

# Lab Assignment 07



Inspiring Excellence

Course Code:	CSE111
Course Title:	Programming Language II
Topic:	Inheritance and Overriding
Number of Tasks:	11

*[Submit all the Coding Tasks (Task 1 to 8) in the Google Form shared on buX before the next lab. Submit the Tracing Tasks (Task 9 to 11) handwritten to your Lab Instructors at the beginning of the lab]*

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

### Task 1

Given the following classes, write the code for the **BBAStudent** class so that the following output is printed when we run the TestStudent class.

Driver Code and Parent Class	Output
<pre>public class Student{     private String name = "Just a Student";     private String department = "nothing";      public void setDepartment(String dpt){         this.department = dpt;     }     public void setName(String name){         this.name = name;     }     public void details(){         System.out.println("Name : " + name + " Department: " + department);     } }  //Tester Class public class TestStudent{     public static void main(String [] args){         BBAStudent b1 = new BBAStudent();         BBAStudent b2 = new BBAStudent("Humty Dumty");         BBAStudent b3 = new BBAStudent("Little Bo Peep");         b1.details();         System.out.println("1-----");         b2.details();         System.out.println("2-----");         b3.details();     } }</pre>	<pre>Name: Default Department: BBA 1----- Name: Humty Dumty Department: BBA 2----- Name: Little Bo Peep Department: BBA</pre>

## Task 2

Design the **CheckingAccount** class derived from the **Account** class with appropriate attributes and properties so that the driver code can generate the output given below.

Driver Code and Parent Class	Output
<pre>public class Account{     public double balance = 0.0;      public Account(double balance){         this.balance = balance;     }     public double showBalance(){         return balance;     } }  //Tester Class public class TestAccount{     public static void main(String [] args){         System.out.println("Total Checking Accounts: "+CheckingAccount.count);         CheckingAccount c1 = new CheckingAccount();         System.out.println("Account Balance: " + c1.showBalance());         CheckingAccount c2 = new CheckingAccount(100.0);         System.out.println("Account Balance: " + c2.showBalance());         CheckingAccount c3 = new CheckingAccount(200.0);         System.out.println("Account Balance: " + c3.showBalance());         System.out.println("Total Checking Accounts: "+CheckingAccount.count);     } }</pre>	<pre>Total Checking Accounts: 0 Account Balance: 0.0 Account Balance: 100.0 Account Balance: 200.0 Total Checking Accounts: 3</pre>

### Task 3

Design the **Dog** and **Cat** class derived from the **Animal** class with appropriate attributes and properties so that the driver code can generate the output given below.

Driver Code and Parent Class	Output
<pre>public class Animal {     public String name;     public int age;     public String color;     public Animal(String name, int age, String color) {         this.name = name;         this.age = age;         this.color = color;     }     public void makeSound() {         System.out.println("Animal makes a sound");     }     public String info() {         return "Name: "+name+"\nAge: "+age+"\nColor: "+color+"\n";     } }  public class AnimalTester {     public static void main(String[] args) {         Dog dog = new Dog("Buddy", 5, "Brown", "Bulldog");         Cat cat = new Cat("Kitty", 3, "White", "Persian");         System.out.println("1.=====");         System.out.println(dog.info());         System.out.println("2.=====");         System.out.println(cat.info());         System.out.println("3.=====");         dog.makeSound();         System.out.println("4.=====");         cat.makeSound();     } }</pre>	<pre>1.===== Name: Buddy Age: 5 Color: Brown Breed: Bulldog 2.===== Name: Kitty Age: 3 Color: White Breed: Persian 3.===== Brown color Buddy is barking 4.===== White color Kitty is meowing</pre>

## Task 4

Given the following classes, write the code for the **Vehicle2010** class to print the following output when we run the Vehicle2010User class.

Driver Code and Parent Class	Output
<pre>public class Vehicle{     public int x;     public int y;      public void moveUp(){         y = y+1;     }     public void moveDown(){         y = y-1;     }     public void moveLeft(){         x = x-1;     }     public void moveRight(){         x = x+1;     }     public String toString(){         return "("+ x + ", "+ y + ")";     } }  //Tester Class public class Vehicle2010User{     public static void main(String[] args){         Vehicle2010 car1 = new Vehicle2010();         System.out.println(car1);         car1.moveLowerLeft();         System.out.println(car1);          Vehicle2010 car2 = new Vehicle2010();         System.out.println(car2);         car2.moveUpperRight();         System.out.println(car2);         car2.moveLowerRight();         System.out.println(car2);     } }</pre>	<pre>(0,0) (-1,-1) (0,0) (1,1) (2,0)</pre>

## Task 5

Design the **ComplexNumber** class with the necessary property to produce the output from the given driver code.

Driver Code and Parent Class	Output
<pre>public class RealNumber {     public double realValue;     public RealNumber() {         this(0.0);     }     public RealNumber(double realValue) {         this.realValue = realValue;     }     public String toString(){         return "RealPart: " + realValue;     } }  public class ComplexNumberTester {     public static void main(String[] args) {         ComplexNumber cn1 = new ComplexNumber();         System.out.println(cn1);         System.out.println("-----");         ComplexNumber cn2 = new ComplexNumber(5.0, 7.0);         System.out.println(cn2);     } }</pre>	<pre>RealPart: 1.0 ImaginaryPart: 1.0 ----- RealPart: 5.0 ImaginaryPart: 7.0</pre>

## Task 6

Design the **Manager** and **Developer** class derived from the **Employee** class with appropriate attributes and properties so that the driver code can generate the output given below. [Hint:

Manager:

1. Adds a bonus to the base salary if the manager works more than 40 hours.
2. If the manager works more than 100 hours, the full amount is approved; if they work more than 80 hours, half the amount is approved. Otherwise, the increment is denied.

Developer:

1. Adds \$700 to the base salary if the developer works with Java programming language.]

Driver Code and Parent Class	Output
<pre> public class Employee {     public String name;     private double baseSalary;     private int hoursWorked;      public Employee(String name, double baseSalary, int hoursWorked){         this.name = name;         this.baseSalary = baseSalary;         this.hoursWorked = hoursWorked;     }     public double getBaseSalary() {         return baseSalary;     }     public void setBaseSalary(double baseSalary) {         this.baseSalary = baseSalary;     }     public int getHoursWorked() {         return hoursWorked;     }     public void setHoursWorked(int hoursWorked) {         this.hoursWorked = hoursWorked;     }     public void displayInfo() {         System.out.println("Name: " + name);         System.out.println("Base Salary: \$" + baseSalary);         System.out.println("Work Hours: " + hoursWorked);     } }  public class EmployeeTester {     public static void main(String[] args) {         Manager neymar = new Manager("Neymar",1000, 45, 10);         Developer messi = new Developer("Messi",1000,50,"Java");         Developer chiesa = new Developer("Chiesa", 1000, 50, "Javascript");         neymar.calculateSalary();         System.out.println("1.=====");         neymar.displayInfo();         System.out.println("2.=====");         neymar.requestIncrement(100);         System.out.println("3.=====");         neymar.setHoursWorked(85);         neymar.requestIncrement(100);         System.out.println("4.=====");         neymar.calculateSalary();         System.out.println("5.=====");         neymar.displayInfo();         System.out.println("6.=====");         messi.calculateSalary();         System.out.println("7.=====");         messi.displayInfo();     } } </pre>	<pre> 1.===== Name: Neymar Base Salary: \$1000.0 Work Hours: 45 Bonus: 10.0 % Final Salary: \$1100.0 2.===== Increment denied. 3.===== \$50 Increment approved. 4.===== 5.===== Name: Neymar Base Salary: \$1050.0 Work Hours: 85 Bonus: 10.0 % Final Salary: \$1155.0 6.===== 7.===== Name: Messi Base Salary: \$1000.0 Work Hours: 50 Language: Java Final Salary: \$1700.0 8.===== Name: Chiesa Base Salary: \$1000.0 Work Hours: 50 Language: Javascript Final Salary: \$1000.0 </pre>

<pre> System.out.println("8.====="); chiesa.calculateSalary(); System.out.println("9.====="); chiesa.displayInfo(); } } </pre>	
--	--

## Task 7

Design the **CinemexTicket** class derived from the **MovieTicket** Class so that the given output is produced:

- ❖ The **seatTypes** and **seatPrices** arrays contain the type of the seat and its corresponding price
- ❖ Night show charge (15% of ticket price) will be applicable if the time is between 6:00 PM - 11:00 PM
- ❖ Unique id for a ticket is generated by:  
MovieName-FirstLetterOfSeatType-TicketCount
- ❖ You may need to use **.split()** and **Integer.parseInt()** built-in methods

Driver Code and Parent Class	Output
<pre> public class MovieTicket {     public static String [] seatTypes = {"Regular", "Premium", "IMAX 3D"};     public static double [] seatPrices = {300.0, 450.0, 600.0};     public static int nightShowCharge = 15;     private String movie;     public String showtime;     public String date;     private double price;     public String seat;      public MovieTicket(String movie, String date, String showtime, double price) {         this.movie = movie;         this.showtime = showtime;         this.date = date;         this.price = price;         this.seat = "Not Selected";     }     public void setPrice(double price) {         this.price = price;     }     public double getPrice() {         return price;     }     public String getMovie() {         return movie;     } </pre>	<pre> Total movie ticket(s): 1 1===== Ticket price is calculated successfully. 2===== Ticket ID: Deadpool and Wolverine-R-1 Movie: Deadpool and Wolverine Showtime: 18:30 Date: July 24, 2024 Genre: Action-Comedy Seat Type: Regular Price(tk): 345.0 Status: Not Paid 3===== Payment Successful. 4===== Ticket ID: Deadpool and Wolverine-R-1 Movie: Deadpool and Wolverine Showtime: 18:30 Date: July 24, 2024 Genre: Action-Comedy Seat Type: Regular Price(tk): 345.0 Status: Paid 5===== </pre>



```

    }

    public String toString() {
        return "Movie: " + movie + "\nShowtime: " + showtime +
"\nDate: " + date;
    }
}

//Driver Code
public class Tester {
    public static void main(String[] args) {
        CinemexTicket ticket1 = new CinemexTicket("Deadpool and
Wolverine", "18:30", "Action-Comedy", "July 24, 2024");
        System.out.println("Total movie ticket(s): " +
CinemexTicket.getTotalTickets());
        System.out.println("1=====");
        ticket1.calculateTicketPrice();
        System.out.println("2=====");
        System.out.println(ticket1);
        System.out.println("3=====");
        System.out.println(ticket1.confirmPayment());
        System.out.println("4=====");
        System.out.println(ticket1);
        System.out.println("5=====");
        CinemexTicket ticket2 = new CinemexTicket("Twisters", "10:00",
"Sci-Fi", "August 10, 2024", "Premium");
        System.out.println("Total movie ticket(s): " +
CinemexTicket.getTotalTickets());
        System.out.println("6=====");
        ticket2.calculateTicketPrice();
        System.out.println("7=====");
        System.out.println(ticket2.confirmPayment());
        System.out.println("8=====");
        System.out.println(ticket2);
        System.out.println("9=====");
        System.out.println(ticket2.confirmPayment());
    }
}

```

```

Total movie ticket(s): 2
6=====
Ticket price is calculated
successfully.
7=====
Payment Successful.
8=====
Ticket ID: Twisters-P-2
Movie: Twisters
Showtime: 10:00
Date: August 10, 2024
Genre: Sci-Fi
Seat Type: Premium
Price(tk): 450.0
Status: Paid
9=====
Ticket price is already paid!

```

## Task 8

Design the **KKTea** (parent) and **KKFlavouredTea** (child) classes so that the following output is produced. The **KKFlavouredTea** class should inherit **KKTea** and **KKTea** should inherit the **Tea** class. Note that:

- An object of either class represents a single box of teabags.
- Each tea bag weighs 2 grams.
- The status of an object refers to whether it is sold or not

Driver Code and Parent Class	Output
<pre> public class Tea {     public String name;     public int price;     public boolean status;      public Tea(String name, int price) {         this.name = name;         this.price = price;         this.status = false;     }      public void productDetail() {         System.out.println("Name: " + name + ", Price: " + price);         System.out.println("Status: " + status);     } } //Driver Code public class TeaTester{     public static void main(String[] args) {         KKTea t1 = new KKTea(250, 50);         System.out.println("-----1-----");         t1.productDetail();         System.out.println("-----2-----");         KKTea.totalSales();         System.out.println("-----3-----");         KKTea t2 = new KKTea(470, 100);         KKTea t3 = new KKTea(360, 75);         KKTea.updateSoldStatusRegular(t1);         KKTea.updateSoldStatusRegular(t2);         System.out.println("-----4-----");         t2.productDetail();         System.out.println("-----5-----");         KKTea.totalSales();         System.out.println("-----6-----");         KKFlavouredTea t4 = new KKFlavouredTea("Jasmine", 260, 50);         KKFlavouredTea t5 = new KKFlavouredTea("Honey Lemon", 270, 45);         KKFlavouredTea t6 = new KKFlavouredTea("Honey Lemon", 270, 45);         System.out.println("-----7-----"); </pre>	<pre> -----1----- Name: KK Regular Tea, Price: 250 Status: false Weight: 100, Tea Bags: 50 -----2----- Total Sales: 0 KK Regular Tea: 0 -----3----- -----4----- Name: KK Regular Tea, Price: 470 Status: true Weight: 200, Tea Bags: 100 -----5----- Total Sales: 2 KK Regular Tea: 2 -----6----- -----7----- Name: KK Jasmine Tea, Price: 260 Status: false Weight: 100, Tea Bags: 50 -----8----- Name: KK Honey Lemon Tea, Price: 270 Status: false Weight: 90, Tea Bags: 45 -----9----- -----10----- Total Sales: 5 KK Regular Tea: 2 KK Flavoured Tea: 3 </pre>

<pre>t4.productDetail(); System.out.println("-----8-----"); t6.productDetail(); System.out.println("-----9-----"); KKFlavouredTea.updateSoldStatusFlavoured(t4); KKFlavouredTea.updateSoldStatusFlavoured(t5); KKFlavouredTea.updateSoldStatusFlavoured(t6); System.out.println("-----10-----"); KKTea.totalSales(); } }</pre>	
--	--

## Task 9

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

A a1 = new A();	x	y	sum
B b1 = new B();			
B b2 = new B(b1);			
a1.methodA(1, 1);			
b1.methodA(1, 2);			
b2.methodB(3, 2);			

## Task 10

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

A a1 = new A();	x	y	sum
B b1 = new B();			
B b2 = new B(b1);			
a1.methodA(1, 1);			
b1.methodA(1, 2);			
b2.methodB(3, 2);			

## Task 11

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

<code>int x[] = {23};</code>			
<code>A a1 = new A();</code>			
<code>B b1 = new B();</code>			
<code>B b2 = new B(b1);</code>			
<code>a1.methodA(1, x);</code>			
<code>b2.methodB(3, 2);</code>			
<code>a1.methodA(1, x);</code>			

## Ungraded Tasks (Optional)

(You don't have to submit the ungraded tasks)

### Task 1

Design the class **Dog** so that the desired outputs are generated properly.

Driver Code and Parent Class	Expected Output
<pre>public class AnimalTester{     public static void main(String args[]){         Animal a1 = new Animal();         System.out.println("1-----");         a1.details();         System.out.println("2-----");         Dog d1 = new Dog();         d1.name = "Pammy";         System.out.println("3-----");         System.out.println("Name: " + d1.getName());         d1.details();         System.out.println("4-----");         d1.updateSound("Bark");         System.out.println("5-----");         d1.details();     } }  public class Animal{     public int legs = 4;     public String sound = "Not defined";      public void details(){         System.out.println("Legs: "+legs);         System.out.println("Sound: "+sound);     } }</pre>	<pre>1----- Legs: 4 Sound: Not defined 2----- The dog says hello! 3----- Name: Pammy Legs: 4 Sound: Not defined 4----- 5----- Legs: 4 Sound: Bark</pre>

### Task 2

Design the **ScienceExam** class with the necessary property to produce the output from the given driver code.

Driver Code	Output
<pre>public class Exam {     public int marks;     public int time;      public Exam(int marks) {</pre>	<pre>Marks: 100 Time: 90 minutes Number of Parts: 4 ----- Maths, English, Physics, HigherMaths Part 1 - Maths Part 2 - English</pre>



```

        this.marks = marks;
        this.time = 60;
    }
    public String examSyllabus() {
        return "Maths, English";
    }
    public String examParts() {
        return "Part 1 - Maths\nPart 2 -
English\n";
    }
}

//Tester Class
public class ExamTester {
    public static void main(String[] args) {
        ScienceExam ex1 = new ScienceExam(100, 90,
"Physics", "HigherMaths");
        System.out.println(ex1);
        System.out.println("-----");
        System.out.println(ex1.examSyllabus());
        System.out.println(ex1.examParts());
        System.out.println("=====");
        ScienceExam ex2 = new ScienceExam(100, 120,
"Physics", "HigherMaths", "Drawing");
        System.out.println(ex2);
        System.out.println("-----");
        System.out.println(ex2.examSyllabus());
        System.out.println(ex2.examParts());
    }
}

```

```

Part 3 - Physics
Part 4 - HigherMaths

=====
Marks: 100 Time: 120 minutes Number of Parts: 5
-----
Maths, English, Physics, HigherMaths, Drawing
Part 1 - Maths
Part 2 - English
Part 3 - Physics
Part 4 - HigherMaths
Part 5 - Drawing

```

### Task 3

1	public class A {
2	public static int temp = 4;
3	public static int x = -10;
4	public int sum = 0;
5	public int y = 0;
7	public A() {
8	y = temp - 2;
9	sum = temp + 1;
10	temp -= 2;
11	}
13	public void methodA(int m, int n) {
14	int x = 0;
15	y = y + m + (temp++);
16	x = x + 1 + n;
17	sum = sum + x + y;
18	System.out.println(x + " " + y + " " + sum);
19	}
20	}
22	public class B extends A {
23	public static int x = 0;
24	public int sum = -6;
25	public B() {
26	super();
27	sum = 0;
28	y = temp + 3;
29	super.sum = 3 + temp + 2;
30	temp -= 2;
31	}
33	public B(B b) {
34	super();
35	if (b == null) {
36	y = temp + 3;
37	sum = 3 + temp + 2;
38	temp -= 2;
39	} else {
40	sum = b.sum + super.sum;
41	x = b.x;
42	b.methodB(2, 3);
43	}
44	}
46	public void methodB(int m, int n) {
45	int y = 0;

