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);
	Shape Name: Circle
	Area: 78.54
	1---------------
	Shape Name: Triangle
	Area: 14.0
	2---------------
	Shape Name: Rectangle
	Area: 10.56

```
    System.out.println(circle.details());
    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 **Library** class with the necessary properties so that the given output is produced for the provided driver code.

Driver Code	Output

public class Tester{
  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
	Maximum capacity exceeds. 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
	Maximum capacity exceeds. You can't add more than 4 books
	8-------------
	Maximum Capacity: 4
	Total Books: 4
	Book list:
	Onnobhubon
	Ami
	Deyal
	Himu

Task 5

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 6

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---------------");
	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

std3.checkScholarshipEligibility();
  System.out.println("7---------------");
  std2.showDetails();
  System.out.println("8---------------");
  std3.showDetails();
  }
}
	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 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);
	====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====

```
    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();
  }
}
```

```
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

```
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 ==========");
   }
}
```

Output

```
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

Task 1

Design the **Student** class in such a way that it produces the following output.

Driver Code	Expected Output

`public class StudentTester{`
`  public static void main(String[] args){`
`  Student s1 = new Student("Harry", "CSE");`
`  System.out.println(s1.name);`
`  s1.updateName("Harry Potter");`
`  System.out.println(s1.accessName());`
`  System.out.println(s1.prog);`
`  s1.updateProgram("CS");`
`  String prog = s1.accessProgram();`
`  System.out.println(prog);`
`  }`
`}`
	`Harry`
	`Harry Potter`
	`CSE`
	`CS`

Task 2

Design the **Toy** class in such a way that it produces the following output

Driver Code	Expected Output

`public class ToyTester{`
`  public static void main(String[] args){`
`  Toy t1 = new Toy("Car", 230);`
`  System.out.println("1================");`
`  t1.updatePrice(250);`
`  System.out.println("2================");`
`  System.out.println(t1.name);`
`  t1.showPrice();`
`  System.out.println("3================");`
`  Toy t2 = new Toy("Robot", 450);`
`  System.out.println("4================");`
`  t2.updateName("Autobot");`
`  System.out.println("5================");`
`  System.out.println(t2.name);`
`  t2.showPrice();`
`  }`
`}`
	`A new toy has been made!`
	`1================`
	`2================`
	`Car`
	`price: 250 Taka`
	`3================`
	`A new toy has been made!`
	`4================`
	`Changing old name: Robot`
	`new name: Autobot`
	`5================`
	`Autobot`
	`price: 450 Taka`

Task 3

Design the **Shape2D** class in such a way that it produces the following output.

Driver Code	Expected Output

`public class Shape2DTester {`
`  public static void main(String[] args) {`
`  Shape2D sq = new Shape2D();`
`  System.out.println("---------1----------");`
`  sq.area();`
`  System.out.println("---------2----------");`
`  Shape2D rectangle = new Shape2D(5,6);`
`  System.out.println("---------3----------");`
`  rectangle.area();`
`  System.out.println("---------4----------");`
`  Shape2D tri1 = new Shape2D(5,6,"Triangle");`
`  System.out.println("---------5----------");`
`  tri1.area();`
`  System.out.println("---------6----------");`
`  Shape2D tri2 = new Shape2D(5,6,7);`
`  System.out.println("---------7----------");`
`  tri2.area();`
`  System.out.println("---------8----------");`
`  }`
`}`
	`A Square has been created with length: 5`
	`---------1----------`
	`The area of the Square is: 25.0`
	`---------2----------`
	`A Rectangle has been created with length: 5 and breadth:  6`
	`---------3----------`
	`The area of the Rectangle is: 30.0`
	`---------4----------`
	`A Triangle has been created with height: 5 and base:  6`
	`---------5----------`
	`The area of the Triangle is: 15.0`
	`---------6----------`
	`A Triangle has been created with the following sides: 5, 6, 7`
	`---------7----------`
	`The area of the Triangle is: 14.69`
	`---------8----------`

Task 4

Write "**Student**" class to show the following expected outputs

Note:

- A student can't take any course until the CGPA is set.
- A student cannot take more than 4 courses.
- A student with CGPA below 3 cannot take more than 3 courses.

Driver Code	Expected Output

public class StudentDriver {
  public static void main(String[] args){
  Student student1 = new Student(12345678);
  System.out.println("1----------");
  student1.addCourse("CSE110");
  System.out.println("2----------");
  student1.storeCG(2.5);
  student1.addCourse(" ");
  student1.addCourse("ENG101");
  student1.showAdvisee();
  System.out.println("3----------");
  student1.removeAllCourse();
  student1.showAdvisee();
  System.out.println("4----------");
  student1.storeID(54652365);
  String[] courses = {"SOC101","CSE111","ENG102"};
  student1.addCourse(courses);
  student1.showAdvisee();
  System.out.println("5----------");
  student1.addCourse("CSE230");
  student1.showAdvisee();
  System.out.println("6----------");
  Student student2 = new Student(975738383,3.7);
  System.out.println("7----------");
  String[] courses2 = {"CSE220","PHY112","MAT120","BUS101","CHN101"};
  student2.addCourse(courses2);
  student2.showAdvisee();
  }
}
	A student with ID 12345678 has been created.
	1----------
	Failed to add CSE110
	Set CG first
	2----------
	Student ID: 12345678, CGPA: 2.5
	Added courses are:
	CSE110 ENG101
	3----------
	Student ID: 12345678, CGPA: 2.5
	No courses added.
	4----------
	Student ID: 54652365, CGPA: 2.5
	Added courses are:
	SOC101 CSE111 ENG102
	5----------
	Failed to add CSE230
	CG is low. Can't add more than 3 courses.
	Student ID: 54652365, CGPA: 2.5
	Added courses are:
	SOC101 CSE111 ENG102
	6----------
	A student with ID 975738383 and cgpa 3.7 has been created.
	7----------
	Failed to add CHN101
	Maximum 4 courses allowed.
	Student ID: 975738383, CGPA: 3.7
	Added courses are:
	CSE220 PHY112 MAT120 BUS101

Task 5

Design the **Triangle** Class that will produce the following output. We will consider both triangles to have the same sides if all sides are equal in the same orientation/sequence only.

Types of Triangle:

- Equilateral: When all sides in the same orientation are equal.
- Isosceles: When any two sides of a triangle in the same orientation are equal.
- Scalene: When all sides are of different lengths.

Driver Code	Output

public class TriangleTester{
public static void main(String args[]){
Triangle t1 = new Triangle();
Triangle t2 = new Triangle();
Triangle t3 = new Triangle();
Triangle t4 = new Triangle();
 	
t1.updateSides(4, 4, 4);
t2.updateSides(4, 5, 6);
t3.updateSides(4, 5, 6);
t4.updateSides(5, 4, 6);
 	
t1.triangleDetails();
System.out.println("--------1--------");
System.out.println(t1.printTriangleType());
System.out.println("--------2--------");
t3.triangleDetails();
System.out.println(t3.printTriangleType());
System.out.println("--------3--------");
t4.triangleDetails();
System.out.println(t4.printTriangleType());
System.out.println("--------4--------");
t2.compareTrinagles(t3);
System.out.println("--------5--------");
t1.compareTrinagles(t2);
System.out.println("--------6--------");
t1 = t2;
t1.compareTrinagles(t2);
System.out.println("--------7--------");
t3.compareTrinagles(t4);
}
}
	Three sides of the triangle are: 4, 4, 4
	Perimeter: 12
	--------1--------
	This is an Equilateral Triangle.
	--------2--------
	Three sides of the triangle are: 4, 5, 6
	Perimeter: 15
	This is a Scalene Triangle.
	--------3--------
	Three sides of the triangle are: 5, 4, 6
	Perimeter: 15
	This is a Scalene Triangle.
	--------4--------
	Addresses are different but the sides of the triangles are equal.
	--------5--------
	Addresses, length of the sides and perimeter all are different.
	--------6--------
	These two triangle objects have the same address.
	--------7--------
	Only the perimeter of both triangles is equal.

Task 6

Write the **Teacher** and **Course** classes so that the TestTeacher class produces the outputs given.
Hint: A teacher can add a maximum of 3 courses.

Driver Code	Output

`public class TestTeacher{`
`  public static void main(String [] args){`
`  Teacher t1 = new Teacher("Matin Saad Abdullah","MSA");`
`  Teacher t2 = new Teacher("Mumit Khan","MMK");`
`  Teacher t3 = new Teacher("Sadia Hamid Kazi","SKZ");`
`  Course c1 = new Course("CSE 110");`
`  Course c2 = new Course("CSE 111");`
`  Course c3 = new Course("CSE 220");`
`  Course c4 = new Course("CSE 221");`
`  Course c5 = new Course("CSE 230");`
`  Course c6 = new Course("CSE 310");`
`  Course c7 = new Course("CSE 320");`
`  Course c8 = new Course("CSE 340");`
`  t1.addCourse(c1);`
`  t1.addCourse(c2);`
`  t2.addCourse(c3);`
`  t2.addCourse(c4);`
`  t2.addCourse(c5);`
`  t3.addCourse(c6);`
`  t3.addCourse(c7);`
`  t3.addCourse(c8);`
`  System.out.println("1=======================");`
`  t1.printDetail();`
`  System.out.println("2=======================");`
`  t2.printDetail();`
`  System.out.println("3=======================");`
`  t3.printDetail();`
`  }`
`}`
	`A new teacher has been created`
	`A new teacher has been created`
	`A new teacher has been created`
	`1=======================`
	`Name: Matin Saad Abdullah`
	`Initial: MSA`
	`List of courses:`
	`CSE 110`
	`CSE 111`
	`2=======================`
	`Name: Mumit Khan`
	`Initial: MMK`
	`List of courses:`
	`CSE 220`
	`CSE 221`
	`CSE 230`
	`3=======================`
	`Name: Sadia Hamid Kazi`
	`Initial: SKZ`
	`List of courses:`
	`CSE 310`
	`CSE 320`
	`CSE 340`

Task 7

Design the required class/es so that the following output is generated. Read the following description:

1. You may assume that to board a bus, a student must have the bus pass, and his/her destination must match the route of the bus.
2. Additionally, the default maximum capacity of the bus is 2.

Driver Code

```
public class BracuStudentTester {
 public static void main(String[] args) {
  BracuStudent st1 = new BracuStudent("Afif", "Mirpur");
  System.out.println("1==============");
  BracuStudent st2 = new BracuStudent("Shanto", "Motijheel");
  BracuStudent st3 = new BracuStudent("Taskin", "Mirpur");
  st1.showDetails();
  st2.showDetails();
  System.out.println("2==============");
  st3.showDetails();
  System.out.println("3==============");
  BracuBus bus1 = new BracuBus("Mirpur");
  BracuBus bus2 = new BracuBus("Azimpur", 5);
  bus1.showDetails();
  bus2.showDetails();
  System.out.println("4==============");
  st2.getPass();
  st3.getPass();
  System.out.println("5==============");
  st2.showDetails();
  st3.showDetails();
  System.out.println("6==============");
  bus1.board();
  System.out.println("7==============");
  bus1.board(st1, st2);
  System.out.println("8==============");
  st1.getPass();
  st2.updateHome("Mirpur");
  st1.showDetails();
  st2.showDetails();
  System.out.println("9==============");
  bus1.board(st1);
  bus1.board(st2, st3);
  System.out.println("10==============");
  bus1.showDetails();
 }
}
```

Output

```
1==============
Student Name: Afif
Lives in Mirpur
Have Bus Pass? false
Student Name: Shanto
Lives in Motijheel
Have Bus Pass? false
2==============
Student Name: Taskin
Lives in Mirpur
Have Bus Pass? false
3==============
Bus Route: Mirpur
Passenger Count: 0 (Max: 2)
Passengers on Board:
Bus Route: Azimpur
Passenger Count: 0 (Max: 5)
Passengers on Board:
4==============
5==============
Student Name: Shanto
Lives in Motijheel
Have Bus Pass? true
Student Name: Taskin
Lives in Mirpur
Have Bus Pass? true
6==============
No passengers
7==============
You don't have a bus pass!
You got on the wrong bus!
8==============
Student Name: Afif
Lives in Mirpur
Have Bus Pass? true
Student Name: Shanto
Lives in Mirpur
Have Bus Pass? true
9==============
Afif boarded the bus.
Shanto boarded the bus.
Bus is full!
10==============
Bus Route: Mirpur
Passenger Count: 2 (Max: 2)
Passengers on Board:
Afif Shanto
```

Task 8

Design the **Student** and the **Usis** class so that the following output is produced.
Note:

- A student's email, password, and login status are null by default while creating an object of the Student class.
- Your code should satisfy the conditions mentioned in the output only.
- Usis class will have two instance variables: totalAdvisee and an array of Student type to store the student object. The array will be updated inside the advising() method only when the advising is successful.
- Usis can take at most 5 advisees.

Driver Code

```
public class UsisTester {
  public static void main(String[] args) {
    Student rakib = new Student("Rakib", 12301455,
"CSE");
    Student roy = new Student("Roy", 12501345, "CS");
    System.out.println("1*************");
    Usis usisObj = new Usis();
    System.out.println("2*************");
    usisObj.login(rakib);
    System.out.println("3*************");
    usisObj.advising(rakib);
    System.out.println("4*************");
    rakib.email = "rakib@hotmail.com";
    rakib.password = "1234";
    System.out.println("5*************");
    usisObj.login(rakib);
    System.out.println("6*************");
    usisObj.advising(rakib);
    System.out.println("7*************");
    usisObj.advising(rakib, "CSE110", "PHY111", "MAT110",
"CSE260");
    System.out.println("8*************");
    usisObj.advising(rakib, "CSE110", "PHY111","MAT110");
    System.out.println("9*************");
    usisObj.allAdviseeInfo();
    System.out.println("10*************");
    roy.email = "roy@hotmail.com";
    roy.password = "abcd";
    usisObj.login(roy);
    System.out.println("11*************");
    usisObj.advising(roy, "CSE110", "ENG101", "PHY112");
    System.out.println("12*************");
    usisObj.allAdviseeInfo();
  }
}
```

Expected Output

```
Student object is created
Student object is created
1*************
Usis is ready to use!
2*************
Email and password need to be set.
3*************
Please login to advise courses!
4*************
5*************
Login successful
6*************
You haven't selected any courses.
7*************
You need special approval to take
more than 3 courses.
8*************
Advising successful!
9*************
Total Advisee: 1
Name: Rakib ID: 12301455
Department: CSE
Advised Courses:
CSE110 PHY111 MAT110
==============
10*************
Login successful
11*************
Advising successful!
12*************
Total Advisee: 2
Name: Rakib ID: 12301455
Department: CSE
Advised Courses:
CSE110 PHY111 MAT110
==============
Name: Roy ID: 12501345
Department: CS
Advised Courses:
```

```
CSE110 ENG101 PHY112
==============
```

Task 9

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

```
public class Tester9 {
   public static void main(String args []){
    B b1 = new B();
    B b2 = new B(b1);
    b1.methodA(1, 2);
    b2.methodB(3, 2);
   }
}
```

Outputs		

Task 10

```
1  public class msgClass{
2    public int content;
3  }
4  class FinalT5A{
5    public int sum = 2, y = 1, x = 1;
6    public void methodA(){
7      int x=6, y =0;
8      msgClass myMsg = new msgClass();
9      myMsg.content = this.x;
10     x = x + myMsg.content;
11     this.y = this.y + methodB(myMsg, myMsg.content);
12     System.out.println(x + " " + this.y+ " " + sum);
13     y =  this.y/2 + this.x;
14     x = y + sum/2;
15     sum = x + y + myMsg.content;
16     System.out.println(x + " " + y+ " " + sum);
17   }
18   public int methodB(msgClass mg2, int mg1){
19     int x = 0;
20     y = y + mg2.content;
21     mg2.content = y + mg1;
22     x = this.x + 3 + mg1;
23     sum = sum + x + y;
24     System.out.println(this.x + " " + this.y+ " " + sum);
25     mg2.content = sum - mg1 ;
26     return sum;
27   }
28 }
```

DRIVER CODE	OUTPUTS		

`public class Tester10{`
`  public static void main(String args []){`
`  FinalT5A fT5A = new FinalT5A();`
`  fT5A.methodA();`
`  }`
`}`			

Task 11

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

Driver Code	Output		

`public class Tester11 {`
`  public static void main(String args[]){`
`  int[] x = {35};`
`  A a1 = new A();`
`  A a2 = new A(-5,-7);`
`  a1.methodA(1, x);`
`  a2.methodA(1, x);`
`  }`
`}`			

Ungraded Tasks (Optional)

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

Task 1

Design the **Parcel** class in such a way that it produces the following output.

NOTE: For the method ***calcFee()***, if the delivery location is ***Dhanmondi***, then the location charge will be 50 taka or else it'll be free. Also, while calculating total fee, if the product weight is 0 the total_fee would also be 0.

Formula: fee = (weight * 20) + *location_charge (if any)*

Driver Code	Expected Output

`public class ParcelDriver {`
`  public static void main(String[] args){`
`  Parcel p1 = new Parcel();`
`  p1.printDetails();`
`  p1.name = "Spongebob";`
`  p1.printDetails();`
`  System.out.println("1**********");`
`  Parcel p2 = new Parcel("Bob the Builder");`
`  p2.weight = 15;`
`  p2.calcFee("Gulshan");`
`  p2.printDetails();`
`  System.out.println("2**********");`
`  p2.addWeight(25);`
`  p2.calcFee("Banani");`
`  p2.printDetails();`
`  System.out.println("3**********");`
`  Parcel p3 = new Parcel("Dora the Explorer", 10);`
`  p3.addWeight(15);`
`  p3.calcFee("Dhanmondi");`
`  p3.printDetails();`
`  }`
`}`
	`Set name first`
	`Name: Spongebob`
	`Total Weight: 0`
	`Total Fee: 0.0`
	`1**********`
	`Name: Bob the Builder`
	`Total Weight: 15`
	`Total Fee: 300.0`
	`2**********`
	`Updated Weight: 40`
	`Name: Bob the Builder`
	`Total Weight: 40`
	`Total Fee: 800.0`
	`3**********`
	`Updated Weight: 25`
	`Name: Dora the Explorer`
	`Total Weight: 25`
	`Total Fee: 550.0`

Task 2

Design the program to get the output as shown.

Hints:

- Create an array in the Team class to store the player's object
- Use constructor overloading technique for Team class

```
public class TeamTester {
  public static void main(String[] args) {
    Team b = new Team();
    b.updateName("Bangladesh");
    Player mashrafi =  new Player("Mashrafi", 42, 100);
    b.addPlayer(mashrafi);
    Player tamim =  new Player("Tamim", 35, 70);
    b.addPlayer(tamim);
    b.printDetail();
    System.out.println("==============");
    Team a = new Team("Australia");
    Player ponting =  new Player("Ponting", 50, 300);
    a.addPlayer(ponting);
    Player lee =  new Player("Lee", 49, 200);
    a.addPlayer(lee);
    a.printDetail();
  }
}
```

Output:

```
Team: Bangladesh
List of players:
Name: Mashrafi
Age: 42, Total Matches: 100
Name: Tamim
Age: 35, Total Matches: 70
==============
Team: Australia
List of players:
Name: Ponting
Age: 50, Total Matches: 300
Name: Lee
Age: 49, Total Matches: 200
```

Task 3

```
1  public class TracingX {
2    public int x, y = 1;
3    public int metA(int y){
4      y += x + 3;
5      int temp = y + this.y;
6      if (temp % 2 == 0){
7        return temp;
8      }
9      TracingX t = new TracingX();
10     t.y = this.x - (++x) + t.x;
11     this.y = y + t.metA(t.x);
12     System.out.println(x +" "+ y +" "+temp);
13     return temp+this.y;
14   }
15 }
```

Driver code:

```
public class TesterX {
 public static void main(String[] args) {
   TracingX t1 = new TracingX();
   t1.y = t1.x = 5;
   TracingX t2 = new TracingX();
   t2.x = t1.metA(2);
   t2.y = t2.metA(4);
   System.out.println(t1.y +t1.x +" "+t2.x +" "+t2.y);
 }
}
```

Output:

Task 3

Design the **Parcel** class in such a way that it produces the following output.

NOTE: For the method ***calcFee()***, if the delivery location is ***Dhanmondi***, then the location charge will be 50 taka or else it'll be free. Also, while calculating total fee, if the product weight is 0 the total_fee would also be 0.

Formula: fee = (weight * 20) + *location_charge (if any)*

Driver Code	Expected Output

`public class ParcelDriver {`
`  public static void main(String[] args){`
`  Parcel p1 = new Parcel();`
`  p1.printDetails();`
`  p1.name = "Spongebob";`
`  p1.printDetails();`
`  System.out.println("1**********");`
`  Parcel p2 = new Parcel("Bob the Builder");`
`  p2.weight = 15;`
`  p2.calcFee("Gulshan");`
`  p2.printDetails();`
`  System.out.println("2**********");`
`  p2.addWeight(25);`
`  p2.calcFee("Banani");`
`  p2.printDetails();`
`  System.out.println("3**********");`
`  Parcel p3 = new Parcel("Dora the Explorer", 10);`
`  p3.addWeight(15);`
`  p3.calcFee("Dhanmondi");`
`  p3.printDetails();`
`  }`
`}`
	`Set name first`
	`Name: Spongebob`
	`Total Weight: 0`
	`Total Fee: 0.0`
	`1**********`
	`Name: Bob the Builder`
	`Total Weight: 15`
	`Total Fee: 300.0`
	`2**********`
	`Updated Weight: 40`
	`Name: Bob the Builder`
	`Total Weight: 40`
	`Total Fee: 800.0`
	`3**********`
	`Updated Weight: 25`
	`Name: Dora the Explorer`
	`Total Weight: 25`
	`Total Fee: 550.0`

Task 4

Design the **Shape2D** class in such a way that it produces the following output.

Driver Code	Expected Output

`public class Shape2DTester {`
`  public static void main(String[] args) {`
`  Shape2D sq = new Shape2D(5);`
`  System.out.println("---------1----------");`
`  sq.area();`
`  System.out.println("---------2----------");`
`  Shape2D rectangle = new Shape2D(5,6);`
`  System.out.println("---------3----------");`
`  rectangle.area();`
`  System.out.println("---------4----------");`
`  Shape2D tri1 = new Shape2D(5,6,"Triangle");`
`  System.out.println("---------5----------");`
`  tri1.area();`
`  System.out.println("---------6----------");`
`  Shape2D tri2 = new Shape2D(5,6,7);`
`  System.out.println("---------7----------");`
`  tri2.area();`
`  System.out.println("---------8----------");`
`  }`
`}`
	`A Square has been created with length: 5`
	`---------1----------`
	`The area of the Square is: 25.0`
	`---------2----------`
	`A Rectangle has been created with length: 5 and breadth:  6`
	`---------3----------`
	`The area of the Rectangle is: 30.0`
	`---------4----------`
	`A Triangle has been created with height: 5 and base:  6`
	`---------5----------`
	`The area of the Triangle is: 15.0`
	`---------6----------`
	`A Triangle has been created with the following sides: 5, 6, 7`
	`---------7----------`
	`The area of the Triangle is: 14.69`
	`---------8----------`

Task 5

Write "**Book**" class to show the following output :

Driver Code	Output

public class BookTester {
  public static void main(String[] args) {
  System.out.println("< --------1--------->");
  Book b1 = new Book("The Alchemist");
  b1.displayDetails();
  System.out.println("< --------2--------->");
  Book b2 = new Book("1984", "George Orwell");
  b2.displayDetails();
  System.out.println("< --------3--------->");
  Book b3 = new Book("To Kill a Mockingbird", "Harper Lee", 300);
  b3.displayDetails();
  System.out.println("< --------4--------->");
  b1.setDetails(250);
  b1.displayDetails();
  System.out.println("< --------5--------->");
  b2.setDetails("Orwell", 350);
  b2.displayDetails();
  }
}
	< --------1-------->
	Title: The Alchemist
	< --------2-------->
	Title: 1984, Author: George Orwell
	< --------3-------->
	Title: To Kill a Mockingbird, Author: Harper Lee, Price: 300
	< --------4-------->
	Title: The Alchemist, Price: 250
	< --------5-------->
	Title: 1984, Author: Orwell, Price: 350

Task 6

Write the "**Product**" class to show the following output

Note: Make sure to use proper ***Encapsulation concepts*** **for the setter & getter methods**. All the attributes should have **Private** access.

Driver Code	Output

public class ProductTester{
  public static void main(String[] args) {
  System.out.println("< ---------1---------->");
  Product product1 = new Product();
  product1.displayInfo();
  System.out.println();
  System.out.println("< ---------2---------->");
  Product product2 = new Product("Laptop", 1200.00);
  product2.setQuantity(10);
  product2.displayInfo(true);
  System.out.println("< ---------3---------->");
  System.out.println("Retrieved Price: $" +
product2.getPrice());
  System.out.println("Retrieved Quantity: " +
product2.getQuantity());
  }
}
	< ---------1---------->
	Product Name: Unknown
	Price: $0.0
	< ---------2---------->Product Name: Laptop
	Price: $1200.0
	Quantity: 10
	< ---------3---------->Retrieved Price: $1200.0
	Retrieved Quantity: 10

Task 7

Write "**Student**" class to show the following expected outputs

Note:

- Make sure to use proper ***Encapsulation concepts* for the setter methods**. All the attributes should have **Private** access.
- A student can't take any course until the CGPA is set.
- A student cannot take more than 4 courses.
- A student with CGPA below 3 cannot take more than 3 courses.

Driver Code	Expected Output

`public class StudentDriver {`
`  public static void main(String[] args){`
`  System.out.println("-----------------");`
`  Student student1 = new Student(12345678);`
`  student1.addCourse("CSE110");`
`  student1.setCG(2.5);`
`  student1.addCourse("CSE110");`
`  student1.addCourse("ENG101");`
`  student1.showAdvisee();`
`  System.out.println("-----------------");`
`  student1.rmAllCourse();`
`  student1.showAdvisee();`
`  System.out.println("-----------------");`
`  student1.setID(54652365);`
`  String[] courses = {"SOC101","CSE111","ENG102"};`
`  student1.addCourse(courses);`
`  student1.showAdvisee();`
`  System.out.println("-----------------");`
`  student1.addCourse("CSE230");`
`  student1.showAdvisee();`
`  System.out.println("-----------------");`
`  Student student2 = new Student(975738383,3.7);`
`  String[] courses2 =`
`{"CSE220","PHY112","MAT120","BUS101","CHN101"};`
`  student2.addCourse(courses2);`
`  student2.showAdvisee();`
`  }`
`}`
	`-----------------`
	`Failed to add CSE110`
	`Set CG first`
	`Student ID: 12345678, CGPA: 2.5`
	`Added courses are:`
	`CSE110 ENG101`
	`-----------------`
	`Student ID: 12345678, CGPA: 2.5`
	`No courses added.`
	`-----------------`
	`Student ID: 54652365, CGPA: 2.5`
	`Added courses are:`
	`SOC101 CSE111 ENG102`
	`-----------------`
	`Failed to add CSE230`
	`CG is low. Can't add more than 3`
	`courses.`
	`Student ID: 54652365, CGPA: 2.5`
	`Added courses are:`
	`SOC101 CSE111 ENG102`
	`-----------------`
	`Failed to add CHN101`
	`Maximum 4 courses allowed.`
	`Student ID: 975738383, CGPA: 3.7`
	`Added courses are:`
	`CSE220 PHY112 MAT120 BUS101`

Task 8

Design "**ABCServer**" class to show the following output :

Driver Class	Output

public class ABCServerTester{
 public static void main (String args []){
  ABCServer server1 = new ABCServer();
  server1.details();
  System.out.println("---------------");
  ABCServer server2 = new ABCServer("Heroes Reborn",6);
  server2.details();
  System.out.println("---------------");
  server2.addMembers("Edward");
  server2.addMembers("William");
  System.out.println("----------------");
  server2.details();
  System.out.println("----------------");
  server2.addMembers("John", "Hero's Mentor");
  server2.addMembers("Albert", "Thunderstrike");
  server2.addMembers("Max", "Radiant Avenger");
  System.out.println("----------------");
  server2.details();
  System.out.println("----------------");
  server2.addMembers("Daniel");
  server2.addMembers("Donal", "Valor Knight");
  System.out.println("----------------");
  server2.details();
 }
}
	Server Name: Default
	Member Capacity: 10
	Total Members: 0
	Members:
	--------------- 	
	Server Name: Heroes Reborn
	Member Capacity: 6
	Total Members: 0
	Members:
	--------------- 	
	Rising Hero is added.
	Rising Hero is added.
	---------------- 	
	Server Name: Heroes Reborn
	Member Capacity: 6
	Total Members: 2
	Members:
	Name:Role --> Edward:Rising Hero
	Name:Role --> William:Rising Hero
	---------------- 	
	Hero's Mentor is added.
	Thunderstrike is added.
	Radiant Avenger is added.
	---------------- 	
	Server Name: Heroes Reborn
	Member Capacity: 6
	Total Members: 5
	Members:
	Name:Role --> Edward:Rising Hero
	Name:Role --> William:Rising Hero
	Name:Role --> John:Hero's Mentor
	Name:Role --> Albert:Thunderstrike
	Name:Role --> Max:Radiant Avenger
	---------------- 	
	Rising Hero is added.
	Sorry, maximum capacity exceeded!
	---------------- 	
	Server Name: Heroes Reborn
	Member Capacity: 6
	Total Members: 6

	Members: Name:Role --> Edward:Rising Hero Name:Role --> William:Rising Hero Name:Role --> John:Hero's Mentor Name:Role --> Albert:Thunderstrike Name:Role --> Max:Radiant Avenger Name:Role --> Daniel:Rising Hero

Task 9

```
1  public class Trace1{
2        public int[] q;
3        public int x, y;
4        public Trace1(int[] p){
5              this.q = p;
6              System.out.println(q[1]+" "+q[2]+" "+q[3]);
7        }
8        public int m2(int a, int b){
9              x = b++;
10             y = ++a/x;
11             q[x] = b + q[x];
12             if(b%2==0){
13                   q[b] = x;
14                   System.out.println(q[a]+" "+y+" "+x);
15                   this.m1(b,a);
16             }
17             else{
18                   System.out.println(x+" "+y+" "+q[x]);
19             }
20             return x+y;
21       }
22       public void m1(int x, double y){
23             this.y = (int)y;
24             System.out.println(q[x]+" "+(++x)+" "+y );
25             m2(this.y,x-1);
26       }
27 }
```

DRIVER CODE	OUTPUTS		

public class Main {
  public static void main(String[] args){
  int[] arr = {5,3,9,2,1};
  Trace1 t1 = new Trace1(arr);
  int x = t1.m2(7,2);
  System.out.println(arr[0]+" "+x+" "+arr[4]);
  t1.m1(2,7);
  }
}			

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 Puzzle{
2      public int x,z;
3      public Puzzle(int x){
4          this.x = x;
5      }
6      public Puzzle(int x, int z){
7          this.x = x;
8          this.z = z;
9      }
10     public void methodA(){
11         z=x+methodB(x);
12         System.out.println(x+" "+z);
13         methodB(x,z);
14     }
15     public int methodB(int y){
16         x=y+x;
17         System.out.println(x+" "+y);
18         return x+3;
19     }
20     public void methodB(int z, int x){
21         z=this.z+1;
22         x=x+1;
23         System.out.println(z+" "+x);
24     }
25 }
```

DRIVER CODE	OUTPUTS	

`public class PuzzleTester{`
`  public static void main(String[]args){`
`  Puzzle p = new Puzzle(5,8);`
`  Puzzle p1 = new Puzzle(8);`
`  p.methodA();`
`  System.out.println(p.methodB(7)+" "+p1.methodB(7));`
`  }`
`}`		

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

After YouTube Music, Spotify has decided to redesign their Playlist system. However, they decided to **not use arrays** to store their music, instead, they will use OOP concepts to create the new Playlist system. You have been assigned to build the system by using 3 classes (**Song**, **Playlist,** and **SpotifyTester**).
Each song will have the *name of the song*, *artist name*, *length of the song in minutes* and *the next song*. Each playlist will have a name and it can contain multiple songs. Both classes will have some features which will be demonstrated in each task.
[You are not allowed to use Array or any built-in libraries for this assignment]

Task 1

Design the **Song** class with *constructor* and *songInfo()* method along with necessary attributes in such a way that it produces the following output.

Driver Code	Output

`public class SpotifyTester {`
`  public static void main(String[] args) {`
`  Song s1 = new Song("Song-A", "Artist-A", 3);`
`  System.out.println("1==========");`
`  s1.songInfo();`
`  System.out.println("2==========");`
`//  More lines will be added in this Tester class`
	`1==========`
	`Title: Song-A`
	`Artist: Artist-A`
	`Length: 3 minutes`
	`2==========`

Task 2

Design the **Playlist** class constructor along with necessary attributes in such a way that it produces the following output.

Driver Code	Output

`System.out.println("2==========");`
`//  Continuation from Task 1`
`Playlist p1 = new Playlist("First Playlist");`
`System.out.println("3==========");`
	`2==========`
	`First Playlist created.`
	`3==========`

Task 3 & 4

Create *addSong()* method and *info()* method inside the **Playlist** class to produce the following output.

Driver Code	Output

`System.out.println("3==========");`
`//  Continuation from Task 2`
`p1.info();`
`System.out.println("4==========");`
`p1.addSong(s1);`
`System.out.println("5==========");`
`p1.info();`
`System.out.println("6==========");`
`Song s2 = new Song("Song-B", "Artist-B", 4);`
`Song s3 = new Song("Song-C", "Artist-C", 2);`
`p1.addSong(s2);`
`p1.addSong(s3);`
`System.out.println("7==========");`
`p1.info();`
`System.out.println("8==========");`
	`3==========`
	`First Playlist has the following songs:`
	`No songs in First Playlist.`
	`4==========`
	`Song-A added to First Playlist.`
	`5==========`
	`First Playlist has the following songs:`
	`Song-1`
	`Title: Song-A`
	`Artist: Artist-A`
	`Length: 3 minutes`
	`6==========`
	`Song-B added to First Playlist.`
	`Song-C added to First Playlist.`
	`7==========`
	`First Playlist has the following songs:`
	`Song-1`
	`Title: Song-A`
	`Artist: Artist-A`
	`Length: 3 minutes`
	`Song-2`
	`Title: Song-B`
	`Artist: Artist-B`
	`Length: 4 minutes`
	`Song-3`
	`Title: Song-C`
	`Artist: Artist-C`
	`Length: 2 minutes`
	`8==========`

Task 5

Create *addSong()* [overloaded] method inside the **Playlist** class to produce the following output.

Driver Code	Output

```
   System.out.println("8==========");
//    Continuation from Task 3&4
   Song s4 = new Song("Song-D", "Artist-D", 3);
   Song s5 = new Song("Song-E", "Artist-E", 4);
   Song s6 = new Song("Song-F", "Artist-F", 2);
   Song s7 = new Song("Song-G", "Artist-G", 2);
   p1.addSong(s4, 0);
   p1.addSong(s5, 2);
   p1.addSong(s6, 5);
   p1.addSong(s7, 10);
   System.out.println("9==========");
   p1.info();
   System.out.println("10==========");
```

```
8==========
Song-D added to First Playlist.
Song-E added to First Playlist.
Song-F added to First Playlist.
Cannot add song to Index 10.
9==========
First Playlist has the following
songs:
Song-1
Title: Song-D
Artist: Artist-D
Length: 3 minutes
Song-2
Title: Song-A
Artist: Artist-A
Length: 3 minutes
Song-3
Title: Song-E
Artist: Artist-E
Length: 4 minutes
Song-4
Title: Song-B
Artist: Artist-B
Length: 4 minutes
Song-5
Title: Song-C
Artist: Artist-C
Length: 2 minutes
Song-6
Title: Song-F
Artist: Artist-F
Length: 2 minutes
10==========
```

Task 6

Create *playSong()* method inside the **Playlist** class to produce the following output.

Driver Code	Output

`System.out.println("10==========");`
`//  Continuation from Task 5`
`p1.playSong("Song-F");`
`p1.playSong("Song-G");`
`p1.playSong("Song-B");`
`System.out.println("11==========");`
	`10==========`
	`Playing Song-F by Artist-F.`
	`Song-G not found in playlist First Playlist.`
	`Playing Song-B by Artist-B.`
	`11==========`

Task 7

Create the *playSong()* [overloaded] method inside the **Playlist** class to produce the following output.

Driver Code	Output

`System.out.println("11==========");`
`//    Continuation from Task 6`
`p1.playSong(0);`
`p1.playSong(4);`
`p1.playSong(8);`
`System.out.println("12==========");`
	`11==========`
	`Playing Song-D by Artist-D.`
	`Playing Song-C by Artist-C.`
	`Song at Index 8 not found in First Playlist.`
	`12==========`

Task 8

Create the *deleteSong()* method inside the **Playlist** class to produce the following output.

Driver Code	Output

`System.out.println("12==========");`
`//  Continuation from Task 7`
`p1.deleteSong("Song-D");`
`p1.deleteSong("Song-B");`
`p1.deleteSong("Song-F");`
`p1.deleteSong("Song-K");`
`System.out.println("13==========");`
`p1.info();`
`System.out.println("14==========");`
	`12==========`
	`Song-D deleted from First Playlist.`
	`Song-B deleted from First Playlist.`
	`Song-F deleted from First Playlist.`
	`Song-K not found in First Playlist.`
	`13==========`
	`First Playlist has the following songs:`
	`Song-1`
	`Title: Song-A`
	`Artist: Artist-A`
	`Length: 3 minutes`
	`Song-2`
	`Title: Song-E`
	`Artist: Artist-E`
	`Length: 4 minutes`
	`Song-3`
	`Title: Song-C`
	`Artist: Artist-C`
	`Length: 2 minutes`
	`14==========`

Task 9

Create the *totalSong()* method inside the **Playlist** class to produce the following output.

Driver Code	Output

`System.out.println("14==========");`
`//  Continuation from Task 8`
`System.out.println(p1.name +" has "+p1.totalSong() +" songs");`
`System.out.println("15==========");`
	`14==========`
	`First Playlist has 3 songs`
	`15==========`

Task 10

Create the *merge()* method inside the **Playlist** class to produce the following output.

Driver Code	Output

`System.out.println("15==========");`
`//  Continuation from Task 9`
`Song ns1 = new Song("Song-Z", "Artist-Z", 3);`
`Song ns2 = new Song("Song-Y", "Artist-Y", 4);`
`Song ns3 = new Song("Song-X", "Artist-X", 2);`
`System.out.println("16==========");`
`Playlist p2 = new Playlist("Second Playlist");`
`p2.addSong(ns1);`
`p2.addSong(ns2);`
`p2.addSong(ns3);`
`System.out.println("17==========");`
`p1.info();`
`System.out.println("18==========");`
`p2.info();`
`System.out.println("19==========");`
`p1.merge(p2);`
`System.out.println("20==========");`
`p1.info();`
`System.out.println("21==========");`
	15==========
	16==========
	Second Playlist created.
	Song-Z added to Second Playlist.
	Song-Y added to Second Playlist.
	Song-X added to Second Playlist.
	17==========
	First Playlist has the following songs:
	Song-1
	Title: Song-A
	Artist: Artist-A
	Length: 3 minutes
	Song-2
	Title: Song-E
	Artist: Artist-E
	Length: 4 minutes
	Song-3
	Title: Song-C
	Artist: Artist-C
	Length: 2 minutes
	18==========
	Second Playlist has the following songs:
	Song-1
	Title: Song-Z
	Artist: Artist-Z
	Length: 3 minutes
	Song-2
	Title: Song-Y
	Artist: Artist-Y
	Length: 4 minutes
	Song-3
	Title: Song-X
	Artist: Artist-X
	Length: 2 minutes
	19==========
	Merge Completed!
	20==========
	First Playlist has the following songs:
	Song-1
	Title: Song-A
	Artist: Artist-A
	Length: 3 minutes
	Song-2
	Title: Song-E
	Artist: Artist-E
	Length: 4 minutes
	Song-3
	Title: Song-C
	Artist: Artist-C
	Length: 2 minutes
	Song-4
	Title: Song-Z
	Artist: Artist-Z
	Length: 3 minutes
	Song-5
	Title: Song-Y
	Artist: Artist-Y
	Length: 4 minutes
	Song-6
	Title: Song-X
	Artist: Artist-X
	Length: 2 minutes
	21==========

Task 11 [Ungraded]

Create the *showHistory()* method inside the **Playlist** class to produce the following output. [Hint: showHistory() only shows the songs which were played from the playlist. So you might need to update the method which is used to play Songs.]

Driver Code	Output

`System.out.println("21==========");`
`//  Continuation from Task 10`
`p1.showHistory();`
`System.out.println("22==========");`
`p2.showHistory();`
`System.out.println("23==========");`
`}`
`}`
	21==========
	History of First Playlist:
	Title: Song-F
	Artist: Artist-F
	Length: 2 minutes
	Title: Song-B
	Artist: Artist-B
	Length: 4 minutes
	Title: Song-D
	Artist: Artist-D
	Length: 3 minutes
	Title: Song-C
	Artist: Artist-C
	Length: 2 minutes
	22==========
	History of Second Playlist:
	No songs were played from Second Playlist.
	23==========

Complete driver code and expected output:

Driver Code	Output

public class SpotifyTester {
 public static void main(String[] args) {
  Song s1 = new Song("Song-A", "Artist-A", 3);
  System.out.println("1==========");
  s1.songInfo();
  System.out.println("2==========");
  Playlist p1 = new Playlist("First Playlist");
  System.out.println("3==========");
  p1.info();
  System.out.println("4==========");
  p1.addSong(s1);
  System.out.println("5==========");
  p1.info();
  System.out.println("6==========");
  Song s2 = new Song("Song-B", "Artist-B", 4);
  Song s3 = new Song("Song-C", "Artist-C", 2);
  p1.addSong(s2);
  p1.addSong(s3);
  System.out.println("7==========");
  p1.info();
  System.out.println("8==========");
  Song s4 = new Song("Song-D", "Artist-D", 3);
  Song s5 = new Song("Song-E", "Artist-E", 4);
  Song s6 = new Song("Song-F", "Artist-F", 2);
  Song s7 = new Song("Song-G", "Artist-G", 2);
  p1.addSong(s4, 0);
  p1.addSong(s5, 2);
  p1.addSong(s6, 5);
  p1.addSong(s7, 10);
  System.out.println("9==========");
  p1.info();
  System.out.println("10==========");
  p1.playSong("Song-F");
  p1.playSong("Song-G");
  p1.playSong("Song-B");
  System.out.println("11==========");
  p1.playSong(0);
  p1.playSong(4);
  p1.playSong(8);
  System.out.println("12==========");
  p1.deleteSong("Song-D");
  p1.deleteSong("Song-B");
  p1.deleteSong("Song-F");
  p1.deleteSong("Song-K");
	1==========
	Title: Song-A
	Artist: Artist-A
	Length: 3 minutes
	2==========
	First Playlist created.
	3==========
	First Playlist has the following songs:
	No songs in First Playlist.
	4==========
	Song-A added to First Playlist.
	5==========
	First Playlist has the following songs:
	Song-1
	Title: Song-A
	Artist: Artist-A
	Length: 3 minutes
	6==========
	Song-B added to First Playlist.
	Song-C added to First Playlist.
	7==========
	First Playlist has the following songs:
	Song-1
	Title: Song-A
	Artist: Artist-A
	Length: 3 minutes
	Song-2
	Title: Song-B
	Artist: Artist-B
	Length: 4 minutes
	Song-3
	Title: Song-C
	Artist: Artist-C
	Length: 2 minutes
	8==========
	Song-D added to First Playlist.
	Song-E added to First Playlist.
	Song-F added to First Playlist.
	Cannot add song to Index 10.

```
   System.out.println("13==========");
   p1.info();
   System.out.println("14==========");
   System.out.println(p1.name +" has "+
p1.totalSong() + " songs");
   System.out.println("15==========");
   Song ns1 = new Song("Song-Z", "Artist-Z", 3);
   Song ns2 = new Song("Song-Y", "Artist-Y", 4);
   Song ns3 = new Song("Song-X", "Artist-X", 2);
   System.out.println("16==========");
   Playlist p2 = new Playlist("Second Playlist");
   p2.addSong(ns1);
   p2.addSong(ns2);
   p2.addSong(ns3);
   System.out.println("17==========");
   p1.info();
   System.out.println("18==========");
   p2.info();
   System.out.println("19==========");
   p1.merge(p2);
   System.out.println("20==========");
   p1.info();
   System.out.println("21==========");

 //Ungraded Task
   p1.showHistory();
   System.out.println("22==========");
   p2.showHistory();
   System.out.println("23==========");
 }
}
```

```
9==========
First Playlist has the following songs:
Song-1
Title: Song-D
Artist: Artist-D
Length: 3 minutes
Song-2
Title: Song-A
Artist: Artist-A
Length: 3 minutes
Song-3
Title: Song-E
Artist: Artist-E
Length: 4 minutes
Song-4
Title: Song-B
Artist: Artist-B
Length: 4 minutes
Song-5
Title: Song-C
Artist: Artist-C
Length: 2 minutes
Song-6
Title: Song-F
Artist: Artist-F
Length: 2 minutes
10==========
Playing Song-F by Artist-F.
Song-G not found in playlist First
Playlist.
Playing Song-B by Artist-B.
11==========
Playing Song-D by Artist-D.
Playing Song-C by Artist-C.
Song at Index 8 not found in First
Playlist.
12==========
Song-D deleted from First Playlist.
Song-B deleted from First Playlist.
Song-F deleted from First Playlist.
Song-K not found in First Playlist.
13==========
```

```
First Playlist has the following songs:
Song-1
Title: Song-A
Artist: Artist-A
Length: 3 minutes
Song-2
Title: Song-E
Artist: Artist-E
Length: 4 minutes
Song-3
Title: Song-C
Artist: Artist-C
Length: 2 minutes
14==========
First Playlist has 3 songs
15==========
16==========
Second Playlist created.
Song-Z added to Second Playlist.
Song-Y added to Second Playlist.
Song-X added to Second Playlist.
17==========
First Playlist has the following songs:
Song-1
Title: Song-A
Artist: Artist-A
Length: 3 minutes
Song-2
Title: Song-E
Artist: Artist-E
Length: 4 minutes
Song-3
Title: Song-C
Artist: Artist-C
Length: 2 minutes
18==========
Second Playlist has the following
songs:
Song-1
Title: Song-Z
Artist: Artist-Z
Length: 3 minutes
```

```
Song-2
Title: Song-Y
Artist: Artist-Y
Length: 4 minutes
Song-3
Title: Song-X
Artist: Artist-X
Length: 2 minutes
19==========
Merge Completed!
20==========
First Playlist has the following songs:
Song-1
Title: Song-A
Artist: Artist-A
Length: 3 minutes
Song-2
Title: Song-E
Artist: Artist-E
Length: 4 minutes
Song-3
Title: Song-C
Artist: Artist-C
Length: 2 minutes
Song-4
Title: Song-Z
Artist: Artist-Z
Length: 3 minutes
Song-5
Title: Song-Y
Artist: Artist-Y
Length: 4 minutes
Song-6
Title: Song-X
Artist: Artist-X
Length: 2 minutes
21==========
History of First Playlist:
Title: Song-F
Artist: Artist-F
Length: 2 minutes
Title: Song-B
```

	Artist: Artist-B Length: 4 minutes Title: Song-D Artist: Artist-D Length: 3 minutes Title: Song-C Artist: Artist-C Length: 2 minutes 22========== History of Second Playlist: No songs were played from Second Playlist. 23==========