Lab Assignment 03

Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Topic:	**Instance Method and Method Overloading**
Number of Tasks:	**11 (Coding: 08, Tracing: 03)**

[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

Design the **BankAccount** class in such a way so that the following code provides the expected output.

Driver Class	Output

public class BankAccountTester{
  public static void main(String args[]){
  BankAccount acc1 = new BankAccount();
  System.out.println(acc1.printDetails());
  System.out.println("-----1-----");
  acc1.setInfo(1456890,"Salary");
  System.out.println("-----2-----");
  System.out.println(acc1.printDetails());
  System.out.println("-----3-----");
  BankAccount acc2 = new BankAccount();
  acc2.setInfo(1765498,"Student");
  System.out.println("-----4-----");
  System.out.println(acc2.printDetails());
  }
}
	Account No: 0
	Type: Not Set
	-----1-----
	Account information updated!
	-----2-----
	Account No: 1456890
	Type: Salary
	-----3-----
	Account information updated!
	-----4-----
	Account No: 1765498
	Type: Student

Task 2

Design the **Shape** class with necessary properties to produce the given output for the provided driver code.

Driver Class	Output

public class ShapeTester{
  public static void main(String args []){
  Shape circle = new Shape();
  Shape triangle = new Shape();
  Shape rectangle = new Shape();
 	
  circle.setParameters("Circle", 5);
  triangle.setParameters("Triangle", 4, 7);
  rectangle.setParameters("Rectangle", 2.4, 4.4);
 	
  System.out.println(circle.details());
	Shape Name: Circle
	Area: 78.54
	1---------------
	Shape Name: Triangle
	Area: 14.0
	2---------------
	Shape Name: Rectangle
	Area: 10.56

```
    System.out.println("1---------------");
    System.out.println(triangle.details());
    System.out.println("2---------------");
    System.out.println(rectangle.details());
  }
}
```

Task 3

Design the "**Shelf**" class with necessary properties, so that the given output is produced for the provided driver code.

Driver Class	Output

`public class ShelfTester{`
`  public static void main(String [] args){`
`  Shelf shelf = new Shelf();`
`  shelf.showDetails();`
`  System.out.println("1---------------");`
`  shelf.addBooks(3);`
`  System.out.println("2---------------");`
`  shelf.capacity = 7;`
`  shelf.addBooks(3);`
`  System.out.println("3---------------");`
`  shelf.showDetails();`
`  System.out.println("4---------------");`
`  shelf.addBooks(5);`
`  shelf.showDetails();`
`  shelf.capacity += 4;`
`  System.out.println("6---------------");`
`  shelf.addBooks(5);`
`  shelf.showDetails();`
`  }`
`}`
	`Shelf capacity: 0`
	`Number of books: 0`
	`1---------------`
	`Zero capacity. Cannot add books.`
	`2---------------`
	`3 books added to shelf`
	`3---------------`
	`Shelf capacity: 7`
	`Number of books: 3`
	`4---------------`
	`Exceeds capacity`
	`Shelf capacity: 7`
	`Number of books: 3`
	`6---------------`
	`5 books added to shelf`
	`Shelf capacity: 11`
	`Number of books: 8`

Task 4

Design the **Student** class with the necessary properties to produce the given output for the provided driver code.

Hint:

- A student having **cgpa>=3.5** and **credit>10** is **eligible for scholarship.**
- A student with **cgpa>=3.5 but <3.7** is eligible for **Need-based scholarship.**
- A student having **cgpa >=3.7** is eligible for **Merit based scholarship**

Driver Code	Output

public class StudentTester{
  public static void main(String[] args) {
  Student std1 = new Student();
  std1.showDetails();
  System.out.println("1---------------");
  std1.updateDetails("Alif", 3.99, 12);
  System.out.println("2---------------");
  std1.checkScholarshipEligibility();
  System.out.println("3---------------");
  std1.showDetails();
  Student std2 = new Student();
  std2.updateDetails("Mim", 3.4);
  Student std3 = new Student();
  std3.updateDetails("Henry", 3.5, 15, "BBA");
  System.out.println("5---------------");
  std2.checkScholarshipEligibility();
  System.out.println("6---------------");
  std3.checkScholarshipEligibility();
  System.out.println("7---------------");
  std2.showDetails();
  System.out.println("8---------------");
  std3.showDetails();
  }
}
	Name: Not Set
	Department: CSE
	CGPA: 0.0
	Credits: 9
	Scholarship Status: Not Set
	1---------------
	2---------------
	Alif is eligible for Merit based scholarship
	3---------------
	Name: Alif
	Department: CSE
	CGPA: 3.99
	Credits: 12
	Scholarship Status: Merit based scholarship
	5---------------
	Mim is not eligible for scholarship
	6---------------
	Henry is eligible for Need based scholarship
	7---------------
	Name: Mim
	Department: CSE
	CGPA: 3.4
	Credits: 9
	Scholarship Status: No scholarship
	8---------------
	Name: Henry
	Department: BBA
	CGPA: 3.5
	Credits: 15
	Scholarship Status: Need based scholarship

Task 5

Design the **Library** class with the necessary properties so that the given output is produced for the provided driver code.

Driver Code	Output

public class Tester5{
  public static void main(String[] args) {
  Library a1 = new Library();
  a1.setBookCapacity(3);
  System.out.println("1-------------");
  a1.addBook("Ice");
  System.out.println("2-------------");
  a1.printDetail();
  System.out.println("3-------------");
  a1.addBook("Emma");
  a1.addBook("Wings");
  a1.addBook("Next");
  System.out.println("4-------------");
  a1.printDetail();
  Library a2 = new Library();
  a2.setBookCapacity(4);
  System.out.println("5-------------");
  a2.addBook("Onnobhubon");
  a2.addBook("Ami");
  System.out.println("6-------------");
  a2.printDetail();
  System.out.println("7-------------");
  a2.addBook("Deyal");
  a2.addBook("Himu");
  a2.addBook("Megher Upor Bari");
  System.out.println("8-------------");
  a2.printDetail();
  }
}
	1-------------
	Book 'Ice' added to the library
	2-------------
	Maximum Capacity: 3
	Total Books: 1
	Book list:
	Ice
	3-------------
	Book 'Emma' added to the library
	Book 'Wings' added to the library
	Exceeds maximum capacity. You can't add more than 3 books
	4-------------
	Maximum Capacity: 3
	Total Books: 3
	Book list:
	Ice
	Emma
	Wings
	5-------------
	Book 'Onnobhubon' added to the library
	Book 'Ami' added to the library
	6-------------
	Maximum Capacity: 4
	Total Books: 2
	Book list:
	Onnobhubon
	Ami
	7-------------
	Book 'Deyal' added to the library
	Book 'Himu' added to the library
	Exceeds maximum capacity. You can't add more than 4 books
	8-------------
	Maximum Capacity: 4
	Total Books: 4
	Book list:
	Onnobhubon
	Ami

	Deyal Himu

Task 6

Design the **TaxiLagbe** class with necessary properties to produce the given output for the provided driver code.

Driver Code	Output

public class TaxiTester{
 public static void main(String[] args) {
  TaxiLagbe taxi1 = new TaxiLagbe();
  taxi1.storeInfo("1010-01", "Dhaka");
  System.out.println("1-------------");
  taxi1.printDetails();
  System.out.println("2-------------");
  taxi1.addPassenger("Wilson", 105);
  System.out.println("3-------------");
  taxi1.printDetails();
  System.out.println("4-------------");
  taxi1.addPassenger("Walker", 100, "Wood", 200);
  System.out.println("5-------------");
  taxi1.printDetails();
  System.out.println("6-------------");
  taxi1.addPassenger("Karen", 200);
  taxi1.addPassenger("Donald", 130);
  System.out.println("7-------------");
  taxi1.printDetails();
  System.out.println("8-------------");
  TaxiLagbe taxi2 = new TaxiLagbe();
  taxi2.storeInfo("1010-02", "Khulna");
  taxi2.addPassenger("Don", 115, "Parker", 215);
  System.out.println("9-------------");
  taxi2.printDetails();
 }
	1-------------
	Taxi number: 1010-01
	This taxi can cover Dhaka area
	Total Passenger: 0
	Passenger Lists:
	Total collected fare: 0 Taka
	2-------------
	Dear Wilson! Welcome to TaxiLagbe
	3-------------
	Taxi number: 1010-01
	This taxi can cover Dhaka area
	Total Passenger: 1
	Passenger Lists:
	Wilson
	Total collected fare: 105 Taka
	4-------------
	Dear Walker! Welcome to TaxiLagbe
	Dear Wood! Welcome to TaxiLagbe
	5-------------
	Taxi number: 1010-01
	This taxi can cover Dhaka area
	Total Passenger: 3
	Passenger Lists:
	Wilson Walker Wood
	Total collected fare: 405 Taka
	6-------------
	Dear Karen! Welcome to TaxiLagbe
	Taxi Full! No more passengers can be added
	7-------------

} 	Taxi number: 1010-01 This taxi can cover Dhaka area Total Passenger: 4 Passenger Lists: Wilson Walker Wood Karen Total collected fare: 605 Taka 8------------ Dear Don! Welcome to TaxiLagbe Dear Parker! Welcome to TaxiLagbe 9------------ Taxi number: 1010-02 This taxi can cover Khulna area Total Passenger: 2 Passenger Lists: Don Parker Total collected fare: 330 Taka

Task 7

Complete the following **Cart** class to generate the given output from the tester code:

- A cart will have a cart number which will be assigned in ***create_cart()*** method.
- Each cart can hold up to 3 items (at max).
- Each cart must have two arrays to store items and their respective prices.
- The items inside a cart will be added in ***addItem()*** method only if the cart items do not exceed 3.
- The ***giveDiscount()*** method saves the discount given to that cart object and updates the price accordingly.

Driver Code	Output

public class CartTester{
  public static void main(String [] args){
  Cart c1 = new Cart ();
  Cart c2 = new Cart ();
  Cart c3 = new Cart ();
 	
  c1.create_cart(1);
  c2.create_cart(2);
  c3.create_cart(3);
  System.out.println("====1====");
  c1.addItem("Table", 3900.5);
  c1.addItem("Chair", 1400.76);
  c1.addItem(5400.87, "Television");
  c1.addItem(5000.0, "Refrigerator");
 	
  System.out.println("====2====");
  c2.addItem("Stove",439.90);
 	
  System.out.println("====3====");
  c3.addItem("Chair",1400.5);
  c3.addItem(3400.0, "Chair");
 	
  System.out.println("====4====");
  c1.cartDetails();
 	
  System.out.println("====5====");
  c2.cartDetails();
 	
  System.out.println("====6====");
  c3.cartDetails();
  c1.giveDiscount(10);
 	
  System.out.println("====7====");
  c1.cartDetails();
  }
}
	====1====
	Table added to cart 1.
	You have 1 item(s) in your cart now.
	Chair added to cart 1.
	You have 2 item(s) in your cart now.
	Television added to cart 1.
	You have 3 item(s) in your cart now.
	You already have 3 items on your cart
	====2====
	Stove added to cart 2.
	You have 1 item(s) in your cart now.
	====3====
	Chair added to cart 3.
	You have 1 item(s) in your cart now.
	Chair added to cart 3.
	You have 2 item(s) in your cart now.
	====4====
	Your cart(c1) :
	Table - 3900.5
	Chair - 1400.76
	Television - 5400.87
	Discount Applied: 0.0%
	Total price: 10702.130000000001
	====5====
	Your cart(c2) :
	Stove - 439.9
	Discount Applied: 0.0%
	Total price: 439.9
	====6====
	Your cart(c3) :
	Chair - 1400.5
	Chair - 3400.0
	Discount Applied: 0.0%
	Total price: 4800.5
	====7====
	Your cart(c1) :
	Table - 3900.5
	Chair - 1400.76
	Television - 5400.87
	Discount Applied: 10.0%
	Total price: 9631.917000000001

Task 8

Design the **Reader** class in such a way so that the following code provides the expected output.

- A reader will have a name, capacity to read and an array of books they are reading.
- The initial capacity of a reader will be 0. The initial name will be "New user".

Driver Code	Expected Output

`public class Reader_tester {`
`  public static void main(String[] args){`
`  Reader r1 = new Reader();`
`  Reader r2 = new Reader();`
 	
`  System.out.println("1 ==========");`
`  System.out.println(r1.createReader("Messi", 2));`
`  System.out.println(r2.createReader("Ronaldo", 3));`
 	
`  System.out.println("2 ==========");`
`  r1.readerInfo();`
 	
`  System.out.println("3 ==========");`
`  r2.addBook("Java");`
`  r2.addBook("Python");`
`  r2.addBook("C++");`
`  r2.readerInfo();`
 	
`  System.out.println("4 ==========");`
`  r1.addBook("C#");`
`  r1.addBook("Rust");`
`  r1.addBook("GoLang");`
 	
`  System.out.println("5 ==========");`
`  r2.addBook("Python");`
 	
`  System.out.println("6 ==========");`
`  r1.readerInfo();`
`  }`
`}`
	`1 ==========`
	`A new reader is created!`
	`A new reader is created!`
	`2 ==========`
	`Name: Messi`
	`Capacity: 2`
	`Books:`
	`No books added yet`
	`3 ==========`
	`Name: Ronaldo`
	`Capacity: 3`
	`Books:`
	`Book 1: Java`
	`Book 2: Python`
	`Book 3: C++`
	`4 ==========`
	`No more capacity`
	`5 ==========`
	`No more capacity`
	`6 ==========`
	`Name: Messi`
	`Capacity: 2`
	`Books:`
	`Book 1: C#`
	`Book 2: Rust`

Task 9

```
1  public class Task9 {
2     public int temp = 4;
3     public int sum;
4     public int y;
5     public int x;
6     public void methodA(int m){
7         int [] n = {2,5};
8         int x = 0;
9         y = y + m + this.methodB(x,m++)+(temp)+y;
10        x = this.x + 2 + (++n[0]);
11        sum = sum + x + y;
12        n[0] = sum + 2;
13       System.out.println(n[0] + x + " " + y+ " " + sum);
14        }
15    public int methodB(int m, int n){
16        int [] y = {1};
17        this.y = y[0] + this.y + m;
18        x = this.y + 2 + temp - n;
19        sum = x + y[0] + this.sum;
20        System.out.println( y[0]+ x + " " + y[0] + " " +sum);
21        return y[0];
22    }
23 }
```

```
public class Tester9 {
   public static void main(String [] args){
       Task9 t1 = new Task9();
       t1.methodA(5);
       t1.methodA(3);
       Task9 t2 = new Task9();
       t2.methodA(4);
   }
}
```

Outputs		

Task 10

```
1  public class Maze{
2          public int x;
3          public void methodA(){
4              int m = 0, x = 9;
5               m = methodB(m-3)+x;
6               this.x = ++x;
7              System.out.println(this.x+" "+m);
8              methodB(x,m);
9               System.out.println(x+" "+(m+this.x));
10              methodB(m);
11          }
12         public int methodB(int y){
13              x=y*y;
14              System.out.println(x+" "+y);
15             return x-11;
16          }
```

```
17          public void methodB(int z, int x){
18              z=z-2;
19              x=this.x-2*x;
20              System.out.println(z+" "+this.x);
21          }
22  }
```

DRIVER CODE	OUTPUTS	

public class MazeTester{
  public static void main(String args []){
  Maze m1 = new Maze();
  m1.methodA();
  }
}		

Task 11

```
1   public class Task11 {
2       int x = 2, y = 4, z = 5;
3       double p = 0.0;
4       public void methodA(int x, int m) {
5           this.x = methodC(this.x);
6           p = x + this.x % m * 3.0;
7           y = y + methodB(x++, this.x);
8           System.out.println(this.x +" " + x + y + " " + p) ;
9       }
10      public int methodB(int q, int n) {
```

```
11          int arr[] = {3,4,5};
12          arr[0] = arr[0] + this.x + n;
13          arr[1] = q + arr[1];
14          System.out.println(arr[0] +" " + arr[1] + " " + arr[2]) ;
15          return arr[1] + arr[2];
16      }
17      public int methodC(int y) {
18          if(y % 2 == 0) {
19              int temp = methodB(2, y);
20              return temp;
21          }
22          else{
23              return 4;
24         }
25      }
26  }
```

Driver Code	Output		

public class Tester11 {
public static void main(String [] args){
Task11 t1 = new Task11();
t1.methodA(2,3);
t1.methodB(5,4);
}
}			

Ungraded Tasks (Optional)

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

Task 1

You are building a tracker system that will keep track of a person's income and expenses.

- When the ***createTracker()*** method is invoked it sets the balance to 1.0 taka.
- The ***info()*** method **returns** a String with the trackers information.
- If the total balance becomes 0 after the ***expense()*** method is called it prints "You're broke!" . Again if the available balance is less than the expense it prints "Not enough balance.". Otherwise the method prints "Balance updated" after updating the balance.
- The last expense and income history can be seen by using the ***history()*** method.

Driver Code	Output

`public class Tester4{`
`  public static void main(String[] args) {`
`  MoneyTracker tr1 = new MoneyTracker();`
`  System.out.println(tr1.info());`
`  tr1.createTracker("John");`
`  System.out.println("1 ==========");`
`  System.out.println(tr1.info());`
`  System.out.println("2 ==========");`
`  tr1.income(1000);`
`  System.out.println(tr1.info());`
`  System.out.println("3 ==========");`
`  tr1.expense(800);`
`  tr1.expense(100);`
`  System.out.println(tr1.info());`
`  System.out.println("4 ==========");`
`  tr1.showHistory();`
`  System.out.println("5 ==========");`
`  tr1.expense(101);`
`  System.out.println("6 ==========");`
`  tr1.expense(200);`
`  System.out.println("7 ==========");`
`  tr1.income(200);`
`  tr1.showHistory();`
`  System.out.println("8 ==========");`
`  }`
`}`
	`Name: null`
	`Current Balance: 0.0`
	`1 ==========`
	`Name: John`
	`Current Balance: 1.0`
	`2 ==========`
	`Balance Updated!`
	`Name: John`
	`Current Balance: 1001.0`
	`3 ==========`
	`Balance Updated.`
	`Balance Updated.`
	`Name: John`
	`Current Balance: 101.0`
	`4 ==========`
	`Last added: 1000.0`
	`Last spent: 100.0`
	`5 ==========`
	`You're broke!`
	`6 ==========`
	`Not enough balance.`
	`7 ==========`
	`Balance Updated!`
	`Last added: 200.0`
	`Last spent: 100.0`
	`8 ==========`

Task 2

```
1  public class Test2 {
2      int x = 3, y = 1, z = -4;
3      double p = 2.5;
4      public void methodA(int n, int x) {
5          this.x = methodB(x, n);
6          p = this.x + n % x * 2.0;
7          y = (z++) + methodB(z, (int) p) + (++z);
8          System.out.println(this.x + " " + (n + y) + " " + (x + z)) ;
9      }
10     public int methodB(int q, int n) {
11         int arr[] = {2, -5, 6};
12         arr[0] = arr[2] - this.x + n;
13         arr[1] = q - arr[1];
14         arr[2] = arr[q % 3] + arr[n % 2];
15         System.out.println(arr[0] + " " + arr[1] + " " + arr[2]) ;
16         return arr[1] + arr[2] - arr[0];
17     }
18 }
```

```
public class Tester2{
    public static void main(String [] args){
        Test2 t = new Test2();
        t.methodA(3, 4);
    }
}
```

Outputs		

Task 3

```
1  public class Test3 {
2      int x = 2, y = 4, sum = 3;
3      int arr[] = {x, y, sum};
4      public void methodA(int x) {
5          arr[0] += methodB(y, this.x) + methodC(x);
6          System.out.println(x + " " + this.x + " " + sum);
7          arr[1] += this.x * (++y) / (sum % x);
8          System.out.println(y + " " + sum + " " + this.x);
9          arr[2] += methodC(x) + methodB(this.x, sum);
10         System.out.println(arr[0] + " " + arr[1] + " " + arr[2]);
11     }
12     public int methodB(int q, int n) {
13         int arr2[] = {7, 8};
14         int a = (arr2[0]++) - q;
15         int b = (++arr2[1]) - n;
16         return a + b;
17     }
18     public int methodC(int z) {
19         z = sum + methodB(x, sum) - z;
20         return z/2;
21     }
22 }
```

```
public class Tester3{
    public static void main(String [] args){
        Test3 t3 = new Test3();
        t3.methodA(7);
    }
}
```

Outputs		

Task 4

Driver Code	Output

`public class CustomerTester {`
`  public static void main(String[] args) {`
`  Customer c1 = new Customer();`
`  c1.createCustomer("John");`
`  System.out.println("1====================");`
`  c1.showCart();`
`  System.out.println("2====================");`
`  c1.addItem("Apple", 2);`
`  c1.addItem("Orange", 5);`
`  c1.addItem("Bread", 5);`
`  c1.addItem("Milk", 3);`
`  c1.addItem("Eggs", 2);`
`  System.out.println("3====================");`
`  c1.showCart();`
`  System.out.println("4====================");`
`  c1.calculatePrice();`
`  System.out.println("5====================");`
`  Customer c2 = new Customer();`
`  c2.createCustomer("Jane");`
`  c2.addItem("Apple", 2, "Orange", 5);`
`  c2.addItem("Chocolates", 15, "Bread", 5);`
`  c2.addItem("Milk", 3);`
`  System.out.println("6====================");`
`  c2.showCart();`
`  System.out.println("7====================");`
`  c2.calculatePrice();`
`  }`
`}`
	`1====================`
	`Customer: John`
	`2====================`
	`Apple added to cart`
	`Orange added to cart`
	`Bread added to cart`
	`Milk added to cart`
	`Cart is full`
	`3====================`
	`Customer: John`
	`Item: Apple Price: 2`
	`Item: Orange Price: 5`
	`Item: Bread Price: 5`
	`Item: Milk Price: 3`
	`4====================`
	`Total: 15`
	`5====================`
	`Apple and Orange added to cart`
	`Chocolates and Bread added to cart`
	`Cart is full`
	`6====================`
	`Customer: Jane`
	`Item: Apple Price: 2`
	`Item: Orange Price: 5`
	`Item: Chocolates Price: 15`
	`Item: Bread Price: 5`
	`7====================`
	`Total: 27`

Task 5

Driver Code	Sample Output

public class CalculatorTester {
  public static void main(String[] args) {
  Calculator calc = new Calculator();
 	
  System.out.println("1============");
  calc.add(10, 20);
  System.out.println("2============");
  calc.add(5, 15, 25);
  System.out.println("3============");
  calc.multiply(6, 7);
  System.out.println("4============");
  calc.multiply(2, 3, 4);
  System.out.println("5============");
  calc.multiply("Hello", 3);
  System.out.println("6============");
  calc.multiply("Java", 5);
  }
}
	1============
	30
	2============
	45
	3============
	42
	4============
	24
	5============
	Hello-Hello-Hello
	6============
	Java-Java-Java-Java-Java

Task 6

Driver Code	Sample Output

public class LibraryTest {
  public static void main(String[] args) {
 	
  Book book1 = new Book();
  book1.createBook("The Great Gatsby");
 	
  Book book2 = new Book();
  book2.createBook("1984", "George Orwell");
 	
  Book book3 = new Book();
  book3.createBook("To Kill a Mockingbird", "Harper Lee", "Fiction");
 	
  System.out.println("|---Book Customization---|");
  book1.customizeGenre("Classic");
  book1.customizePages(180);
 	
  book2.customizeGenre("Dystopian");
  book2.customizePages(328);
 	
  book3.customizePages(281);
 	
  System.out.println();
 	
  System.out.println("|---Library Inventory---|");
	|---Book Customization---|
	Updated genre of "The Great Gatsby" to Classic.
	Updated pages of "The Great Gatsby" to 180 pages.
	Updated genre of "1984" to Dystopian.
	Updated pages of "1984" to 328 pages.
	Updated pages of "To Kill a Mockingbird" to 281 pages.
	
	|---Library Inventory---|
	Title: The Great Gatsby, Author: Unknown, Genre: Classic, Pages: 180
	Title: 1984, Author: George Orwell, Genre: Dystopian, Pages: 328
	Title: To Kill a Mockingbird, Author: Harper Lee, Genre: Fiction, Pages: 281

book1.displayDetails();
book2.displayDetails();
book3.displayDetails();
}
}	

Task 7

Driver Code	Sample Output

public class MovieManagerTest {
public static void main(String[] args) {
Movie inception = new Movie();
inception.setMovieDetails("Inception", "Christopher Nolan", 8.8);
 	
System.out.println("1====================");
inception.addActors("Leonardo DiCaprio", "Joseph Gordon-Levitt");
inception.addActors("Ellen Page");
inception.showInfo();
 	
System.out.println("2====================");
Movie avengers = new Movie();
avengers.setMovieDetails("Avengers: Endgame", "Anthony Russo", 8.4);
avengers.addActors("Robert Downey Jr.", "Chris Evans", "Scarlett Johansson");
avengers.showInfo();
 	
System.out.println("3====================");
Movie parasite = new Movie();
parasite.setMovieDetails("Parasite", "Bong Joon-ho");
parasite.addActors("Song Kang-ho", "Choi Woo-shik");
parasite.updateRating(8.6);
parasite.showInfo();
 	
System.out.println("4====================");
parasite.updateRating(8.9);
parasite.showInfo();
}
}
	1====================
	Added actor "Leonardo DiCaprio" to "Inception".
	Added actor "Joseph Gordon-Levitt" to "Inception".
	Added actor "Ellen Page" to "Inception".
	Title: Inception
	Director: Christopher Nolan
	Rating: 8.8
	Actors: Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen Page
	2====================
	Added actor "Robert Downey Jr." to "Avengers: Endgame".
	Added actor "Chris Evans" to "Avengers: Endgame".
	Added actor "Scarlett Johansson" to "Avengers: Endgame".
	Title: Avengers: Endgame
	Director: Anthony Russo
	Rating: 8.4
	Actors: Robert Downey Jr., Chris Evans, Scarlett Johansson
	3====================
	Added actor "Song Kang-ho" to "Parasite".
	Added actor "Choi Woo-shik" to "Parasite".
	Updated rating of "Parasite" to 8.6
	Title: Parasite
	Director: Bong Joon-ho
	Rating: 8.6
	Actors: Song Kang-ho, Choi Woo-shik
	4====================
	Updated rating of "Parasite" to 8.9
	Title: Parasite
	Director: Bong Joon-ho
	Rating: 8.9
	Actors: Song Kang-ho, Choi Woo-shik

Task 8

Design the **Course** class with the necessary properties so that the given output is produced for the provided driver code.

Driver Class	Output

public class CourseTester2{
  public static void main(String [] args){
  Course c1 = new Course();
  c1.updateDetails("PL II", "CS11");
  System.out.println("--------1--------");
  c1.printDetails();
  System.out.println("--------2--------");
  c1.addContent("Overloading");
  c1.printDetails();
  System.out.println("--------3--------");
  c1.addContent("Encapsulation");
  c1.addContent("Static", "Polymorphism");
  c1.printDetails();
  System.out.println("--------4--------");
  c1.addContent("Inheritance");
  System.out.println("--------5--------");
  Course c2 = new Course();
  c2.updateDetails("DS", "CS22");
  c2.addContent("Stack");
  c2.addContent("Recursion","Tree");
  c2.addContent("Heap","Hashing");
  System.out.println("--------6--------");
  c2.printDetails();
  }
}
	--------1--------
	Course details:
	Course Name: PL II
	Course Code: CS11
	Course Syllabus:
	No content yet.
	--------2--------
	Overloading was added.
	Course details:
	Course Name: PL II
	Course Code: CS11
	Course Syllabus:
	Overloading
	--------3--------
	Encapsulation was added.
	Static was added.
	Polymorphism was added.
	Course details:
	Course Name: PL II
	Course Code: CS11
	Course Syllabus:
	Overloading, Encapsulation, Static, Polymorphism
	--------4--------
	Cannot add more content
	--------5--------
	Stack was added.
	Recursion was added.
	Tree was added.
	Heap was added.
	Cannot add more content
	--------6--------
	Course details:
	Course Name: DS
	Course Code: CS22
	Course Syllabus:
	Stack, Recursion, Tree, Heap