Assignment No. 4 & 5

Class, Object, Constructor & Constructor chaining

1. Create a class named 'Student' with String variable 'name' and integer variable 'roll_no'. Assign the value of roll_no as '2' and that of name as "John" by creating an object of the class Student.

```
class Student {
    String name;
    int roll_no;

    Student(String name, int roll_no) {
        this.name = name;
        this.roll_no = roll_no;
    }

    public void display() {
       System.out.println("Name: " + name + "\nRoll No.: " + roll_no);
    }
}

public class Question1 {
    public static void main(String[] args) {
        Student s = new Student("John", 2);
        s.display();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question1.java
PS D:\CDAC\Java\AssignmentNo1> java Question1
Name: John
Roll No.: 2
```

2. **Assign and print the roll number, phone number and address of two students having names "Sam" and "John" respectively by creating two objects of class 'Student'.**

```
class Student {
    String name;
    int roll_no;
    long phone_no;
    String address;

    Student(String name, int roll_no, long phone_no, String address)
{
        this.name = name;
        this.roll_no = roll_no;
```

```
        this.phone_no = phone_no;
        this.address = address;
    }

    public void display() {
        System.out.println(
                "Name: " + name + "\nRoll No.: " + roll_no + "\nPhone
Number: " + phone_no + "\nAddress: " + address);
    }
}

public class Question2 {
    public static void main(String[] args) {
        Student s1 = new Student("Sam", 131, 9988765422l, "Mumbai");
        s1.display();

       Student s2 = new Student("John", 152, 2269876542l, "Kolkata");
        s2.display();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question2.java
PS D:\CDAC\Java\AssignmentNo1> java Question2
Name: Sam
Roll No.: 131
Phone Number: 9988765422
Address: Mumbai
Name: John
Roll No.: 152
Phone Number: 2269876542
Address: Kolkata
```

3. Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.

```
class Triangle {
    int a;
    int b;
    int c;

    Triangle() {
        this.a = 3;
        this.b = 4;
        this.c = 5;
    }

    public int getPerimeter() {
```

```
        return (a + b + c);
    }

    public double getArea() {
        double semiperi = (a + b + c) / 2;
        return Math.sqrt(semiperi * (semiperi - a) * (semiperi - b) *
(semiperi - c));
    }
}

public class Question3 {
    public static void main(String[] args) {
        Triangle t = new Triangle();
        System.out.println("Area of Triangle: " + t.getArea());
               System.out.println("Perimeter   of   Triangle:   "   +
t.getPerimeter());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question3.java
PS D:\CDAC\Java\AssignmentNo1> java Question3
Area of Triangle: 6.0
Perimeter of Triangle: 12
```

4. Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with the constructor having the three sides as its parameters.

```
class Triangle {
    int a;
    int b;
    int c;

    Triangle(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public int getPerimeter() {
        return (a + b + c);
    }

    public double getArea() {
        double semiperi = (a + b + c) / 2;
        return Math.sqrt(semiperi * (semiperi - a) * (semiperi - b) *
(semiperi - c));
```

```
    }
}

public class Question4 {
    public static void main(String[] args) {
        Triangle t = new Triangle(3, 4, 5);
        System.out.println("Area of Triangle: " + t.getArea());
               System.out.println("Perimeter   of   Triangle:   "   +
t.getPerimeter());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question4.java
PS D:\CDAC\Java\AssignmentNo1> java Question4
Area of Triangle: 6.0
Perimeter of Triangle: 12
```

5. Write a program to print the area of two rectangles having sides (4,5) and (5,8) respectively by creating a class named 'Rectangle' with a method named 'Area' which returns the area and length and breadth passed as parameters to its constructor.

```
class Rectangle {
    int length;
    int breadth;

    Rectangle(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    public int getArea() {
        return (2 * length + 2 * breadth);
    }
}

public class Question5 {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(4, 5);
            System.out.println("Area   of   rectangle   (4,   5):   "   +
r1.getArea());

        Rectangle r2 = new Rectangle(5, 8);
            System.out.println("Area   of   rectangle   (5,   8):   "   +
r2.getArea());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question5.java
PS D:\CDAC\Java\AssignmentNo1> java Question5
Area of rectangle (4, 5): 18
Area of rectangle (5, 8): 26
```

6. Write a program to print the area of a rectangle by creating a class named 'Area' having two methods. First method named as 'setDim' takes length and breadth of the rectangle as parameters and the second method named as 'getArea' returns the area of the rectangle. Length and breadth of the rectangle are entered through the keyboard.

```
class Rectangle {
    int length;
    int breadth;

    public void setDim(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    public int getArea() {
        return (2 * length + 2 * breadth);
    }
}

public class Question6 {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle();
        r1.setDim(12, 30);
            System.out.println("Area of rectangle (12, 30): " +
r1.getArea());

        Rectangle r2 = new Rectangle();
        r2.setDim(17, 10);
            System.out.println("Area of rectangle (17, 10): " +
r2.getArea());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question6.java
PS D:\CDAC\Java\AssignmentNo1> java Question6
Area of rectangle (12, 30): 84
Area of rectangle (17, 10): 54
```

7. Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through the keyboard.

```java
import java.util.Scanner;

class Area {
    int length;
    int breadth;

    Area(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    public int returnArea() {
        return (2 * length + 2 * breadth);
    }
}

public class Question7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter length: ");
        int l = sc.nextInt();
        System.out.println("Enter breadth: ");
        int b = sc.nextInt();

        Area r1 = new Area(l, b);
        System.out.println("Area of rectangle: " + r1.returnArea());

        sc.close();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question7.java
PS D:\CDAC\Java\AssignmentNo1> java Question7
Enter length:
12
Enter breadth:
4
Area of rectangle: 32
```

8. Print the average of three numbers entered by the user by creating a class named 'Average' having a method to calculate and print the average.

```
import java.util.Scanner;

class Average {
    int a;
    int b;
    int c;

    Average(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public void printAverage() {
        double avg = (a + b + c) / 3;
        System.out.println("Average: " + avg);
    }
}

public class Question8 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number 1: ");
        int n1 = sc.nextInt();
        System.out.println("Enter number 2: ");
        int n2 = sc.nextInt();
        System.out.println("Enter number 3: ");
        int n3 = sc.nextInt();

        Average a = new Average(n1, n2, n3);
        a.printAverage();

        sc.close();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question8.java
PS D:\CDAC\Java\AssignmentNo1> java Question8
Enter number 1:
125
Enter number 2:
36
Enter number 3:
450
Average: 203.0
```

9. Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by the user.

```
import java.util.Scanner;

class Complex {
    int real;
    int imag;

    Complex(int real, int imag) {
        this.real = real;
        this.imag = imag;
    }

    public void printComplex() {
        System.out.println(real + "+" + imag + "i");
    }

    public static Complex sum(Complex a, Complex b) {
        return new Complex((a.real + b.real), (a.imag + b.imag));
    }

    public static Complex diff(Complex a, Complex b) {
        return new Complex((a.real - b.real), (a.imag - b.imag));
    }

    public static Complex product(Complex a, Complex b) {
        return new Complex(((a.real * b.real) - (a.imag * b.imag)),
((a.real * b.imag) + (a.imag * b.real)));
    }
}

public class Question9 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("First Complex Number:");
        System.out.print("Enter real term: ");
        int x = sc.nextInt();
        System.out.print("Enter imaginary term: ");
        int y = sc.nextInt();

        Complex c1 = new Complex(x, y);

        System.out.println("\nSecond Complex Number");
        System.out.print("Enter real term: ");
        int u = sc.nextInt();
        System.out.print("Enter imaginary term: ");
        int v = sc.nextInt();
```

```
        Complex c2 = new Complex(u, v);

        Complex add = Complex.sum(c1, c2);
        Complex sub = Complex.diff(c1, c2);
        Complex mul = Complex.product(c1, c2);

        System.out.println("\nSum of complex numbers: ");
        add.printComplex();
        System.out.println("Difference of complex numbers: ");
        sub.printComplex();
        System.out.println("Product of complex numbers: ");
        mul.printComplex();

        sc.close();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question9.java
PS D:\CDAC\Java\AssignmentNo1> java Question9
First Complex Number:
Enter real term: 12
Enter imaginary term: 45

Second Complex Number
Enter real term: 34
Enter imaginary term: 8

Sum of complex numbers:
46+53i
Difference of complex numbers:
-22+37i
Product of complex numbers:
48+1626i
```

10. Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee'. The output should be as follows: Name Year of joining Address Robert 1994 64C- WallsStreat Sam 2000 68D- WallsStreat John 1999 26B- WallsStreat

```
class Employee {
    String name;
    int year;
    String address;

    Employee(String name, int year, String address) {
        this.name = name;
```

```
        this.year = year;
        this.address = address;
    }

    public void display() {
       System.out.printf("%-12s %-20d %-20s\n", name, year, address);
    }
}

public class Question10 {
    public static void main(String[] args) {
       Employee e1 = new Employee("Robert", 1994, "64C-WallsStreat");
        Employee e2 = new Employee("Sam", 2000, "68C-WallsStreat");
        Employee e3 = new Employee("John", 1999, "26C-WallsStreat");

         System.out.printf("%-12s %-20s %-20s\n", "Name", "Year of
Joining", "Address");
        e1.display();
        e2.display();
        e3.display();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac Question10.java
PS D:\CDAC\Java\AssignmentNo1> java Question10
Name         Year of Joining      Address
Robert       1994                 64C-WallsStreat
Sam          2000                 68C-WallsStreat
John         1999                 26C-WallsStreat
```

Write a Java class Clock for dealing with the day time represented by hours, minutes, and seconds. Your class must have the following features:

- **Three instance variables for the hours (range 0 - 23), minutes (range 0 - 59), and seconds (range 0 - 59).**
- **Three constructors:**
  - **default (with no parameters passed; is should initialize the represented time to 12:0:0)**
  - **a constructor with three parameters: hours, minutes, and seconds.**
  - **a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes, and seconds)**
- **Instance methods:**

- **a set-method method setClock() with one parameter seconds since midnight (to be converted into the time value in hours, minutes, and seconds as above).**
- **get-methods getHours(), getMinutes(), getSeconds() with no parameters that return the corresponding values.**
- **set-methods setHours(), setMinutes(), setSeconds() with one parameter each that set up the corresponding instance variables.**
- **method tick() with no parameters that increments the time stored in a Clock object by one second.**
- **method addClock() accepting an object of type Clock as a parameter. The method should add the time represented by the parameter class to the time represented in the current class.**
- **Add an instance method toString() with no parameters to your class.**
- **toString() must return a String representation of the Clock object in the form "(hh:mm:ss)", for example "(03:02:34)".**
- **Add an instance method tickDown() which decrements the time stored in a Clock object by one second.**
- **Add an instance method subtractClock() that takes one Clock parameter and returns the difference between the time represented in the current Clock object and the one represented by the Clock parameter.**
- **Difference of time should be returned as a clock object.**

Write a separate class ClockDemo with a main() method.

- **The program should:**
  - **instantiate a Clock object firstClock using one integer seconds since midnight obtained from the keyboard.**
  - **tick the clock ten times by applying its tick() method and print out the time after each tick.**
  - **Extend your code by appending to it instructions instantiating a Clock object secondClock by using three integers (hours, minutes, seconds) read from the keyboard.**
  - **Then tick the clock ten times, printing the time after each tick.**
  - **Add the secondClock time in firstClock by calling the method addClock.**
  - **Print both clock objects calling toString method**

```
import java.util.Scanner;

class Clock {
    int hours;
    int minutes;
    int seconds;
```

```
    Clock() {
        this.hours = 12;
        this.minutes = 00;
        this.seconds = 00;
    }

    Clock(int hours, int minutes, int seconds) {
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
    }

    Clock(int seconds) {
        setClock(seconds);
    }

    public void setClock(int seconds) {
        this.hours = (seconds / 3600) % 24;
        this.minutes = (seconds / 60) % 60;
        this.seconds = seconds % 60;
    }

    public int getHours() {
        return hours;
    }

    public int getMinutes() {
        return minutes;
    }

    public int getSeconds() {
        return seconds;
    }

    public void setHours(int hours) {
        this.hours = hours;
    }

    public void setMinutes(int minutes) {
        this.minutes = minutes;
    }

    public void setSeconds(int seconds) {
        this.seconds = seconds;
    }

    public void tick() {
```

```
        seconds += 1;
        if (seconds == 60) {
            seconds = 0;
            minutes += 1;
        }
        if (minutes == 60) {
            minutes = 0;
            hours += 1;
        }
        if (hours == 24) {
            hours = 0;
        }
    }

    public void addClock(Clock c) {
        seconds += c.seconds;
        if (seconds >= 60) {
            seconds -= 60;
            minutes += 1;
        }
        minutes += c.minutes;
        if (minutes >= 60) {
            minutes -= 60;
            hours += 1;
        }
        hours += c.hours;
        if (hours >= 24) {
            hours -= 24;
        }
    }

    public String toString() {
        return "(" + hours + " : " + minutes + " : " + seconds +
")";
    }

    public void tickDown() {
        seconds -= 1;
        if (seconds < 0) {
            seconds = 59;
            minutes -= 1;
        }
        if (minutes < 0) {
            minutes = 59;
            hours -= 1;
        }
        if (hours < 0) {
            hours = 23;
```

```
        }
    }

    public void subtractClock(Clock c) {
        seconds -= c.seconds;
        if (seconds < 0) {
            seconds += 60;
            minutes -= 1;
        }
        minutes -= c.minutes;
        if (minutes < 0) {
            minutes += 60;
            hours -= 1;
        }
        hours -= c.hours;
        if (hours < 0) {
            hours += 24;
        }
    }

    public Clock diffClock(Clock c2) {
        int currentTotal = hours * 3600 + minutes * 60 + seconds;
        int newTotal = c2.hours * 3600 + c2.minutes * 60 +
c2.seconds;

        int diff = currentTotal - newTotal;

        if (diff < 0) {
            diff += 24 * 3600;
        }

        Clock result = new Clock();
        result.setClock(diff);
        return result;
    }
}

public class ClockDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter time in seconds: ");
        int s1 = sc.nextInt();
        Clock firstClock = new Clock(s1);
        Clock originalFirstClock = new Clock(firstClock.getHours(),
firstClock.getMinutes(), firstClock.getSeconds());

        System.out.println("First Clock Time: " +
firstClock.toString());
```

```
        for (int i = 1; i <= 10; i++) {
            firstClock.tick();
            System.out.println("After tick " + i + ": " +
firstClock.toString());
        }

        System.out.println();

        System.out.println("Enter hours: ");
        int h = sc.nextInt();
        System.out.println("Enter minutes: ");
        int m = sc.nextInt();
        System.out.println("Enter seconds: ");
        int s = sc.nextInt();
        Clock secondClock = new Clock(h, m, s);
        Clock originalSecondClock = new
Clock(secondClock.getHours(), secondClock.getMinutes(),
                secondClock.getSeconds());
        System.out.println("\nSecond Clock Time: " +
secondClock.toString());

        for (int j = 1; j <= 10; j++) {
            secondClock.tick();
            System.out.println("After tick " + j + ": " +
secondClock.toString());
        }

        firstClock.addClock(secondClock);
        System.out.println("\nAdding firstClock and secondClock: " +
firstClock.toString());

        System.out.println("\n\nFirst Clock Time: " +
originalFirstClock.toString());
        System.out.println("Second Clock Time: " +
originalSecondClock.toString());

        Clock thirdClock =
originalFirstClock.diffClock(originalSecondClock);
        System.out.println("\nDifference between firstClock and
secondClock: " + thirdClock.toString());

        sc.close();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ClockDemo.java
PS D:\CDAC\Java\AssignmentNo1> java ClockDemo
Enter time in seconds:
34529
First Clock Time: (9 : 35 : 29)
After tick 1: (9 : 35 : 30)
After tick 2: (9 : 35 : 31)
After tick 3: (9 : 35 : 32)
After tick 4: (9 : 35 : 33)
After tick 5: (9 : 35 : 34)
After tick 6: (9 : 35 : 35)
After tick 7: (9 : 35 : 36)
After tick 8: (9 : 35 : 37)
After tick 9: (9 : 35 : 38)
After tick 10: (9 : 35 : 39)

Enter hours:
6
Enter minutes:
33
Enter seconds:
24
```

```
Second Clock Time: (6 : 33 : 24)
After tick 1: (6 : 33 : 25)
After tick 2: (6 : 33 : 26)
After tick 3: (6 : 33 : 27)
After tick 4: (6 : 33 : 28)
After tick 5: (6 : 33 : 29)
After tick 6: (6 : 33 : 30)
After tick 7: (6 : 33 : 31)
After tick 8: (6 : 33 : 32)
After tick 9: (6 : 33 : 33)
After tick 10: (6 : 33 : 34)

Adding firstClock and secondClock: (16 : 9 : 13)


First Clock Time: (9 : 35 : 29)
Second Clock Time: (6 : 33 : 24)

Difference between firstClock and secondClock: (3 : 2 : 5)
```

Question 3

Write a Java class Complex for dealing with complex numbers. Your class must have the following features:

- **Instance variables :**
  realPart for the real part of type double

imaginaryPart for imaginary part of type double.

- **Constructor:**
  public Complex (): A default constructor, it should initialize the number to 0, 0)
  public Complex (double realPart, double imaginaryPart): A constructor with parameters, it creates the complex object by setting the two fields to the passed values.
- **Instance methods:**
  public Complex add (Complex otherNumber): This method will find the sum of the current complex number and the passed complex number. The method returns a new Complex number which is the sum of the two.
  public Complex subtract (Complex otherNumber): This method will find the difference of the current complex number and the passed complex number. The method returns a new Complex number which is the difference of the two.
  public Complex multiply (Complex otherNumber): This method will find the product of the current complex number and the passed complex number. The method returns a new Complex number which is the product of the two.
  public Complex divide (Complex otherNumber): This method will find the ... of the current complex number and the passed complex number. The method returns a new Complex number which is the ... of the two.
  public void setRealPart (double realPart): Used to set the real part of this complex number.
  public void setImaginaryPart (double realPart): Used to set the imaginary part of this complex number.
  public double getRealPart(): This method returns the real part of the complex number
  public double getImaginaryPart(): This method returns the imaginary part of the complex number
  public String toString(): This method allows the complex number to be easily printed out to the screen

Write a separate class ComplexDemo with a main() method and test the Complex class methods.

```
import java.util.Scanner;

class Complex {
    double realPart;
    double imaginaryPart;

    public Complex() {
        this.realPart = 0;
        this.imaginaryPart = 0;
    }
```

```
    public Complex(double realPart, double imaginaryPart) {
        this.realPart = realPart;
        this.imaginaryPart = imaginaryPart;
    }

    public void setRealPart(double realPart) {
        this.realPart = realPart;
    }

    public void setImaginaryPart(double imaginaryPart) {
        this.imaginaryPart = imaginaryPart;
    }

    public double getRealPart() {
        return realPart;
    }

    public double getImaginaryPart() {
        return imaginaryPart;
    }

    public static Complex add(Complex a, Complex b) {
        return new Complex((a.getRealPart() + b.getRealPart()),
(a.getImaginaryPart() + b.getImaginaryPart()));
    }

    public static Complex subtract(Complex a, Complex b) {
        return new Complex((a.getRealPart() - b.getRealPart()),
(a.getImaginaryPart() - b.getImaginaryPart()));
    }

    public static Complex multiply(Complex a, Complex b) {
        return new Complex(((a.getRealPart() * b.getRealPart()) -
(a.getImaginaryPart() * b.getImaginaryPart())),
                ((a.getRealPart() * b.getImaginaryPart()) +
(a.getImaginaryPart() * b.getRealPart())));
    }

    public static Complex divide(Complex a, Complex b) {
        return new Complex(
                ((a.getRealPart() * b.getRealPart()) +
(a.getImaginaryPart() * b.getImaginaryPart()))
                        / ((b.getRealPart() * b.getRealPart()) +
(b.getImaginaryPart() * b.getImaginaryPart())),
                ((a.getImaginaryPart() * b.getRealPart()) -
(a.getRealPart() * b.getImaginaryPart()))
                        / ((b.getRealPart() * b.getRealPart()) +
(b.getImaginaryPart() * b.getImaginaryPart())));
```

```
    }

    public String toString() {
        return realPart + " + " + imaginaryPart + "i";
    }
}

public class ComplexDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("First Complex Number:");
        System.out.print("Enter real term: ");
        int x = sc.nextInt();
        System.out.print("Enter imaginary term: ");
        int y = sc.nextInt();

        Complex c1 = new Complex(x, y);
        System.out.println("First Complex: " + c1.toString());

        System.out.println("\nSecond Complex Number");
        System.out.print("Enter real term: ");
        int u = sc.nextInt();
        System.out.print("Enter imaginary term: ");
        int v = sc.nextInt();

        Complex c2 = new Complex(u, v);
        System.out.println("Second Complex: " + c2.toString());

        Complex add = Complex.add(c1, c2);
        Complex sub = Complex.subtract(c1, c2);
        Complex mul = Complex.multiply(c1, c2);
        Complex div = Complex.divide(c1, c2);

        System.out.println("\nSum of complex numbers: " +
add.toString());
        System.out.println("\nDifference of complex numbers: " +
sub.toString());
        System.out.println("\nProduct of complex numbers: " +
mul.toString());
        System.out.println("\nDivision of complex numbers: " +
div.toString());

        sc.close();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ComplexDemo.java
PS D:\CDAC\Java\AssignmentNo1> java ComplexDemo
First Complex Number:
Enter real term: 12
Enter imaginary term: 5
First Complex: 12.0 + 5.0i

Second Complex Number
Enter real term: 5
Enter imaginary term: 2
Second Complex: 5.0 + 2.0i

Sum of complex numbers: 17.0 + 7.0i

Difference of complex numbers: 7.0 + 3.0i

Product of complex numbers: 50.0 + 49.0i

Division of complex numbers: 2.413793103448276 + 0.0344827586206896551i
```

Question 4

Write a Java class Author with following features:

- **Instance variables :**
  firstName for the author's first name of type String.
  lastName for the author's last name of type String.
- **Constructor:**
  public Author (String firstName, String lastName): A constructor with parameters, it creates the Author object by setting the two fields to the passed values.
- **Instance methods:**
  public void setFirstName (String firstName): Used to set the first name of the author.
  public void setLastName (String lastName): Used to set the last name of the author.
  public double getFirstName(): This method returns the first name of the author.
  public double getLastName(): This method returns the last name of the author.
  public String toString(): This method printed out author's name to the screen

Write a Java class Book with following features:

- **Instance variables :**
  title for the title of book of type String.
  author for the author's name of type String.
  price for the book price of type double.
- **Constructor:**
  public Book (String title, Author name, double price): A constructor with parameters, it creates the Author object by setting the fields to the passed values.
- **Instance methods:**
  public void setTitle(String title): Used to set the title of a book.

public void setAuthor(String author): Used to set the name of the author of a book.
public void setPrice(double price): Used to set the price of a book.
public double getTitle(): This method returns the title of the book.
public double getAuthor(): This method returns the author's name of the book.
public String toString(): This method printed out book's details to the screen

Write a separate class BookDemo with a main() method creates a Book titled "Developing Java Software" with authors Russel Winderand price 79.75. Prints the Book's string representation to standard output (using System.out.println).

```
class Author {
    String firstName;
    String lastName;

    public Author(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String toString() {
        return firstName + " " + lastName;
    }
}

class Book {
    String title;
    Author author;
    double price;

    public Book(String title, Author author, double price) {
        this.title = title;
```

```
        this.author = author;
        this.price = price;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setAuthor(Author author) {
        this.author = author;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getTitle() {
        return title;
    }

    public Author getAuthor() {
        return author;
    }

    public String toString() {
        return "Title: " + title + "\nAuthor: " + author + "\nPrice:
" + price;
    }
}

public class BookDemo {
    public static void main(String[] args) {
        Author a = new Author("Fyodor", "Dostoevsky");
        Book b = new Book("Crime and Punishment", a, 418);
        System.out.println(b.toString());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac BookDemo.java
PS D:\CDAC\Java\AssignmentNo1> java BookDemo
Title: Crime and Punishment
Author: Fyodor Dostoevsky
Price: 418.0
```

1. Write a program by creating an 'Employee' class having the following methods and print the final salary.

1. **'getInfo()' which takes the salary, number of hours of work per day of employee as parameter**
2. **'AddSal()' which adds $10 to the salary of the employee if it is less than $500.**
3. **'AddWork()' which adds $5 to the salary of an employee if the number of hours of work per day is more than 6 hours.**

```
class Employee {
    int salary;
    int noOfHours;

    public void getInfo(int salary, int noOfHours) {
        this.salary = salary;
        this.noOfHours = noOfHours;
    }

    public int addSal() {
        if (salary < 500) {
            salary += 10;
        }
        return salary;
    }

    public int addWork() {
        if (noOfHours > 6) {
            salary += 5;
        }
        return salary;
    }

    public String toString() {
        return "Salary: " + salary + ", Number of hours of work per
day: " + noOfHours;
    }
}

public class EmployeeSalary {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        e1.getInfo(450, 8);
        System.out.println("Before: " + e1.toString());
        e1.addSal();
        e1.addWork();
        System.out.println("After: " + e1.toString());
```

```
        System.out.println();

        Employee e2 = new Employee();
        e2.getInfo(300, 5);
        System.out.println("Before: " + e2.toString());
        e2.addSal();
        e2.addWork();
        System.out.println("After: " + e2.toString());

        System.out.println();

        Employee e3 = new Employee();
        e3.getInfo(600, 10);
        System.out.println("Before: " + e3.toString());
        e3.addSal();
        e3.addWork();
        System.out.println("After: " + e3.toString());
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac EmployeeSalary.java
PS D:\CDAC\Java\AssignmentNo1> java EmployeeSalary
Before: Salary: 450, Number of hours of work per day: 8
After: Salary: 465, Number of hours of work per day: 8

Before: Salary: 300, Number of hours of work per day: 5
After: Salary: 310, Number of hours of work per day: 5

Before: Salary: 600, Number of hours of work per day: 10
After: Salary: 605, Number of hours of work per day: 10
```

2. Create a class called 'Matrix' containing a constructor that initializes the number of rows and number of columns of a new Matrix object. The Matrix class has the following information:

1 - number of rows of matrix
2 - number of columns of matrix
3 - elements of matrix in the form of 2D array

3. The Matrix class has methods for each of the following:

1 - get the number of rows
2 - get the number of columns
3 - set the elements of the matrix at given position (i,j)
4 - adding two matrices. If the matrices are not addable, "Matrices cannot be added" will be displayed.
5 - multiplying the two matrices

```
class Matrix {
    int row;
    int col;
    int[][] elements;

    Matrix(int row, int col) {
        this.row = row;
        this.col = col;
        this.elements = new int[row][col];
    }

    public int getRow() {
        return row;
    }

    public int getColumn() {
        return col;
    }

    public void setElements(int row, int col, int value) {
        elements[row][col] = value;
    }

    public int getElements(int row, int col) {
        return elements[row][col];
    }

    public static Matrix addMatrix(Matrix a, Matrix b) {
        if (a.row != b.row || a.col != b.col) {
            System.out.println("Matrix cannot be added");
            return null;
        }

        Matrix add = new Matrix(a.row, a.col);
        for (int i = 0; i < a.row; i++) {
            for (int j = 0; j < a.col; j++) {
                add.elements[i][j] = a.getElements(i, j) +
b.getElements(i, j);
            }
        }
        return add;
    }

    public static Matrix mulMatrix(Matrix a, Matrix b) {
        if (a.col != b.row) {
            System.out.println("Matrix cannot be multiplied");
            return null;
```

```
        }

        Matrix mul = new Matrix(a.row, b.col);
        for (int i = 0; i < a.row; i++) {
            for (int j = 0; j < b.col; j++) {
                for (int k = 0; k < a.col; k++) {
                    mul.elements[i][j] += a.getElements(i, k) *
b.getElements(k, j);
                }
            }
        }
        return mul;
    }

    public void printMatrix() {
        for (int i = 0; i < row; i++) {
            System.out.print("[");
            for (int j = 0; j < col; j++) {
                System.out.print(elements[i][j] + " ");
            }
            System.out.print("]");
            System.out.println();
        }
    }
}

public class MatrixDemo {
    public static void main(String[] args) {
        Matrix m1 = new Matrix(2, 2);
        m1.setElements(0, 0, 1);
        m1.setElements(0, 1, 2);
        m1.setElements(1, 0, 3);
        m1.setElements(1, 1, 4);

        Matrix m2 = new Matrix(2, 2);
        m2.setElements(0, 0, 5);
        m2.setElements(0, 1, 6);
        m2.setElements(1, 0, 7);
        m2.setElements(1, 1, 8);

        System.out.println("Addition of Matrix: ");
        Matrix addRes = Matrix.addMatrix(m1, m2);
        if (addRes != null) {
            addRes.printMatrix();
        }

        System.out.println("\nMultiplication of Matrix: ");
        Matrix mulRes = Matrix.mulMatrix(m1, m2);
```

```
        if (mulRes != null) {
            mulRes.printMatrix();
        }
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac MatrixDemo.java
PS D:\CDAC\Java\AssignmentNo1> java MatrixDemo
Addition of Matrix:
[6 8 ]
[10 12 ]

Multiplication of Matrix:
[19 22 ]
[43 50 ]
```

1. Constructor Chaining within the Same Class

- **Create a class Car with multiple constructors that initialize different attributes using constructor chaining.**
- **Problem Statement:**
  Create a class Car with attributes brand, model, and price.
  Implement constructor chaining within the same class: One constructor should only take the brand. Another constructor should take brand and model. The final constructor should take brand, model, and price.
  Use the this() keyword to call other constructors.
  Display car details in each constructor.
- **Task: Create objects using different constructors and observe constructor chaining in action.**

```
class Car {
    String brand;
    String model;
    int price;

    Car(String brand) {
        this.brand = brand;
        System.out.println("Brand: " + brand);
    }

    Car(String brand, String model) {
        this(brand);
        this.model = model;
        System.out.println("Model: " + model);
    }
```

```
    Car(String brand, String model, int price) {
        this(brand, model);
        this.price = price;
        System.out.println("Price: " + price);
    }

}

public class ConstructorChaining1 {
    public static void main(String[] args) {
        Car c1 = new Car("Toyota");
        System.out.println();
        Car c2 = new Car("Mahindra", "Thar");
        System.out.println();
        Car c3 = new Car("Hyundai", "Creta", 1111000);
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ConstructorChaining1.java
PS D:\CDAC\Java\AssignmentNo1> java ConstructorChaining1
Brand: Toyota

Brand: Mahindra
Model: Thar

Brand: Hyundai
Model: Creta
Price: 1111000
```

2. Constructor Chaining Using super Keyword (Parent-Child Relationship)

- **Create a class hierarchy where the child class calls the parent class constructor using super().**
- **Problem Statement:**
  Create a Person class with attributes name and age.
  Create a Student class that extends Person and has an additional attribute course.
  Use constructor chaining: Person class should have a constructor initializing name and age. Student class should use super(name, age) to call the Person constructor and then initialize course.
  Display details in both constructors.
- **Task: Create a Student object and verify that both constructors (parent and child) are executed in sequence.**

```java
class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
        System.out.println("Name: " + name + "\nAge: " + age);
    }
}

class Student extends Person {
    String course;

    Student(String name, int age, String course) {
        super(name, age);
        this.course = course;
        System.out.println("Course: " + course);
    }
}

public class ConstructorChaining2 {
    public static void main(String[] args) {
        Student s = new Student("Dante", 25, "MBA");
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ConstructorChaining2.java
PS D:\CDAC\Java\AssignmentNo1> java ConstructorChaining2
Name: Dante
Age: 25
Course: MBA
```

3. Multi-Level Constructor Chaining (Grandparent → Parent → Child)

- **Demonstrate constructor chaining in a multi-level inheritance scenario.**
- **Problem Statement:**
  Create a Vehicle class with an attribute type.
  Create a subclass FourWheeler with an additional attribute brand.
  Create another subclass Car with attributes model and price.
  Use multi-level constructor chaining: Vehicle initializes type. FourWheeler calls super(type) and initializes brand. Car calls super(type, brand), initializes model, and price.
  Display details at each level.
- **Task: Create a Car object and verify that constructors are executed from parent → child → grandchild.**

```
class Vehicle {
    String type;

    Vehicle(String type) {
        this.type = type;
        System.out.println("Type: " + type);
    }
}

class FourWheeler extends Vehicle {
    String brand;

    FourWheeler(String type, String brand) {
        super(type);
        this.brand = brand;
        System.out.println("Brand: " + brand);
    }
}

class Car extends FourWheeler {
    String model;
    int price;

    Car(String type, String brand, String model, int price) {
        super(type, brand);
        this.model = model;
        this.price = price;
        System.out.println("Model: " + model + "\nPrice: " + price);
    }
}

public class ConstructorChaining3 {
    public static void main(String[] args) {
        Vehicle v = new Vehicle("Bus");
        System.out.println();
        FourWheeler f = new FourWheeler("Four Wheeler", "Maruti
Suzuki");
        System.out.println();
        Car c = new Car("Car", "Hyundai", "Creta", 1111000);
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ConstructorChaining3.java
PS D:\CDAC\Java\AssignmentNo1> java ConstructorChaining3
Type: Bus

Type: Four Wheeler
Brand: Maruti Suzuki

Type: Car
Brand: Hyundai
Model: Creta
Price: 1111000
```

4. Constructor Chaining in an E-Commerce Scenario

- **Create an Order class where different constructors initialize order details using constructor chaining.**
- **Problem Statement:**
  Create an Order class with attributes orderId, customerName, and totalAmount. Implement constructor chaining within the same class: One constructor initializes only orderId. Another constructor initializes orderId and customerName by calling the first constructor. The final constructor initializes all three attributes (orderId, customerName, totalAmount) by calling the second constructor.
  Display order details.
- **Task: Create Order objects using different constructors and verify how chaining works.**

```
class Order {
    int orderId;
    String customerName;
    double totalAmount;

    Order(int orderId) {
        this.orderId = orderId;
        System.out.println("Order ID: " + orderId);
    }

    Order(int orderId, String customerName) {
        this(orderId);
        this.customerName = customerName;
        System.out.println("Customer Name: " + customerName);
    }

    Order(int orderId, String customerName, double totalAmount) {
        this(orderId, customerName);
        this.totalAmount = totalAmount;
        System.out.println("Total Amount: " + totalAmount);
```

```
    }
}

public class ConstructorChaining4 {
    public static void main(String[] args) {
        Order o1 = new Order(25364794);
        System.out.println();
        Order o2 = new Order(11456883, "Dante");
        System.out.println();
        Order o3 = new Order(65993314, "Aileen", 352500);
        System.out.println();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ConstructorChaining4.java
PS D:\CDAC\Java\AssignmentNo1> java ConstructorChaining4
Order ID: 25364794

Order ID: 11456883
Customer Name: Dante

Order ID: 65993314
Customer Name: Aileen
Total Amount: 352500.0
```

5. Constructor Chaining in a Banking System

- **Create a BankAccount class where constructor chaining initializes different account types.**
- **Problem Statement:**
  Create a BankAccount class with attributes accountNumber, holderName, and balance.
  Implement constructor chaining: One constructor initializes only accountNumber. Another constructor initializes accountNumber and holderName, calling the first constructor. The final constructor initializes all three attributes by calling the second constructor.
  Implement a method to display account details.
- **Task: Create BankAccount objects using different constructors and verify the constructor call sequence.**

```
class BankAccount {
    int accountNumber;
    String holderName;
    double balance;
```

```
    BankAccount(int accountNumber) {
        this.accountNumber = accountNumber;
    }

    BankAccount(int accountNumber, String holderName) {
        this(accountNumber);
        this.holderName = holderName;
    }

    BankAccount(int accountNumber, String holderName, double
balance) {
        this(accountNumber, holderName);
        this.balance = balance;
    }

    public void displayAccountDetails() {
        System.out.println("Account Number: " + accountNumber);
        if (holderName != null) {
            System.out.println("Holder Name: " + holderName);
        } else {
            System.out.println("Holder Name: N/A");
        }
        if (balance != 0) {
            System.out.println("Balance: " + balance);
        } else {
            System.out.println("Balance: N/A");
        }
    }
}

public class ConstructorChaining5 {
    public static void main(String[] args) {
        BankAccount b1 = new BankAccount(252200316);
        b1.displayAccountDetails();
        System.out.println();
        BankAccount b2 = new BankAccount(252200254, "Dante");
        b2.displayAccountDetails();
        System.out.println();
        BankAccount b3 = new BankAccount(252200070, "Aileen",
3502500);
        b3.displayAccountDetails();
        System.out.println();
    }
}
```

```
PS D:\CDAC\Java\AssignmentNo1> javac ConstructorChaining5.java
PS D:\CDAC\Java\AssignmentNo1> java ConstructorChaining5
Account Number: 252200316
Holder Name: N/A
Balance: N/A

Account Number: 252200254
Holder Name: Dante
Balance: N/A

Account Number: 252200070
Holder Name: Aileen
Balance: 3502500.0
```