out.println(chant);
5    }
6    public void task2(){
7      task1();
8      System.out.println("Doing Task 2 "+ chant);
9    }
10   public void advising(){
11     System.out.println("Advising is Pain.");
12   }
13   public String toString(){
14     advising();
15     return chant;
16   }
17 }
18 public class CSEDept extends Department{
19   public String chant = "CSE is Love.";
20   public void task2(){
21     System.out.println("Doing Task 2 "+ chant);
22   }
23   public void advising(){
24     System.out.println("Advising is Pain.");
25   }
26 }
27 public class EEEDept extends Department{
28   public String chant = "Help.";
29   public void task1(){
30     System.out.println("Doing Task 1 "+ chant);
31   }
32   public void advising(){
33     super.advising();
34     System.out.println(chant);
35   }
36   public String toString(){
37     task2();
38     return chant;
39   }
40 }
41 public class SoftwareDept extends CSEDept{
42   public String chant = "Software is fun!";
43   public String toString(){
```

```
44      advising();
45      return chant;
46    }
47    public void task1(){
48      System.out.println("Doing Task 1 "+ chant);
49      task2();
50    }
51  }
53  public class RoboticsDept extends EEEDept{
54    public String chant = "New Department woohoo!";
55    public void task2(){
56      System.out.println("Doing Task 2 "+ chant);
57    }
58    public void advising(){
59      super.advising();
60      task1();
61      System.out.println(chant);
62    }
63  }
```

Assuming the following variables have been defined:

```
Department b1 = new Department();
Department b2 = new EEEDept();
Department b3 = new SoftwareDept();
CSEDept c1 = new SoftwareDept ();
CSEDept c2 = new CSEDept();
Object o1 = new EEEDept();
Department o2 = new RoboticsDept();
```

In the table below,

- The output produced by the statement in the left-hand column, should be written in the right-hand column
- If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c" to indicate three lines of output with "a" followed by "b" followed by "c".
- If the statement causes an error, fill in the right-hand column with either the phrase "compiler error" or "runtime error" to indicate when the error would be detected.

	Statement	**Output**
1	`c2.task2();`	
2	`b3.task1();`	
3	`System.out.println(b3);`	
4	`((CSEDept)b1).task1();`	
5	`System.out.println(((Object)b2).chant);`	

6	((Department)b2).advising();	
7	((CSEDept)b2).task2();	
8	((EEEDept)b2).task2();	
9	System.out.println((CSEDept)b3);	
10	System.out.println(((Department)c1).chant);	
11	((Object)c1).toString();	
12	System.out.println(((Department)c2));	
13	System.out.println(((Department)o1));	
14	((EEEDept)o2).advising();	
15	((SoftwareDept)o2).task1();	