Task 1

You are given the following "**University**" class:

```
public class University{
    public String name;
    public String country;
}
```

Now write a Java **tester** class named "**UniversityTester**".

a. Write the main method and create 2 objects of **University** class and print the location of the objects and print the instance variables of the objects. Are the location of the objects the same?

b. Now change the instance variables of the first object.
   name = "Imperial College London"
   country = "England"

   Now change the instance variables of the second object.
   name = "Brac University"
   country = "Bangladesh"

   Now check if the instance variables of both objects have changed or not and whether the instance variables of both objects are of the same value or not.

Task 2

Write the driver code of **"Test2"** class to generate the following output:

```
public class Test2{
    public static void main(String [] args){
    //Your code here
    }
}
```

Design Class	Output

`public class Circle {`
`  public double radius = 5;`
`}`
	`Radius of the circle is 5.0`
	`The area of the circle is 78.53981633974483`
	`The circumference of the circle is`
	`31.41592653589793`

Task 3

Design the **“Student”** class so that the main method prints the following:

Tester Class

```
public class Test3{
   public static void main(String [] args){
      Student s1 = new Student();
      System.out.println("Name of the Student: "+s1.name);
      System.out.println("ID of the Student: "+s1.id);
      s1.id = 123;
      System.out.println("ID of the Student: "+s1.id);
   }
}
```

Output

```
Name of the Student: Bob
ID of the Student: 1
ID of the Student: 123
```

Task 4

Write the code in java for the **“Vehicle”** class. The tester class and the output is given below:

Tester class	**Output**

`public class Tester4{`
   `public static void main(String [] args){`
      `Vehicle car = new Vehicle();`
      `System.out.println("Attributes of car object:");`
      `System.out.println(car.type);`
      `System.out.println(car.wheels);`
      `System.out.println(car.color);`
      `System.out.println("=========");`
      `Vehicle bike = new Vehicle();`
      `bike.type="Motor bike";`
      `bike.wheels=2;`
      `bike.color="Red";`
      `System.out.println("Attributes of bike object:");`
      `System.out.println(bike.type);`
      `System.out.println(bike.wheels);`
      `System.out.println(bike.color);`
    `}`
`}`
	`Attributes of car object:`
	`Car`
	`4`
	`White`
	`=========`
	`Attributes of bike object:`
	`Motor bike`
	`2`
	`Red`

Task 5

Write the code in java for the "**Tournament**" class. The tester class and the **output** is given below:

Tester class

```
public class Tester5{
  public static void main(String [] args){
    Tournament asiaCup = new Tournament();
    System.out.println(asiaCup.name+" "+
asiaCup.sportsType+" "+asiaCup.numberOfTeams+"
"+asiaCup.teams);
    System.out.println("**************");
    asiaCup.name="Asia Cup";
    asiaCup.sportsType="Cricket";
    asiaCup.numberOfTeams=4;
    asiaCup.teams = new String[] {"BD","IND","PAK","SL"};
    System.out.printf("%s %s Tournament is played between
%d teams\n",asiaCup.name, asiaCup.sportsType,
asiaCup.numberOfTeams);
    System.out.println("The teams are:");
    for(int i=0; i<asiaCup.teams.length; i++){
        System.out.println(asiaCup.teams[i]);
    }
  }
}
```

Output

```
null null 0 null
**************
Asia Cup Cricket Tournament
is played between 4 teams
The teams are:
BD
IND
PAK
SL
```

Task 6

Design the “**ImaginaryNumber**” to generate the **output** given below:

Tester Class	**Output**

`public class Tester6{`
`  public static void main(String [] args){`
`     ImaginaryNumber num1 = new ImaginaryNumber();`
`     num1.printNumber();`
`     System.out.println("1********");`
`     num1.realPart=3;`
`     num1.imaginaryPart=7;`
`     num1.printNumber();`
`     System.out.println("2********");`
`     ImaginaryNumber num2 = new ImaginaryNumber();`
`     num2.realPart=1;`
`     num2.imaginaryPart=9;`
`     num2.printNumber();`
`    }`
`}`
	`0 + 0i`
	`1********`
	`3 + 7i`
	`2********`
	`1 + 9i`

Task 7

Task 7

Complete the **"Cat"** class so the main method produces the following output:

Test Class

```
public class Test7{
    public static void main(String [] args){
        Cat c1 = new Cat();
        System.out.println("===================");
        c1.printCat();
        c1.color = "Black";
        System.out.println("===================");
        c1.printCat();
        c1.color = "Brown";
        c1.action = "jumping";
        System.out.println("===================");
        c1.printCat();
    }
}
```

Output

```
===================
White cat is sitting
===================
Black cat is sitting
===================
Brown cat is jumping
```

Task 8

Complete the **Bird** class so that main method produces the following **output**:

Test class	Output

`public class Test8{`
`    public static void main(String args[]) {`
`        Bird b1 = new Bird();`
`        b1.name = "Parrot";`
`        b1.flyUp(3);`
`        b1.makeNoise();`
`        b1.flyDown(5);`
`        b1.flyDown(2);`
`        b1.flyDown(1);`
`        Bird b2 = new Bird();`
`        b2.name = "Eagle";`
`        b2.flyUp(5);`
`        b2.flyDown(5);`
`        b2.makeNoise();`
`    }`
`}`
	`Parrot has flown up 3 feet.`
	`Squawk`
	`Parrot cannot fly down 5 feet.`
	`Parrot has flown down 2 feet.`
	`Parrot has flown down 1 feet and`
	`landed.`
	`Eagle has flown up 5 feet.`
	`Eagle has flown down 5 feet and`
	`landed.`
	`Squee`

Task 9

Design the **CellPhone** class so that the **main** method of tester class can produce the following output:

Tester Class	Output

`public class Tester9{`
`  public static void main(String[]args){`
`     CellPhone phone1 = new CellPhone();`
`     phone1.printDetails();`
`     phone1.model ="Nokia 1100";`
`     System.out.println("1##################");`
`     phone1.storeContact("Joy - 01834");`
`     System.out.println("===================");`
`     phone1.printDetails();`
`     System.out.println("2##################");`
`     phone1.storeContact("Toya - 01334");`
`     phone1.storeContact("Aayan - 01135");`
`     System.out.println("===================");`
`     phone1.printDetails();`
`     System.out.println("3##################");`
`     phone1.storeContact("Sani - 01441");`
`     System.out.println("===================");`
`     phone1.printDetails();`
`  }`
`}`
	`Phone Model unknown`
	`Contacts Stored 0`
	`1##################`
	`Contact Stored`
	`===================`
	`Phone Model Nokia 1100`
	`Contacts Stored 1`
	`Stored Contacts:`
	`Joy - 01834`
	`2##################`
	`Contact Stored`
	`Contact Stored`
	`===================`
	`Phone Model Nokia 1100`
	`Contacts Stored 3`
	`Stored Contacts:`
	`Joy - 01834`
	`Toya - 01334`
	`Aayan - 01135`
	`3##################`
	`Memory full. New contact can't be`
	`stored.`
	`===================`
	`Phone Model Nokia 1100`
	`Contacts Stored 3`
	`Stored Contacts:`
	`Joy - 01834`
	`Toya - 01334`
	`Aayan - 01135`

Task 10

Consider the following class:

```
public class Human{
    public int age;
    public double height;
}
```

Show the output of the following sequence of statements:

```
Human h1 = new Human();
Human h2 = new Human();
h1.age = 21;
h1.height = 5.5;
System.out.println(h1.age);
System.out.println(h1.height);
h2.height = h1.height - 3;
System.out.println(h2.height);
h2.age = h1.age++;
System.out.println(h1.age);
h2 = h1;
System.out.println(h2.age);
System.out.println(h2.height);
h2.age++;
h2.height++;
System.out.println(h1.age);
System.out.println(h1.height);
h1.age = ++h2.age;
System.out.println(h2.age);
System.out.println(h2.height);
```

Output

Task 11

Consider the following class:

```
public class Student{
    public String name;
    public double cgpa;
}
```

Show the output of the following sequence of statements:

	Output
`Student s1 = new Student();`	
`Student s2 = new Student();`	
`Student s3 = null;`	
`s1.name = "Student One";`	
`s1.cgpa = 2.3;`	
`s3 = s1;`	
`s2.name = "Student Two";`	
`s2.cgpa = s3.cgpa + 1;`	
`s3.name = "New Student";`	
`System.out.println(s1.name);`	
`System.out.println(s2.name);`	
`System.out.println(s3.name);`	
`System.out.println(s1.cgpa);`	
`System.out.println(s2.cgpa);`	
`System.out.println(s3.cgpa);`	
`s3 = s2;`	
`s1.name = "old student";`	
`s2.name = "older student";`	
`s3.name = "oldest student";`	
`s2.cgpa = s1.cgpa - s3.cgpa + 4.5;`	
`System.out.println(s1.name);`	
`System.out.println(s2.name);`	
`System.out.println(s3.name);`	
`System.out.println(s1.cgpa);`	
`System.out.println(s2.cgpa);`	
`System.out.println(s3.cgpa);`	

Task 2

Design the **“Student”** class so that the main method prints the following:

Tester Class	Output
`public class StudentTester1{`	`Name of the Student: Default`
`   public static void main(String [] args){`	`ID of the Student: 0`
`      Student s1 = new Student();`	`Name of the Student: Bob`
`      System.out.println("Name of the Student: "+s1.name);`	`ID of the Student: 123`
`      System.out.println("ID of the Student: "+s1.id);`	
`      s1.name = "Bob";`	
`      s1.id = 123;`	
`      System.out.println("Name of the Student: "+s1.name);`	
`      System.out.println("ID of the Student: "+s1.id);`	
`   }`	
`}`	

Task 3

Design the **CSECourse** class to generate the correct output from the driver code provided below:

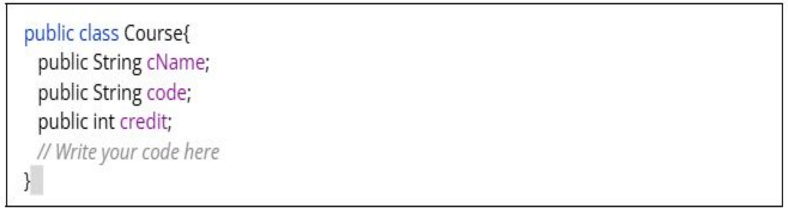
Driver Code

```java
public class CourseTester{
  public static void main(String args []){
    CSECourse c1 = new CSECourse();
    System.out.println("Course Name: "+c1.courseName);
    System.out.println("Course Code: "+c1.courseCode);
    System.out.println("Credit: "+c1.credit);
   }
}
```

Output

```
Course Name: Programming
Language II
Course Code: CSE111
Credit: 3
```

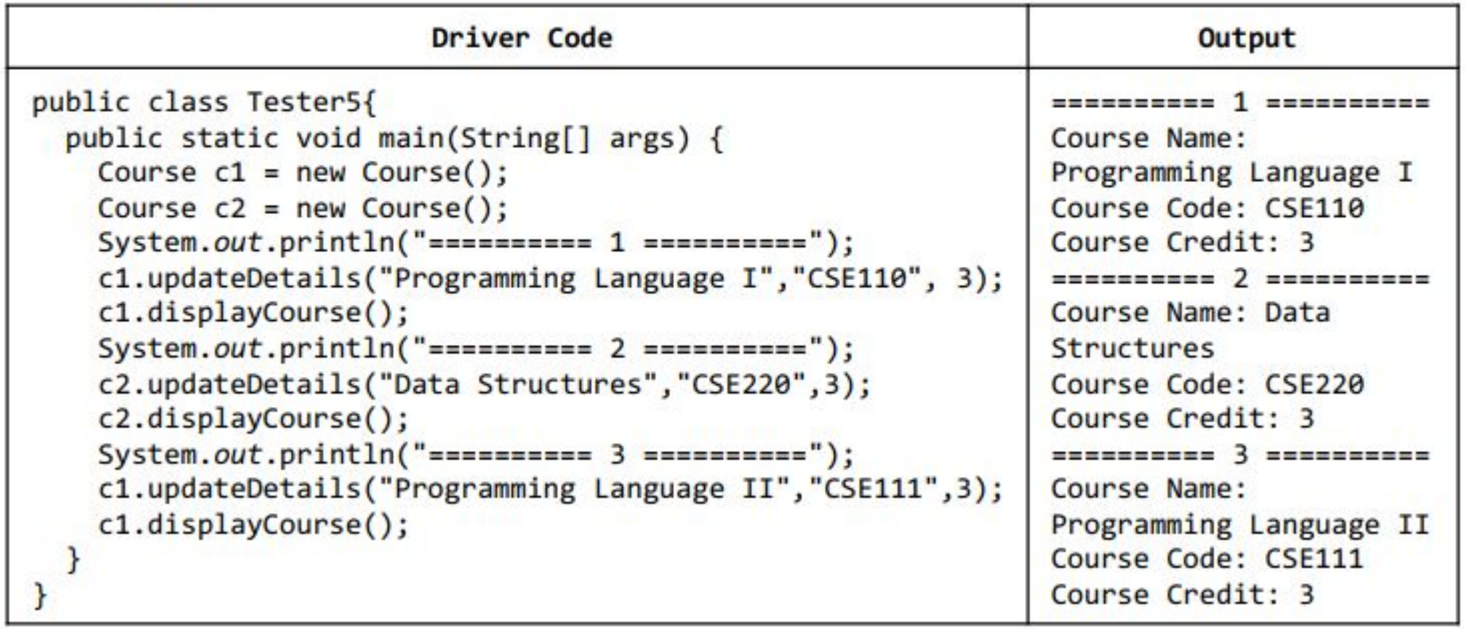
Course Class:



```java
public class Course{
  public String cName;
  public String code;
  public int credit;
  // Write your code here
}
```

Task 5

Design the **Course** class to generate the correct output from the driver code provided below:



Driver Code

```java
public class Tester5{
  public static void main(String[] args) {
    Course c1 = new Course();
    Course c2 = new Course();
    System.out.println("========== 1 ==========");
    c1.updateDetails("Programming Language I","CSE110", 3);
    c1.displayCourse();
    System.out.println("========== 2 ==========");
    c2.updateDetails("Data Structures","CSE220",3);
    c2.displayCourse();
    System.out.println("========== 3 ==========");
    c1.updateDetails("Programming Language II","CSE111",3);
    c1.displayCourse();
  }
}
```

Output

```
========== 1 ==========
Course Name:
Programming Language I
Course Code: CSE110
Course Credit: 3
========== 2 ==========
Course Name: Data
Structures
Course Code: CSE220
Course Credit: 3
========== 3 ==========
Course Name:
Programming Language II
Course Code: CSE111
Course Credit: 3
```

Task 6

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

Driver Class

```java
public class AssignmentTester{
  public static void main(String [] args){
    Assignment as1 = new Assignment();
    as1.printDetails();
    System.out.println("1--------------");
    as1.tasks = 11;
    as1.difficulty = "Moderate";
    as1.submission = true;
    as1.printDetails();
    System.out.println("2--------------");
    System.out.println(as1.makeOptional());
    System.out.println("3--------------");
    as1.printDetails();
    System.out.println("4--------------");
    Assignment as2 = new Assignment();
    as2.tasks = 12;
    as2.difficulty = "Hard";
    as2.submission = false;
    as2.printDetails();
    System.out.println("5--------------");
    System.out.println(as2.makeOptional());
  }
}
```

Output

```
Number of tasks: 0
Difficulty level: null
Submission required: false
1--------------
Number of tasks: 11
Difficulty level: Moderate
Submission required: true
2--------------
Assignment will not require
submission
3--------------
Number of tasks: 11
Difficulty level: Moderate
Submission required: false
4--------------
Number of tasks: 12
Difficulty level: Hard
Submission required: false
5--------------
Submission is already not required
```

Task 8

Create an **Employee** class to provide the expected output.

- An employee will have a name, salary and designation.
- The name will be assigned inside the newEmployee() method
- Whenever a New Employee joins his/her salary will be **Tk. 30,000** and the designation will be **junior**.
- Employees with salaries greater than **Tk. 50,000** and **Tk. 30,000** need to pay **30%** and **10%** of salary as tax respectively.
- Employees can be promoted to **senior**, **lead** and **manager** positions. Based on their promotion they will get an increment of **Tk. 25,000**, **Tk. 50,000** and **Tk. 75,000** respectively.

Driver Code

```java
public class Tester3{
  public static void main(String[] args){

    Employee emp1 = new Employee();
    Employee emp2 = new Employee();
    Employee emp3 = new Employee();

    emp1.newEmployee("Harry Potter");
    emp2.newEmployee("Hermione Granger");
    emp3.newEmployee("Ron Weasley");
    System.out.println("1 ==========");
    emp1.displayInfo();
    System.out.println("2 ==========");
    emp2.displayInfo();
    System.out.println("3 ==========");
    emp3.displayInfo();
    System.out.println("4 ==========");
    emp1.calculateTax();
    System.out.println("5 ==========");
    emp1.promoteEmployee("lead");
    System.out.println("6 ==========");
    emp1.calculateTax();
    System.out.println("7 ==========");
    emp1.displayInfo();
    System.out.println("8 ==========");
    emp3.promoteEmployee("manager");
    System.out.println("9 ==========");
    emp3.calculateTax();
    System.out.println("10 ==========");
    emp3.displayInfo();
  }
}
```

Expected Output

```
1 ==========
Employee Name: Harry Potter
Employee Salary: 30000.0 Tk
Employee Designation: junior
2 ==========
Employee Name: Hermione Granger
Employee Salary: 30000.0 Tk
Employee Designation: junior
3 ==========
Employee Name: Ron Weasley
Employee Salary: 30000.0 Tk
Employee Designation: junior
4 ==========
No need to pay tax
5 ==========
Harry Potter has been promoted to lead
New Salary: 80000.00 Tk
6 ==========
Harry Potter Tax Amount: 24000.00 Tk
7 ==========
Employee Name: Harry Potter
Employee Salary: 80000.0 Tk
Employee Designation: lead
8 ==========
Ron Weasley has been promoted to manager
New Salary: 105000.00 Tk
9 ==========
Ron Weasley Tax Amount: 31500.00 Tk
10 ==========
Employee Name: Ron Weasley
Employee Salary: 105000.0 Tk
Employee Designation: manager
```

Task 11

```
public class Task11 {
   public int p = 3, y = 2, sum;
   public void methodA(){
       int x = 0, y = 0;
       y = y + this.y;
       x = sum + 2 + p;
       sum = x + y + methodB(p, y);
       System.out.println(x + " " + y+ " " + sum);
   }
   public int methodB(int p, int n){
       int x = 0;
       y = y + (++p);
       x = x + 2 + n;
       sum = sum + x + y;
       System.out.println(x + " " + y+ " " + sum);
       return sum;
   }
}
```

Driver code:

```
public class Tester11 {
   public static void main(String [] args){
       Task11 t1 = new Task11 ();
       t1.methodA();
       t1.methodA();
   }
}
```

Outputs		
x	y	Sum

Ungraded Tasks (Optional)

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

Task 1

Complete the **"Cat"** class so the main method produces the following output:

Test Class

```java
public class Test7{
    public static void main(String [] args){
        Cat c1 = new Cat();
        System.out.println("===================");
        c1.printCat();
        c1.color = "Black";
        System.out.println("===================");
        c1.printCat();
        c1.color = "Brown";
        c1.action = "jumping";
        System.out.println("===================");
        c1.printCat();
    }
}
```

Output

```
===================
White cat is sitting
===================
Black cat is sitting
===================
Brown cat is jumping
```

Task 3

Implement the “**ChickenBurger**” class with necessary properties, so that the given output is produced for the provided driver code.
[**Note:**
1. There are four available spice levels: **Mild**, **Spicy**, **Naga** and **Extreme**. You can store these values in a String array.
2. You might need to use the *.equals()* method to compare two string values.]

Driver Class

```java
public class BurgerMaker{
 public static void main(String [] args){
   ChickenBurger b1 = new ChickenBurger();
   System.out.println(b1.bun);
   System.out.println(b1.price);
   System.out.println(b1.sauceOption);
   System.out.println(b1.spiceLevel);
   System.out.println("----------1---------");
   System.out.println(b1.serveBurger());
   System.out.println("----------2---------");
   b1.customizeSpiceLevel("Extreme Jhaal");
   b1.customizeSpiceLevel("Spicy");
   System.out.println("----------3---------");
   System.out.println(b1.serveBurger());
   System.out.println("----------4---------");
   ChickenBurger b2 = new ChickenBurger();
   b2.bun = "Brioche";
   b2.price += 50;
   b2.sauceOption = "Regular";
   b2.customizeSpiceLevel("Naga");
   System.out.println("----------5---------");
   System.out.println(b2.serveBurger());
 }
}
```

Output

```
Sesame
200
Less
Not Set
----------1---------
Cannot serve now. Customize Spice
Level first.
----------2---------
This spice level is unavailable.
Spice level set to Spicy.
----------3---------
The burger is being served:-
Bun Type: Sesame
Price: 200
Sauce Option: Less
Spice Level: Spicy
----------4---------
Spice level set to Naga.
----------5---------
The burger is being served:-
Bun Type: Brioche
Price: 250
Sauce Option: Regular
Spice Level: Naga
```

Task 5

```
public class Test1{
  public int sum;
  public int y;
  public void methodA(){
    int x=2, y =3;
    int [] msg ={3, 7};
    y = this.y + msg[0];
    methodB(msg[1]++, msg[0]);
    x = x + this.y + msg[1];
    sum = x + y + msg[0];
    System.out.println(x + " " + y+ " " + sum);
  }
  public void methodB(int mg2, int mg1){
    int x = 0;
    y = this.y + mg2;
    x = x + 19 + mg1;
    sum = this.sum + x + y;
    mg2 = y + mg1;
    mg1 = mg2 + x + 2;
    System.out.println(x + " " + y+ " " + sum);
  }
}
```

```
public class Tester5{
  public static void main (String args[]){
    Test1 t1 = new Test1();
    t1.methodB(5,-8);
    Test1 t2 = new Test1();
    t2.methodA();
  }
}
```

Outputs		

Task 2

Implement the “**Shelf**” class with necessary properties, so that the given output is produced for the provided driver code.

Driver Class

```
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();
  }
}
```

Output

```
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 3

Implement the "**LightController**" class with necessary properties to produce the given output for the provided driver code.

Driver Class

```
public class LightControllerTester{
  public static void main(String args []){
    LightController c1 = new LightController();
    c1.showLightStatus();
    System.out.println("1---------------");
    c1.adjustBrightness(4);
    c1.switchLight();
    System.out.println("2---------------");
    c1.showLightStatus();
    System.out.println("3---------------");
    c1.adjustBrightness(4);
    System.out.println("4---------------");
    c1.showLightStatus();
    System.out.println("5---------------");
    c1.adjustBrightness(-2);
    c1.adjustBrightness(9);
    System.out.println("6---------------");
    c1.showLightStatus();
    System.out.println("7---------------");
    System.out.println(c1.resetSettings());
    c1.showLightStatus();
    System.out.println("8---------------");
    c1.switchLight();
    System.out.println("9---------------");
    c1.showLightStatus();
  }
}
```

Output

```
Light status: OFF
Brightness Level: 0
1---------------
Please turn on the light first!
Lights are now ON.
2---------------
Light status: ON
Brightness Level: 1
3---------------
Brightness adjusted.
4---------------
Light status: ON
Brightness Level: 5
5---------------
Brightness adjusted.
Brightness out of range. Set between
0 to 10.
6---------------
Light status: ON
Brightness Level: 3
7---------------
Light settings have been reset.
Light status: ON
Brightness Level: 1
8---------------
Lights are now OFF.
9---------------
Light status: OFF
Brightness Level: 0
```

Task 6

Implement the "**Course**" class with necessary properties, so that the given output is produced for the provided driver code.
[Note: Each course can have at max 4 contents in its syllabus]

Driver Class

```java
public class CourseTester{
  public static void main(String [] args){
    Course c1 = new Course();
    c1.createCourse("PL II", "CS11");
    System.out.println("--------1--------");
    c1.printDetails();
    System.out.println("--------2--------");
    c1.addOneContent("Overloading");
    c1.printDetails();
    System.out.println("--------3--------");
    c1.addOneContent("Encapsulation");
    c1.addTwoContent("Static", "Polymorphism");
    c1.printDetails();
    System.out.println("--------4--------");
    c1.addOneContent("Inheritance");
    System.out.println("--------5--------");
    Course c2 = new Course();
    c2.createCourse("DS", "CS22");
    c2.addOneContent("Stack");
    c2.addTwoContent("Recursion","Tree");
    c2.addTwoContent("Heap","Hashing");
    System.out.println("--------6--------");
    c2.printDetails();
  }
}
```

Output

```
--------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 7

Task 7 looks very much similar to Task 6. But there are some slight differences. Can you figure those out? Once you figure those out, write the differences as a comment in your code and create **"Course2"** class.

Driver Class

```
public class CourseTester2{
  public static void main(String [] args){
    Course2 c1 = new Course2();
    c1.createCourse("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--------");
    Course2 c2 = new Course2();
    c2.createCourse("DS", "CS22");
    c2.addContent("Stack");
    c2.addContent("Recursion","Tree");
    c2.addContent("Heap","Hashing");
    System.out.println("--------6--------");
    c2.printDetails();
  }
}
```

Output

```
--------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 8

Implement 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 trapezium = new Shape();`
 	
`    circle.setParameters("Circle", 5);`
`    triangle.setParameters("Triangle", 4, 7);`
`    trapezium.setParameters("Trapezium", 2, 4, 9);`
 	
`    System.out.println(circle.details());`
`    System.out.println("---------------");`
`    System.out.println(triangle.details());`
`    System.out.println("---------------");`
`    System.out.println(trapezium.details());`
`  }`
`}`
	`Shape Name: Circle`
	`Area: 78.54`
	`---------------`
	`Shape Name: Triangle`
	`Area: 14.0`
	`---------------`
	`Shape Name: Trapezium`
	`Area: 27.0`

Task 9

```
public class Test1{
  public int sum;
  public int y;
  public void methodA(){
    int x=2, y =3;
    int [] msg ={3, 7};
    y = this.y + msg[0];
    methodB(msg[1]++, msg[0]);
    x = x + this.y + msg[1];
    sum = x + y + msg[0];
    System.out.println(x + " " + y+ " " + sum);
  }
  public void methodB(int mg2, int mg1){
    int x = 0;
    y = this.y + mg2;
    x = x + 19 + mg1;
    sum = this.sum + x + y;
    mg2 = y + mg1;
    mg1 = mg2 + x + 2;
    System.out.println(x + " " + y+ " " + sum);
  }
}
```

```
public class Tester1{
  public static void main (String args[]){
    Test1 t1 = new Test1();
    t1.methodB(5,-8);
    Test1 t2 = new Test1();
    t2.methodA();
  }
}
```

Outputs		

Task 10

```java
public class Test2 {
    int x = 3, y = 1, z = -4;
    double p = 2.5;
    public void methodA(int n, int x) {
        this.x = methodB(x, n);
        p = this.x + n % x * 2.0;
        y = (z++) + methodB(z, (int) p) + (++z);
        System.out.println(this.x + " " + (n + y) + " " + (x + z)) ;
    }
    public int methodB(int q, int n) {
        int arr[] = {2, -5, 6};
        arr[0] = arr[2] - this.x + n;
        arr[1] = q - arr[1];
        arr[2] = arr[q % 3] + arr[n % 2];
        System.out.println(arr[0] + " " + arr[1] + " " + arr[2]) ;
        return arr[1] + arr[2] - arr[0];
    }
}
```

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

Outputs		

Task 4

Implement the "**MobilePhone**" class with necessary properties to produce the given output for the provided driver code.

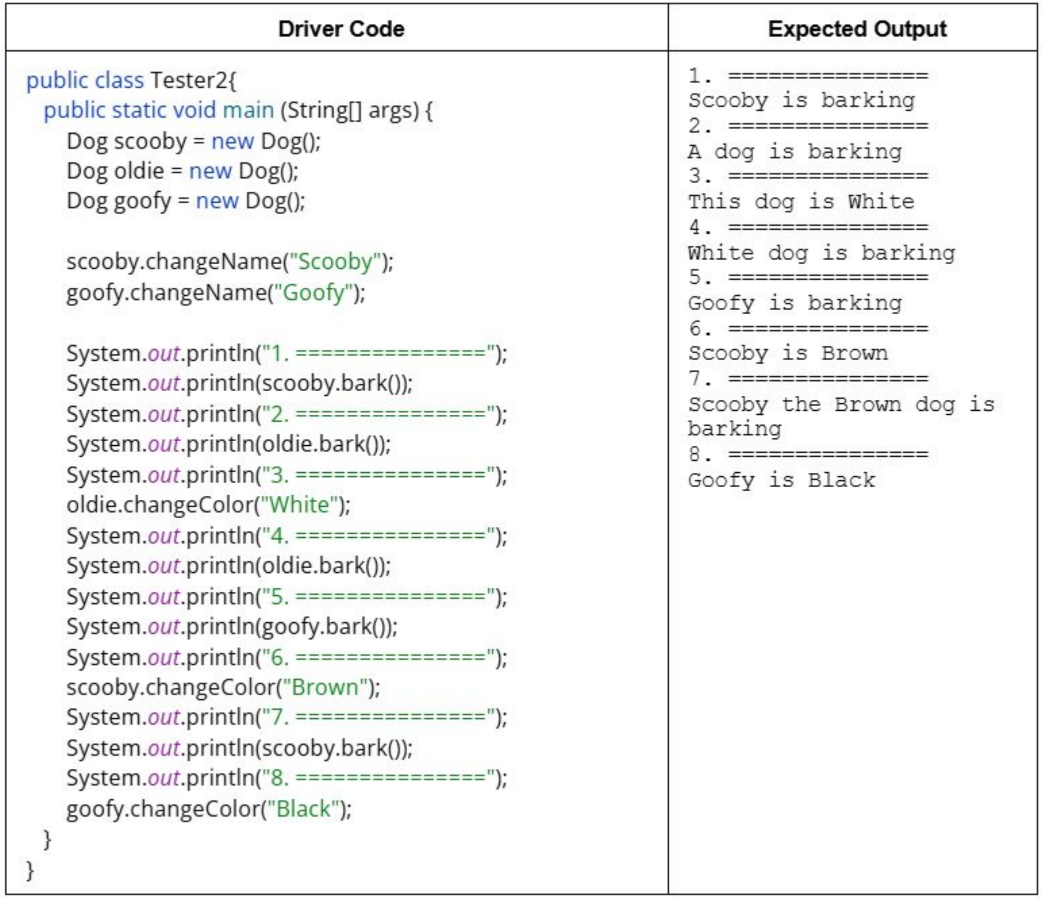
Driver Class

```
public class MobilePhoneTester{
  public static void main(String args []){
    MobilePhone m1 = new MobilePhone();
    MobilePhone m2 = new MobilePhone();
    m1.setContactCapacity(5);
    m2.setContactCapacity(100);
    m1.details();
    System.out.println("1----------------");
    m1.addContact("John", 9866);
    m1.addContact("Maria", 7865);
    System.out.println("2----------------");
    m1.details();
    System.out.println("3----------------");
    m1.makeCall(9866);
    System.out.println("4----------------");
    m1.addContact("Henry", 2365);
    System.out.println("5----------------");
    m1.makeCall(7552);
    m1.makeCall(2365);
    System.out.println("6----------------");
    m1.addContact("Gomes", 4589);
    m1.addContact("Antony", 8421);
    m1.addContact("Tony", 5789);
    System.out.println("7----------------");
    m1.details();
  }
}
```

Output

```
Total Contacts: 0
Contact List:
1----------------
The contact of John is added.
The contact of Maria is added.
2----------------
Total Contacts: 2
Contact List:
John:9866
Maria:7865
3----------------
Calling John . . .
4----------------
The contact of Henry is added.
5----------------
Calling 7552 . . .
Calling Henry . . .
6----------------
The contact of Gomes is added.
The contact of Antony is added.
Storage Full!!
7----------------
Total Contacts: 5
Contact List:
John:9866
Maria:7865
Henry:2365
Gomes:4589
Antony:8421
```

Create a **Dog** class so that the tester code generates the given output:



Driver Code

```java
public class Tester2{
  public static void main (String[] args) {
    Dog scooby = new Dog();
    Dog oldie = new Dog();
    Dog goofy = new Dog();

    scooby.changeName("Scooby");
    goofy.changeName("Goofy");

    System.out.println("1. ===============");
    System.out.println(scooby.bark());
    System.out.println("2. ===============");
    System.out.println(oldie.bark());
    System.out.println("3. ===============");
    oldie.changeColor("White");
    System.out.println("4. ===============");
    System.out.println(oldie.bark());
    System.out.println("5. ===============");
    System.out.println(goofy.bark());
    System.out.println("6. ===============");
    scooby.changeColor("Brown");
    System.out.println("7. ===============");
    System.out.println(scooby.bark());
    System.out.println("8. ===============");
    goofy.changeColor("Black");
  }
}
```

Expected Output

```
1. ===============
Scooby is barking
2. ===============
A dog is barking
3. ===============
This dog is White
4. ===============
White dog is barking
5. ===============
Goofy is barking
6. ===============
Scooby is Brown
7. ===============
Scooby the Brown dog is
barking
8. ===============
Goofy is Black
```

Task 4

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

```java
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 ==========
```

Driver Code

```
public class StrangerMagic {
  public static void main(String[] args){
     MagicItem char1 = new MagicItem();
     MagicItem char2 = new MagicItem();


     char1.newCharacter("Eleven");
     char2.newCharacter("Mike Wheeler");

     System.out.println("1 ==========");
     char1.displayInfo();
     System.out.println("2 ==========");
     char2.displayInfo();
     System.out.println("3 ==========");
     char1.findItem("Potion");
     char1.findItem("Elixir");
     char1.findItem("Elixir");
     char2.findItem("Potion");
     System.out.println("4 ==========");
     char1.findItem("Amulet");
     System.out.println("5 ==========");
     char1.displayInfo();
     System.out.println("6 ==========");
     char1.useItem("Potion");
     char1.useItem("Elixir");
     System.out.println("7 ==========");
     char1.displayInfo();
     System.out.println("8 ==========");
     char1.findItem("Amulet");
     System.out.println("9 ==========");
```

Output

```
1 ==========
Character Name: Eleven
Energy Level: 0
Item 1: null
Item 2: null
Item 3: null
2 ==========
Character Name: Mike Wheeler
Energy Level: 0
Item 1: null
Item 2: null
Item 3: null
3 ==========
Eleven found a Potion
Eleven found a Elixir
Eleven found a Elixir
Mike Wheeler found a Potion
4 ==========
All item slots occupied.
5 ==========
Character Name: Eleven
Energy Level: 0
Item 1: Potion
Item 2: Elixir
Item 3: Elixir
6 ==========
Eleven used a Potion
Energy Level after using item: 50
Eleven used a Elixir
Energy Level after using item: 150
7 ==========
Character Name: Eleven
Energy Level: 150
Item 1: null
Item 2: null
Item 3: Elixir
```

Task 5

reate a **MagicItem** class to provide the expected output. A character will have a name, ergy level, and three individual magic items (item1, item2, and item3).

- The name will be assigned inside the **newCharacter()** method. Whenever a new character is created, they will start with 0 energy and no magic items.
- Characters can find and use magic items, each with a specific energy boost. Magic items include "Potion" (+50), "Elixir" (+100), and "Amulet" (+200).
- Characters can use a magic item if they have it, which increases their energy level.

```
     char1.displayInfo();
     System.out.println("10 =========");
     char2.useItem("Amulet");
     System.out.println("11 =========");
     char2.displayInfo();
  }
}
```

```
8 ==========
Eleven found a Amulet
9 ==========
Character Name: Eleven
Energy Level: 150
Item 1: Amulet
Item 2: null
Item 3: Elixir
10 =========
Item not in inventory.
11 =========
Character Name: Mike Wheeler
Energy Level: 0
Item 1: Potion
Item 2: null
Item 3: null
```

Task 6

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

```
public class Tester6{
  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("Television", 5400.87);
    c1.addItem("Refrigerator", 5000);

    System.out.println("====2====");
    c2.addItem("Stove",439.90);

    System.out.println(""====3====");
    c3.addItem("Chair",1400.5);
    c3.addItem("Chair",3400);

    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();
  }
}
```

Expected Output

```
====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 7

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".
- A new array is created every time a reader's capacity is increased, which replaces the initial array.

Driver Code

```java
public class Reader_tester {
  public static void main(String[] args){
    Reader r1 = new Reader();
    Reader r2 = new Reader();

    r1.createReader("Messi", 2);
    r2.createReader("Ronaldo", 5);

    System.out.println("1 ==========");
    r1.readerInfo();

    System.out.println("2 ==========");
    r2.addBook("Java");
    r2.addBook("Python");
    r2.addBook("C++");
    r2.readerInfo();

    System.out.println("3 ==========");
    r1.addBook("C#");
    r1.addBook("Rust");
    r1.addBook("GoLang");

    System.out.println("4 ==========");
    r1.increaseCapacity(5);
    r1.addBook("Python");

    System.out.println("5 ==========");
    r1.readerInfo();
  }
}
```

Expected Output

```
1 ==========
Name: Messi
Capacity: 2
Books:
No books added yet
2 ==========
Name: Ronaldo
Capacity: 5
Books:
Book 1: Java
Book 2: Python
Book 3: C++
3 ==========
No more space for new book
4 ==========
Messi's capacity increased to 5
5 ==========
Name: Messi
Capacity: 5
Books:
Book 1: C#
Book 2: Rust
Book 3: Python
```

Task 8

You are building a ride booking app called UberApp. Using this app, a customer can book 3 rides.

- ***BookRide(Location, Distance)*** method books rides for a user and prints the fare for that ride based on the distance. After booking the ride, fare will be calculated as below:

  Fare = 30 * distance
- A person can change the location of their last booked ride using ***changeLocation(Location, Distance)*** method. The new fare is calculated as;

  Fare = 30 * distance + 20% of new Fare. i.g. If, new Fare = 210, then the total fare after changing location will be 210 + 210 * 0.2 = 252
- The UberApp keeps track of all the locations visited by the user in an array of String.
- The ***resetMonth()*** method resets the location visited in a month as well as the number of remaining rides of that month.

Design the **UberApp class** that will produce the following output.

Driver Code

```
public class AppTester {
  public static void main(String args[]){

    UberApp account1 = new UberApp();
    UberApp account2 = new UberApp();

    account1.createProfile("Jonas Kahnwald", 24,
"017111111111");
    account2.createProfile("Martha Nielsen", 28,
"018111111111");

    account1.showProfile();
    System.out.println("===== 1 ====");
    System.out.println("You have "+
account1.remainingRides() +" ride(s) remaining.");

    System.out.println("==== 2 ====");
    account2.showProfile();
    System.out.println("You have "+
account2.remainingRides() +" ride(s) remaining.");
```

Output

```
Hello! This is your Profile:
Full Name: Jonas Kahnwald
Age: 24
Phone Number: 017111111111
===== 1 ====
You have 3 ride(s) remaining.
==== 2 ====
Hello! This is your Profile:
Full Name: Martha Nielsen
Age: 28
Phone Number: 018111111111
You have 3 ride(s) remaining.
==== 3 ====
Jonas Kahnwald has booked a ride!
Destination: Merul Badda
Fare: 360.0 Taka
==== 4 ====
Jonas Kahnwald has booked a ride!
Destination: Dhanmondi 27
Fare: 129.0 Taka
Jonas Kahnwald has changed the
destination of his current ride to
Wari
New fare after adding 20% change
fees: 201.6 Taka.
```

```
    System.out.println("==== 3 ====");
    account1.bookRide("Merul Badda", 12.0);

    System.out.println("==== 4 ====");
    account1.bookRide("Dhanmondi 27", 4.3);
    account1.changeLocation("Wari", 5.6);

    System.out.println("==== 5 ====");
    account1.ridingHistory();

    System.out.println("==== 6 ====");
    account2.ridingHistory();

    System.out.println("==== 7 ====");
    account1.bookRide("Banani 11", 6.8);
    account1.bookRide("Gulshan 1", 2.1);

    System.out.println("==== 8 ====");
    account1.resetMonth();
    account1.bookRide("Gulshan 1", 2.1);
    account1.ridingHistory();
    System.out.println("You have "+
account1.remainingRides() +" ride(s) remaining.");
  }
}
```

```
==== 5 ====
Jonas Kahnwald, you have visited
Merul Badda, Wari this month.
==== 6 ====
Martha Nielsen, you haven't visited
anywhere this month.
==== 7 ====
Jonas Kahnwald has booked a ride!
Destination: Banani 11
Fare: 204.0 Taka
Jonas Kahnwald, please update your
plan to premium or wait till next
month!
==== 8 ====
Jonas Kahnwald has booked a ride!
Destination: Gulshan 1
Fare: 63.0 Taka
Jonas Kahnwald, you have visited
Gulshan 1 this month.
You have 2 ride(s) remaining.
```

Task 9

```java
public class Task09 {
    public int p = 3, y = 2, sum;
    public void methodA(){
        int x = 0, y = 0;
        y = y + this.y;
        x = sum + 2 + p;
        sum = x + y + methodB(p, y);
        System.out.println(x + " " + y+ " " + sum);
    }
    public int methodB(int p, int n){
        int x = 0;
        y = y + (++p);
        x = x + 2 + n;
        sum = sum + x + y;
        System.out.println(x + " " + y+ " " + sum);
        return sum;
    }
}
```

Driver code:

```java
public class Tester09 {
    public static void main(String [] args){
        Task09 t1 = new Task09 ();
        t1.methodA();
        t1.methodA();
    }
}
```

Outputs		
x	**y**	**Sum**

Task 10

```java
public class test1 {
   public int x = 3;
   public int y = 0;
   public int z = -1;
   public void case1(int x){
       int y = 12;
       this.x = y + 4 + x;
       y += this.y +1;
       case2(this.y, z);
       System.out.println(x + " "+ y + " "+ z);
       this.y = this.x + z;
       System.out.println(this.x + " "+ this.y + " "+ this.z);
   }
   public void case2(int temp, int z){
       this.x = z + temp + this.z;
       this.z = y + z;
       System.out.println(x + " "+ y + " "+ z);
       y  = x + y + z;
       temp = x;
       x = this.z;
       this.z = temp;
       System.out.println(this.x + " "+ this.y + " "+ this.z);
   }
}
```

Driver code:

```java
public class test1Driver {
   public static void main(String [] args){
       test1 t1 = new test1();
       t1.case1(4);
       t1.case2(5, 6);
   }
}
```

Outputs		

Task 11

```java
public class Task11 {
    public int temp = 4;
    public int sum;
    public int y;
    public int x;
    public void methodA(int m){
        int [] n = {2,5};
        int x = 0;
        y = y + m + this.methodB(x,m)+(temp++)+y;
        x = this.x + 2 + (++n[0]);
        sum = sum + x + y;
        n[0] = sum + 2;
       System.out.println(n[0] + x + " " + y+ " " + sum);
    }
    public int methodB(int m, int n){
        int [] y = {1};
        this.y = y[0] + this.y + m;
        x = this.y + 2 + temp - n;
        sum = x + y[0] + this.sum;
        System.out.println(y[0]+ x + " " + y[0] + " " +sum);
        return y[0];
    }
}
```

```java
public class Tester11 {
   public static void main(String [] args){
       Task11 t1 = new Task11();
       t1.methodA(5);
       t1.methodA(3);
   }
}
```

Outputs		