JAVA PRACTICAL FILE

Object Oriented Programming and Java (MCA-167)

Submitted by:
Name: Abhishek Tyagi
Roll no.: 03311104422
Sem: MCA 1st

Submitted to:
Name: Mr Meetender
Designation: Asst. Professor

Banarsidas Chandiwala Institute of Information Technology

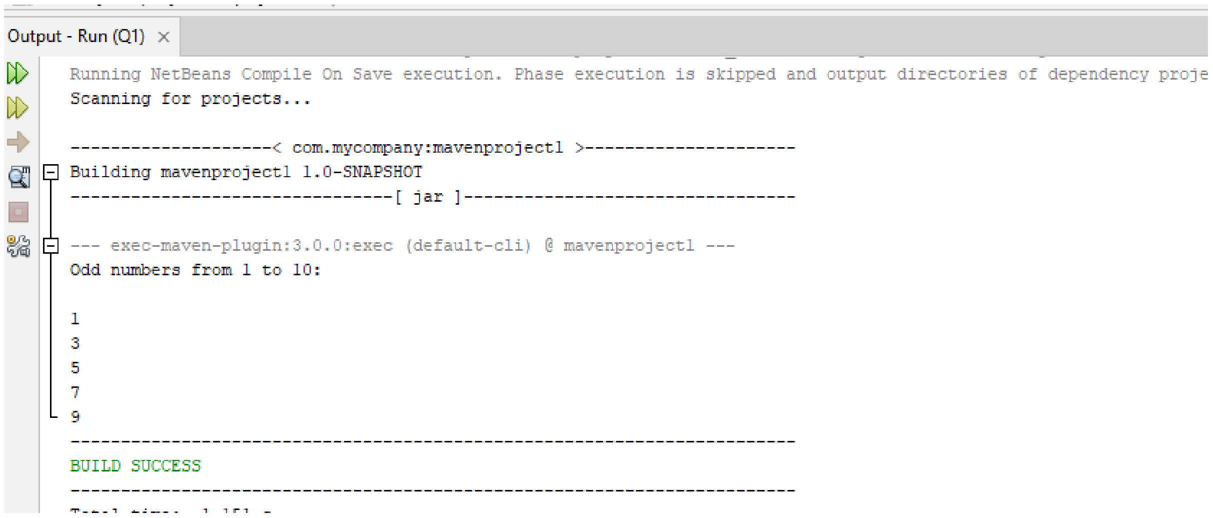
Index

1. Write a Java program to print all odd numbers between 1 to 10.

CODE:

```
import static java.lang.System.out;

    public class Q1 {

      public static void main(String args[]){

         int i=1;
         System.out.println("Odd numbers from 1 to 10:\n");
         for(i=1;i<=10;i+=2)
         System.out.println(i);
      }
    }
```

OUTPUT:



Output - Run (Q1)

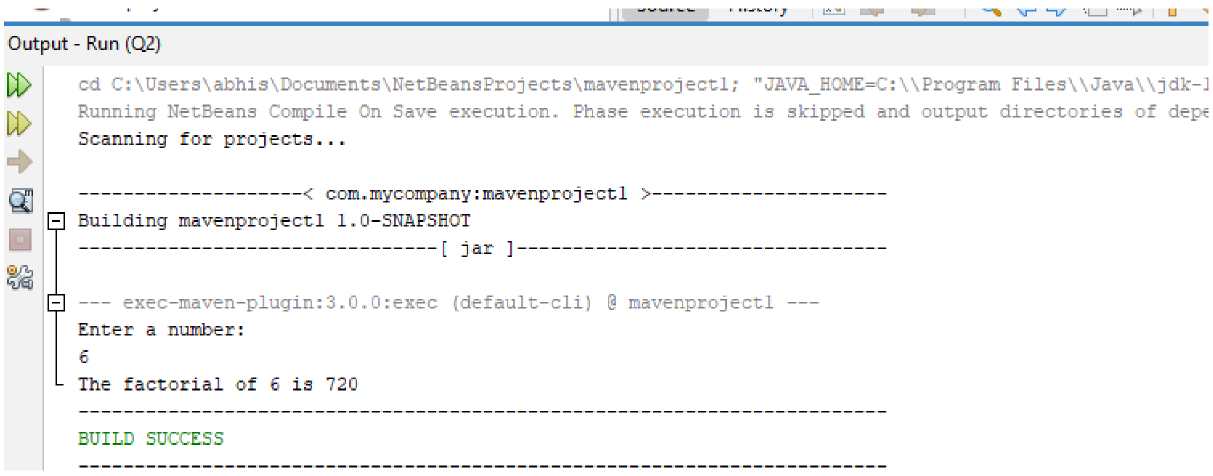
```
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency proje
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Odd numbers from 1 to 10:

1
3
5
7
9
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

2. Write a Java program to find out factorial of a number through recursion.

CODE:

```
import java.util.Scanner;


public class Q2 {
static int factorial(int n) {
if (n == 0)
return 1;
else
return (n * factorial(n - 1));
}


public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter a number: ");
int num = sc.nextInt();
int result = factorial(num);
System.out.println("The factorial of " + num + " is " + result);
}
}
```

OUTPUT:



Output - Run (Q2)

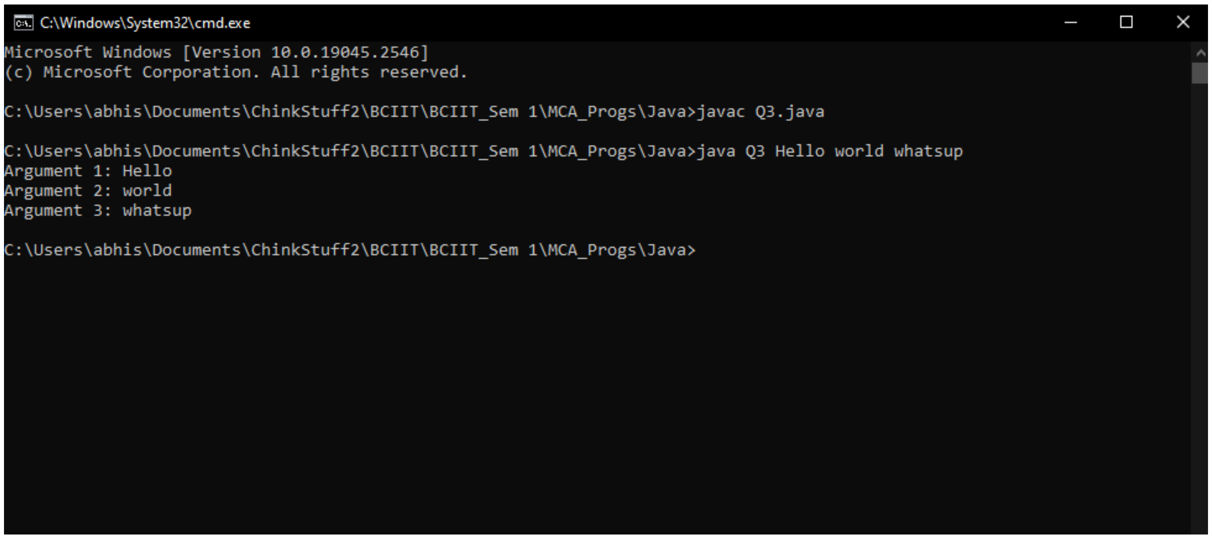
```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\jdk-1
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of depe
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Enter a number:
6
The factorial of 6 is 720
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

3. Write a Java program to accept command line arguments & print them.

CODE:

```
import static java.lang.System.out;


public class Q3 {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++) {
      System.out.println("Argument " + (i + 1) + ": " + args[i]);
    }
  }
}
```

OUTPUT:



C:\Windows\System32\cmd.exe

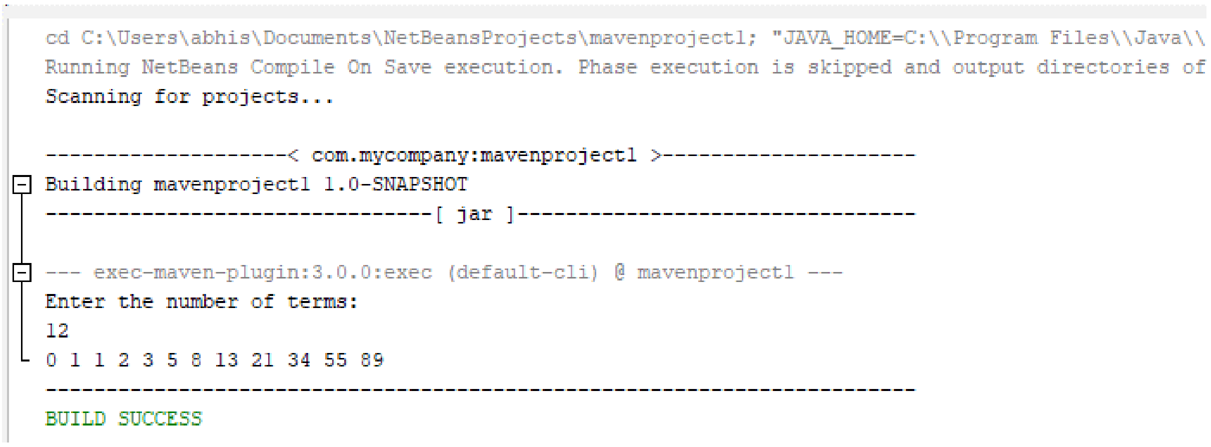
```
Microsoft Windows [Version 10.0.19045.2546]
(c) Microsoft Corporation. All rights reserved.

C:\Users\abhis\Documents\ChinkStuff2\BCIIT\BCIIT_Sem 1\MCA_Progs\Java>javac Q3.java

C:\Users\abhis\Documents\ChinkStuff2\BCIIT\BCIIT_Sem 1\MCA_Progs\Java>java Q3 Hello world whatsup
Argument 1: Hello
Argument 2: world
Argument 3: whatsup

C:\Users\abhis\Documents\ChinkStuff2\BCIIT\BCIIT_Sem 1\MCA_Progs\Java>
```

4. Write a Java program to print fibonacci series.

CODE:

```
import java.util.Scanner;
public class Q4 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number of terms: ");
    int n = sc.nextInt();

    int a = 0, b = 1, c;
    System.out.print(a + " " + b);

    for (int i = 2; i < n; i++) {
      c = a + b;
      System.out.print(" " + c);
      a = b;
      b = c;
    }
  }
}
```

OUTPUT:



```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Enter the number of terms:
12
0 1 1 2 3 5 8 13 21 34 55 89
------------------------------------------------------------------------
BUILD SUCCESS
```

5. Write a Java program that creates a class accounts with following details:

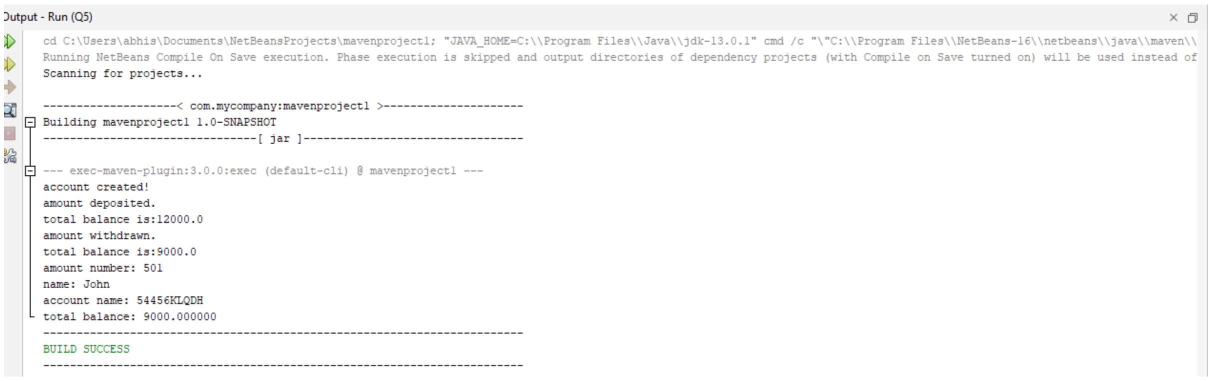
CODE:

Instance variables: ac_no., name, ac_name, balance

Methods: withdrawal(), deposit(), display().use constructors to initialize members.

```
import static java.lang.System.out;

class Accounts {
   private int ac_no;
   private String name, ac_name;
   private double balance;

   public Accounts(int n, String nm, String ac_nm, double bal) {
      ac_no = n;
      name = nm;
      ac_name = ac_nm;
      balance = bal;
      out.println("account created!");
   }

   public void deposit(double amount) {
      balance += amount;
      out.println("amount deposited.\ntotal balance is:" + balance);
   }

   public void withdrawal(double amount) {
        balance -= amount;
        out.println("amount withdrawn.\ntotal balance is:" + balance);
    }

    public void display() {
        out.printf("amount number: %d\nname: %s\naccount name: %s\ntotal balance: %f\n",
            ac_no, name, ac_name, balance);
    }
}

class Q5 {
    public static void main(String[] args) {
        Accounts ac = new Accounts(501, "John", "54456KLQDH", 10000.0);
        ac.deposit(2000.0);
        ac.withdrawal(3000.0);
        ac.display();
    }
}
```

OUTPUT:



Output - Run (Q5)

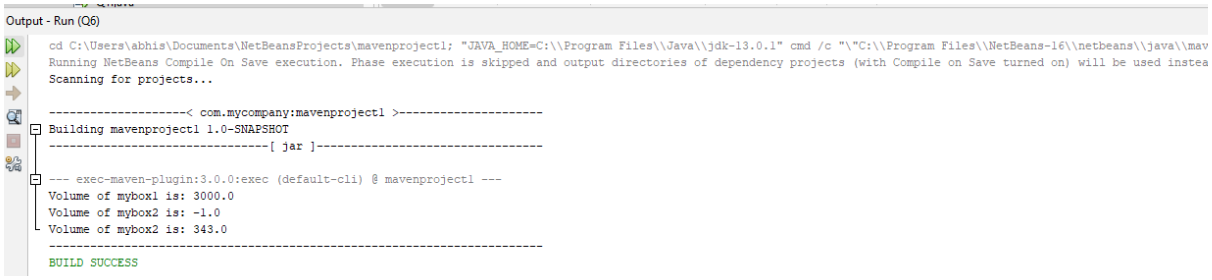
```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\jdk-13.0.1" cmd /c "\"C:\\Program Files\\NetBeans-16\\netbeans\\java\\maven\\
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
account created!
amount deposited.
total balance is:12000.0
amount withdrawn.
total balance is:9000.0
amount number: 501
name: John
account name: 54456KLQDH
total balance: 9000.000000
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

6. Write a Java program to implement constructor overloading.

CODE:

```
import static java.lang.System.out;

class Box {
   double width, height, depth;

   Box() {
      width = height = depth = -1;
   }

   Box(double len) {
      width = height = depth = len;
   }

   Box(double w, double h, double d) {
      width = w;
      height = h;
      depth = d;
   }

   double volume() {
      return width * height * depth;
   }
}

class Q6 {
   public static void main(String[] args) {
      Box mybox1 = new Box(10, 20, 15),
         mybox2 = new Box(),
         mybox3 = new Box(7);
         double vol;
         vol = mybox1.volume();
      out.println("Volume of mybox1 is: " + vol);
      vol = mybox2.volume();
      out.println("Volume of mybox2 is: " + vol);
      vol = mybox3.volume();
      out.println("Volume of mybox2 is: " + vol);
   }
}
```

OUTPUT:



Output - Run (Q6)

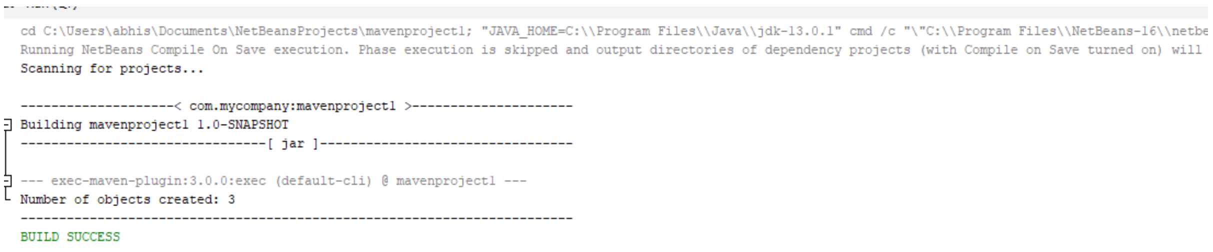
```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\jdk-13.0.1" cmd /c "\"C:\\Program Files\\NetBeans-16\\netbeans\\java\\mav
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instea
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Volume of mybox1 is: 3000.0
Volume of mybox2 is: -1.0
Volume of mybox2 is: 343.0
------------------------------------------------------------------------
BUILD SUCCESS
```

7. Write a Java program to count the no. of objects created in a program.

CODE:

```
class Q7 {
  static int count = 0;

  public Q7() {
    count++;
  }

  public static void main(String[] args) {
    Q7 object1 = new Q7();
    Q7 object2 = new Q7();
    Q7 object3 = new Q7();

    System.out.println("Number of objects created: " + count);
  }
}
```

OUTPUT:



```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\jdk-13.0.1" cmd /c "\"C:\\Program Files\\NetBeans-16\\netbe
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Number of objects created: 3
------------------------------------------------------------------------
BUILD SUCCESS
```

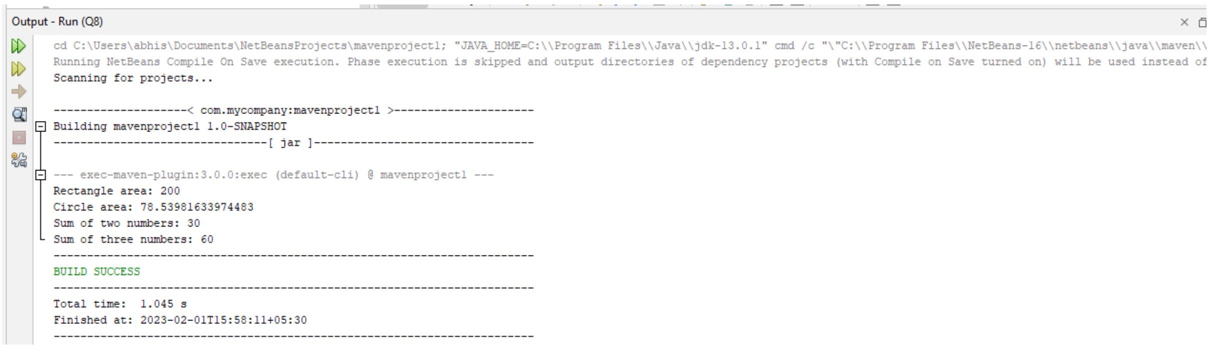
8. .Write a Java program to implement method over ridding & method overloading.

CODE:

```
package com.mycompany.mavenproject1;
class Shape {
  void area() {
    System.out.println("Shape area calculation");
  }
}

class Rectangle extends Shape {
  int length, width;

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

  @Override
  void area() {
    System.out.println("Rectangle area: " + length * width);
  }
}

class Circle extends Shape {
  int radius;

  Circle(int radius) {
    this.radius = radius;
  }

  @Override
  void area() {
    System.out.println("Circle area: " + Math.PI * radius * radius);
  }
}

class Q8 {
  static void displayArea(Shape shape) {
    shape.area();
  }

  static int add(int a, int b) {
    return a + b;
  }

  static int add(int a, int b, int c) {
    return a + b + c;
  }

  public static void main(String[] args) {
    Rectangle rectangle = new Rectangle(10, 20);
    Circle circle = new Circle(5);

    displayArea(rectangle);
    displayArea(circle);

    System.out.println("Sum of two numbers: " + add(10, 20));
    System.out.println("Sum of three numbers: " + add(10, 20, 30));
```

```
 }
}
```

OUTPUT:



Output - Run (Q8)

```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\jdk-13.0.1" cmd /c "\"C:\\Program Files\\NetBeans-16\\netbeans\\java\\maven\\
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Rectangle area: 200
Circle area: 78.53981633974483
Sum of two numbers: 30
Sum of three numbers: 60
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.045 s
Finished at: 2023-02-01T15:58:11+05:30
------------------------------------------------------------------------
```

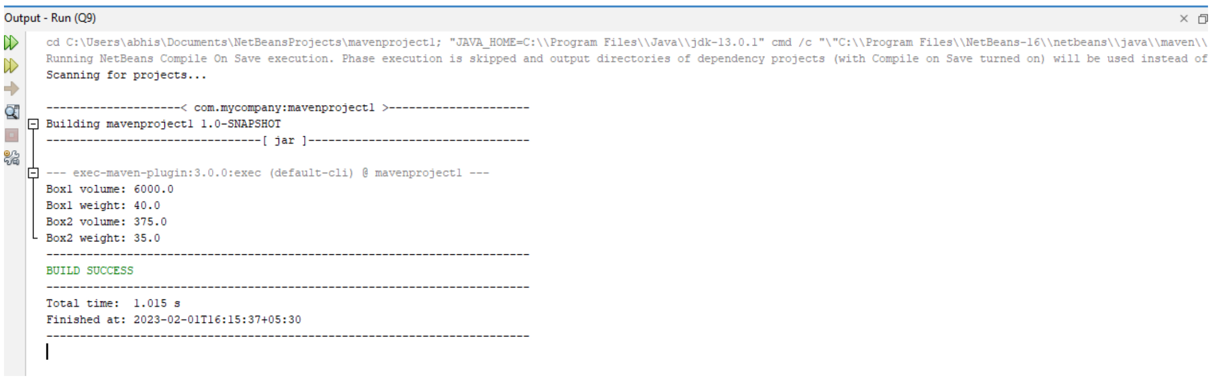
In this example, the Shape class is the superclass, and the Rectangle and Circle classes are subclasses that inherit from Shape and override its area method to provide their own implementation. The Q8 class contains two methods displayArea and add to demonstrate method overloading, where add is overloaded with two different implementations based on the number of arguments passed to it. The main method creates objects of Rectangle and Circle and calls their area method through the displayArea method, and also demonstrates method overloading by calling the add method with different arguments.

9. Create a class box having height, width, depth as the instance variables & calculate its volume. Implement constructor overloading in it. Create a subclass named box_new that has weight as an instance variable. Use super in the box_new class to initialize members of the base class.

CODE:

```
package com.mycompany.mavenproject1;

class Box {
  double height, width, depth;

  Box(double height, double width, double depth) {
    this.height = height;
    this.width = width;
    this.depth = depth;
  }

  Box(double height, double width) {
    this(height, width, 1.0);
  }

  double volume() {
    return height * width * depth;
  }
}

class Box_new extends Box {
  double weight;

  Box_new(double height, double width, double depth, double weight) {
    super(height, width, depth);
    this.weight = weight;
  }

  Box_new(double height, double width, double weight) {
    super(height, width);
    this.weight = weight;
  }
}

class Q9 {
  public static void main(String[] args) {
    Box_new box1 = new Box_new(10, 20, 30, 40);
    Box_new box2 = new Box_new(15, 25, 35);

    System.out.println("Box1 volume: " + box1.volume());
    System.out.println("Box1 weight: " + box1.weight);

    System.out.println("Box2 volume: " + box2.volume());
    System.out.println("Box2 weight: " + box2.weight);
  }
}
```

OUTPUT:



Output - Run (Q9)

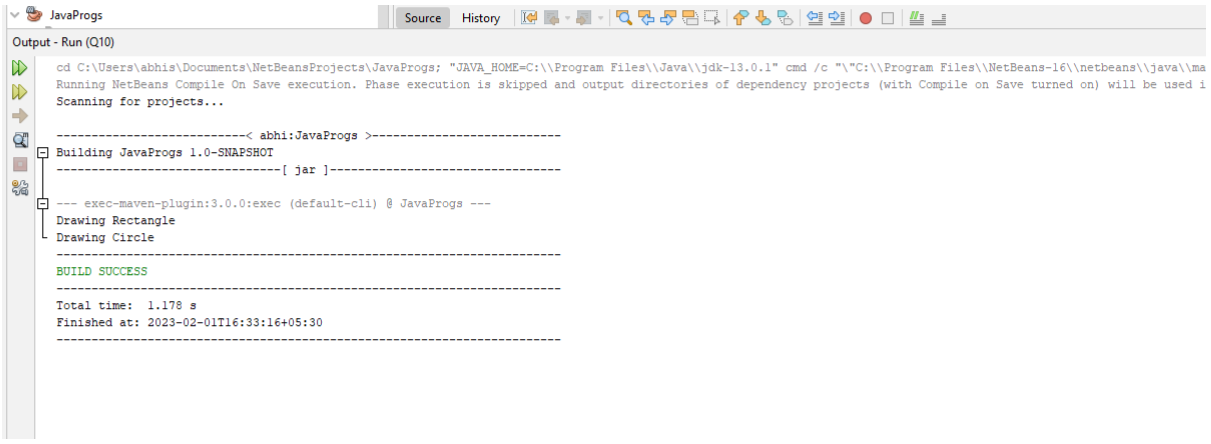
```
cd C:\Users\abhis\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Program Files\\Java\\jdk-13.0.1" cmd /c "\"C:\\Program Files\\NetBeans-16\\netbeans\\java\\maven\\
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used instead of
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >---------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Box1 volume: 6000.0
Box1 weight: 40.0
Box2 volume: 375.0
Box2 weight: 35.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.015 s
Finished at: 2023-02-01T16:15:37+05:30
------------------------------------------------------------------------
```

10.Write a Java program to implement run time polymorphism.

CODE:

```
class Shape {
  void draw() {
    System.out.println("Drawing Shape");
  }
}

class Rectangle extends Shape {
  @Override
  void draw() {
    System.out.println("Drawing Rectangle");
  }
}

class Circle extends Shape {
  @Override
  void draw() {
    System.out.println("Drawing Circle");
  }
}

public class Q10 {
  public static void main(String[] args) {
    Shape s;
    s = new Rectangle();
    s.draw();

    s = new Circle();

    s.draw();

  }

}
```

OUTPUT:



```
cd C:\Users\abhis\Documents\NetBeansProjects\JavaProgs; "JAVA_HOME=C:\\Program Files\\Java\\jdk-13.0.1" cmd /c "\"C:\\Program Files\\NetBeans-16\\netbeans\\java\\ma
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be used i
Scanning for projects...

---------------------------< abhi:JavaProgs >---------------------------
Building JavaProgs 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ JavaProgs ---
Drawing Rectangle
Drawing Circle
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.178 s
Finished at: 2023-02-01T16:33:16+05:30
------------------------------------------------------------------------
```

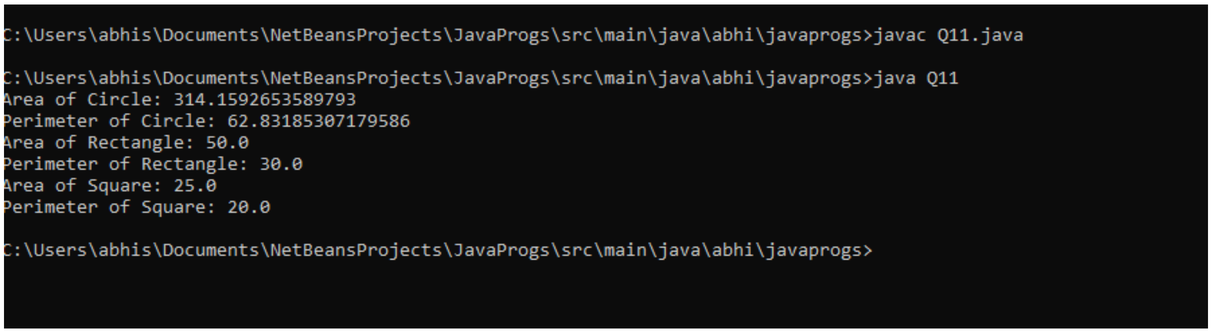
This program demonstrates run-time polymorphism through the use of a base class Shape and two subclasses, Rectangle and Circle, which both extend Shape and provide their own implementation of the draw() method. In the main method of Q10, we create instances of both Rectangle and Circle and assign them to a reference of the base class Shape. This allows us to call the draw() method on the objects, with the correct implementation being called at runtime based on the actual type of the object.

11. Write a Java program to implement interface. Create an interface named shape having area () & perimeter () as its methods. Create three classes circle, rectangle & square that implement this interface.

CODE:

```
interface Shape {
  double area();
  double perimeter();
}

class Circle implements Shape {
  double radius;

  Circle(double radius) {
    this.radius = radius;
  }

  @Override
  public double area() {
    return Math.PI * radius * radius;
  }

  @Override
  public double perimeter() {
    return 2 * Math.PI * radius;
  }
}

class Rectangle implements Shape {
  double width;
  double height;

  Rectangle(double width, double height) {
    this.width = width;
    this.height = height;
  }

  @Override
  public double area() {
    return width * height;
  }

  @Override
  public double perimeter() {
    return 2 * (width + height);
  }
}

class Square implements Shape {
  double side;

  Square(double side) {
    this.side = side;
  }

  @Override
  public double area() {
    return side * side;
  }
  @Override
  public double perimeter() {
    return 4 * side;
  }
}

public class Q11 {
  public static void main(String[] args) {
    Shape shape1 = new Circle(10);
    System.out.println("Area of Circle: " + shape1.area());
    System.out.println("Perimeter of Circle: " + shape1.perimeter());

    Shape shape2 = new Rectangle(5, 10);
    System.out.println("Area of Rectangle: " + shape2.area());
    System.out.println("Perimeter of Rectangle: " + shape2.perimeter());

    Shape shape3 = new Square(5);
    System.out.println("Area of Square: " + shape3.area());
    System.out.println("Perimeter of Square: " + shape3.perimeter());
  }
}
```

OUTPUT:



```
:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q11.java

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q11
rea of Circle: 314.1592653589793
erimeter of Circle: 62.83185307179586
rea of Rectangle: 50.0
erimeter of Rectangle: 30.0
rea of Square: 25.0
erimeter of Square: 20.0

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

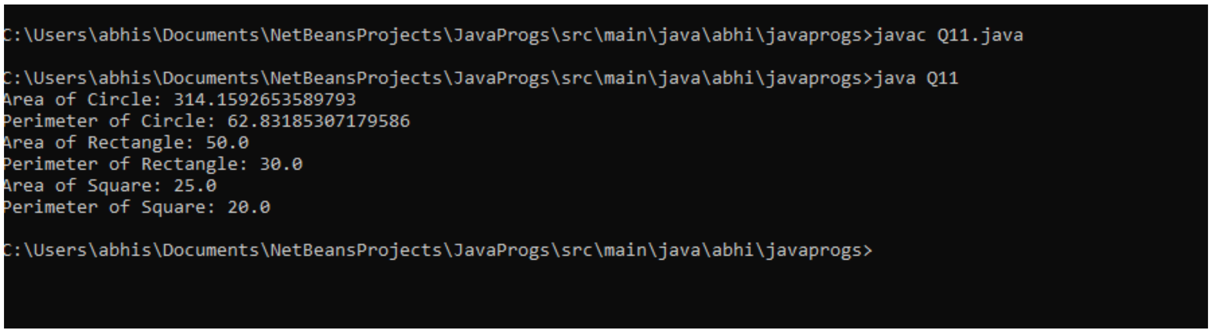
In this example, we define an interface **Shape** that includes two methods, **area()** and **perimeter()**. The **Circle**, **Rectangle**, and **Square** classes all implement the **Shape** interface and provide their own implementation of the **area()** and **perimeter()** methods. In the main method of **Q11**, we create instances of each of the classes and assign them to references of the **Shape** type. This allows us to call the **area()** and **perimeter()** methods on the objects, with the correct implementation being called at runtime based on the actual type of the object.

12.Write a Java program to show multiple inheritance.

CODE:

```
interface Shape {
  double area();
}

interface Printable {
  void print();
}

class Circle implements Shape, Printable {
  double radius;

  Circle(double radius) {
    this.radius = radius;
  }

  @Override
  public double area() {
    return Math.PI * radius * radius;
  }

  @Override
  public void print() {
    System.out.println("Circle: " + area());
  }
}

public class Q12 {
  public static void main(String[] args) {
    Circle c = new Circle(10);
    c.print();
  }
}
```

OUTPUT:



```
:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q11.java

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q11
rea of Circle: 314.1592653589793
erimeter of Circle: 62.83185307179586
rea of Rectangle: 50.0
erimeter of Rectangle: 30.0
rea of Square: 25.0
erimeter of Square: 20.0

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

Java does not support multiple inheritance, meaning a class cannot inherit from multiple classes. However, it is possible to achieve similar functionality through the use of interfaces. An interface can extend multiple interfaces, which allows a class to implement multiple interfaces.

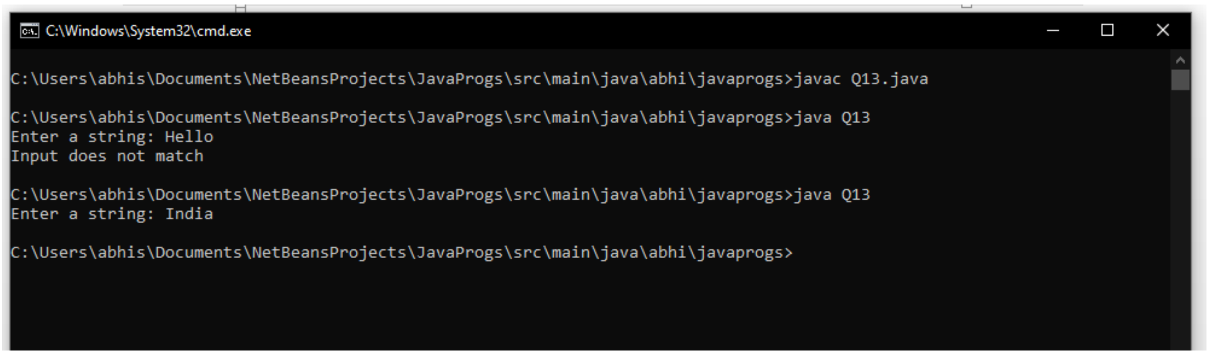
In this example, we define two interfaces, **Shape** and **Printable**, that include methods **area()** and **print()**, respectively. The **Circle** class implements both **Shape** and **Printable**, which allows it to inherit the methods from both interfaces. In the main method of **Q12**, we create an instance of the **Circle** class and call its **print()** method

13. Create a user defined exception named “nomatchexception” that is fired when the string entered by the user is not “India”.

CODE:

```
import java.util.Scanner;


class NoMatchException extends Exception {
   public NoMatchException(String message) {
      super(message);
   }
}


public class Q13 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a string: ");
      String input = sc.nextLine();
      try {
         if (!input.equals("India")) {
            throw new NoMatchException("Input does not match");
         }
      } catch (NoMatchException e) {
         System.out.println(e.getMessage());
      }
   }
}
```

OUTPUT:


C:\Windows\System32\cmd.exe

```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q13.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q13
Enter a string: Hello
Input does not match

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q13
Enter a string: India

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

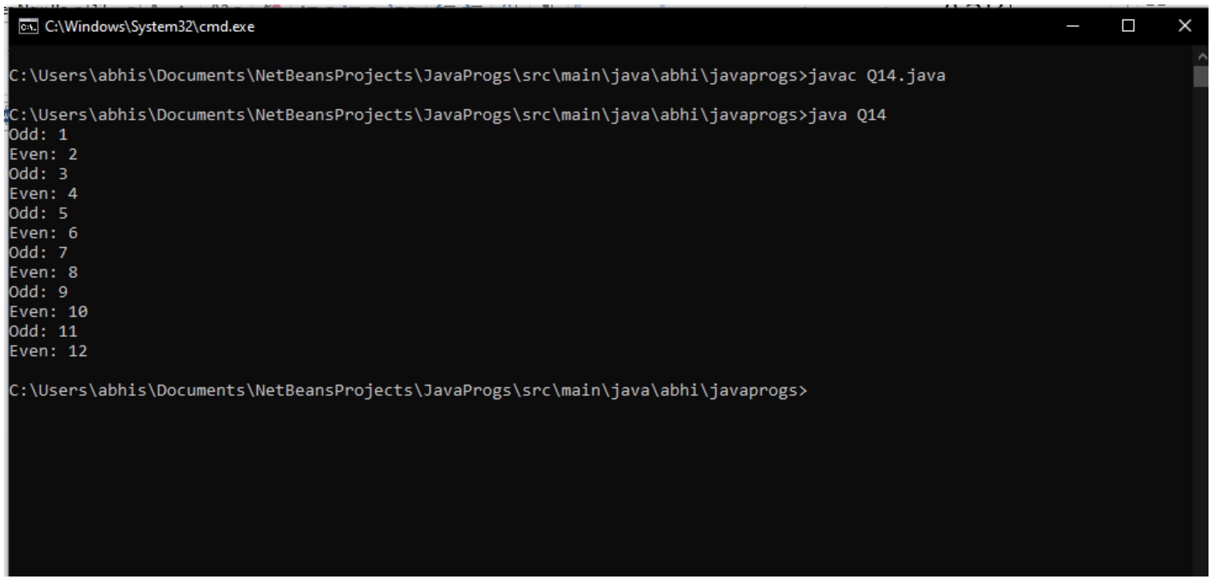
14.Write a Java program to show even & odd numbers by thread.

CODE:

```
class EvenThread implements Runnable {
   @Override
   public void run() {
      for (int i = 2; i <= 12; i += 2) {
         System.out.println("Even: " + i);
      }
   }
}

class OddThread implements Runnable {
   @Override
   public void run() {
      for (int i = 1; i <= 12; i += 2) {
         System.out.println("Odd: " + i);
      }
   }
}

public class Q14 {
   public static void main(String[] args) {
      Thread evenThread = new Thread(new EvenThread());
      Thread oddThread = new Thread(new OddThread());

      evenThread.start();
      oddThread.start();
   }
```

}

OUTPUT:



C:\Windows\System32\cmd.exe

```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q14.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q14
Odd: 1
Even: 2
Odd: 3
Even: 4
Odd: 5
Even: 6
Odd: 7
Even: 8
Odd: 9
Even: 10
Odd: 11
Even: 12

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

In this program, two separate threads are created to display even and odd numbers. The **run** method of each thread class contains a loop to print even and odd numbers respectively. The **start** method of each thread is called to start the execution of each thread.

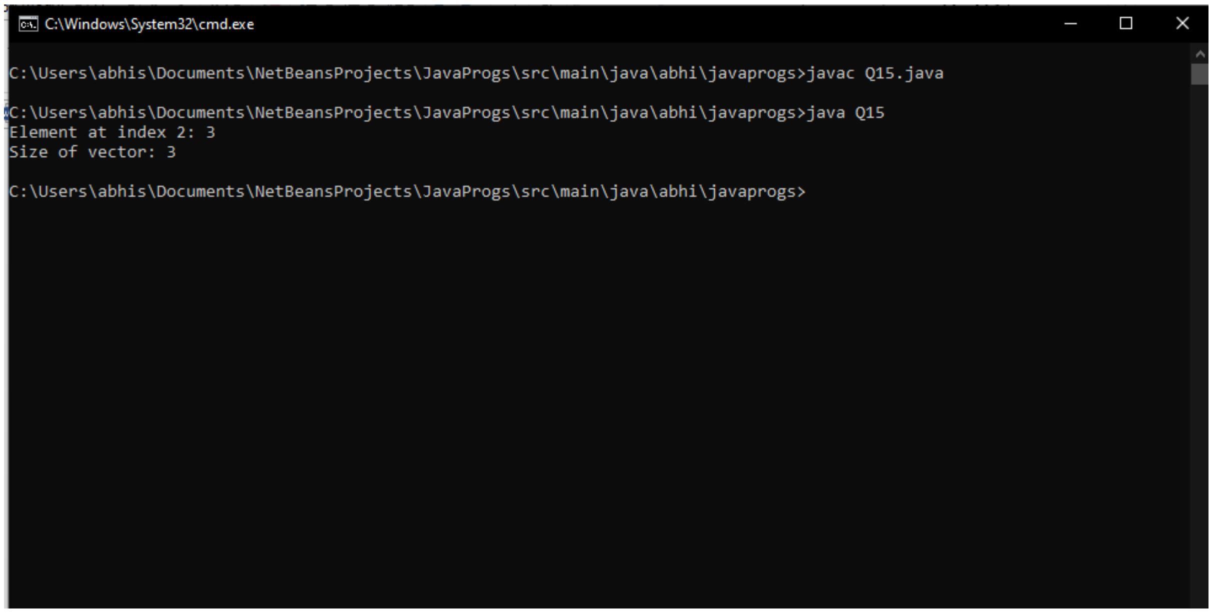
15. Write a Java program to implement vector. [use: addelement(),elementat().removeelement(),size().]

CODE:

```
import java.util.ArrayList;


class Vector<T> {
   private ArrayList<T> list;


   public Vector() {
      list = new ArrayList<T>();
   }


   public void addElement(T element) {
      list.add(element);
   }


   public T elementAt(int index) {
      return list.get(index);
   }


   public void removeElement(int index) {
      list.remove(index);
   }


   public int size() {
      return list.size();
   }
}
```

```
public class Q15 {
  public static void main(String[] args) {
    Vector<Integer> vec = new Vector<Integer>();
    vec.addElement(1);
    vec.addElement(2);
    vec.addElement(3);
    vec.addElement(4);

    System.out.println("Element at index 2: " + vec.elementAt(2));
    vec.removeElement(2);
    System.out.println("Size of vector: " + vec.size());
  }
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q15.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q15
Element at index 2: 3
Size of vector: 3

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

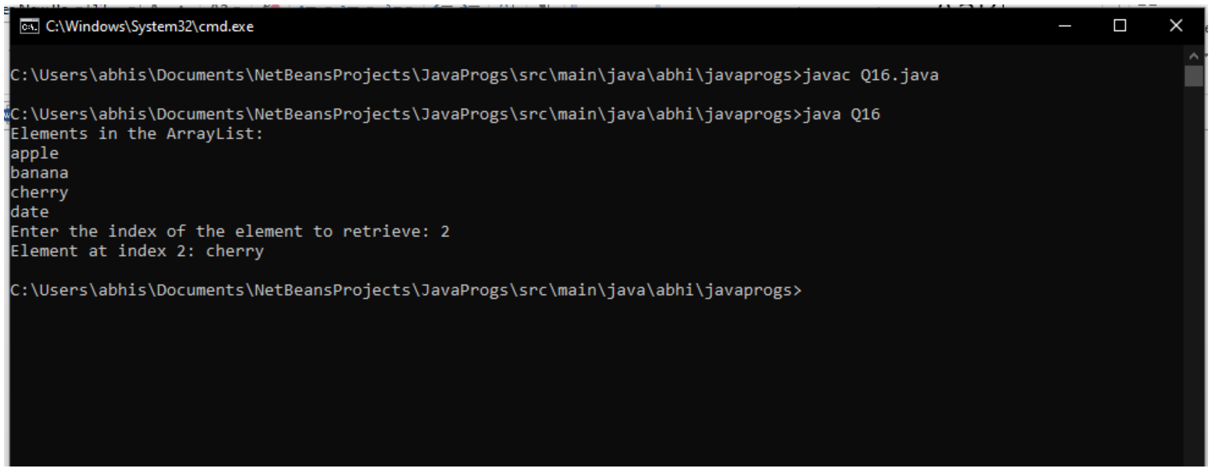
In this program, the Vector class is implemented using an ArrayList to store the elements. The methods **addElement**, **elementAt**, **removeElement**, and **size** are implemented to add an element, get an element at a specific index, remove an element at a specific index, and get the size of the vector, respectively. The main class **Q15** demonstrates how to use the Vector class

16.Write a Java program to iterate through all elements in a array list , and retrieve an element (at a specified index)

CODE:

```
import java.util.ArrayList;
import java.util.Scanner;

public class Q16 {
  public static void main(String[] args) {
    ArrayList<String> list = new ArrayList<String>();
    list.add("apple");
    list.add("banana");
    list.add("cherry");
    list.add("date");

    System.out.println("Elements in the ArrayList: ");
    for (String s : list) {
      System.out.println(s);
    }

    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the index of the element to retrieve: ");
    int index = sc.nextInt();
    if (index >= 0 && index < list.size()) {
      System.out.println("Element at index " + index + ": " + list.get(index));
    } else {
      System.out.println("Invalid index");
    }
    }
}
```

OUTPUT:



```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q16.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q16
Elements in the ArrayList:
apple
banana
cherry
date
Enter the index of the element to retrieve: 2
Element at index 2: cherry

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

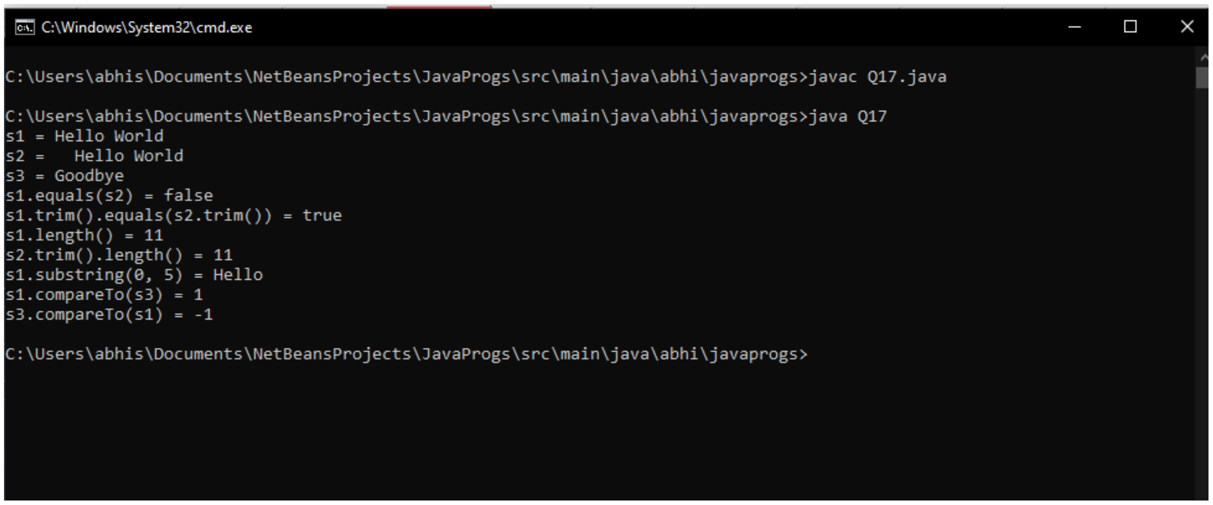
In this program, an ArrayList is created to store a list of elements. A for-each loop is used to iterate through all elements in the ArrayList and print them. The user is prompted to enter the index of an element to retrieve using a Scanner. The element at the specified index is retrieved using the **get** method of the ArrayList and printed. If the specified index is invalid, an error message is printed.

17.Write a Java program to demonstrate the use of equals(), trim() ,length() , substring(), compareto() of string class.

CODE:

```
public class Q17 {
  public static void main(String[] args) {
    String s1 = "Hello World";
    String s2 = "  Hello World  ";
    String s3 = "Goodbye";

    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
    System.out.println("s3 = " + s3);

    System.out.println("s1.equals(s2) = " + s1.equals(s2));
    System.out.println("s1.trim().equals(s2.trim()) = " + s1.trim().equals(s2.trim()));

    System.out.println("s1.length() = " + s1.length());
    System.out.println("s2.trim().length() = " + s2.trim().length());

    System.out.println("s1.substring(0, 5) = " + s1.substring(0, 5));

    System.out.println("s1.compareTo(s3) = " + s1.compareTo(s3));
    System.out.println("s3.compareTo(s1) = " + s3.compareTo(s1));
  }
}
```

OUTPUT:



```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q17.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q17
s1 = Hello World
s2 =   Hello World
s3 = Goodbye
s1.equals(s2) = false
s1.trim().equals(s2.trim()) = true
s1.length() = 11
s2.trim().length() = 11
s1.substring(0, 5) = Hello
s1.compareTo(s3) = 1
s3.compareTo(s1) = -1

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

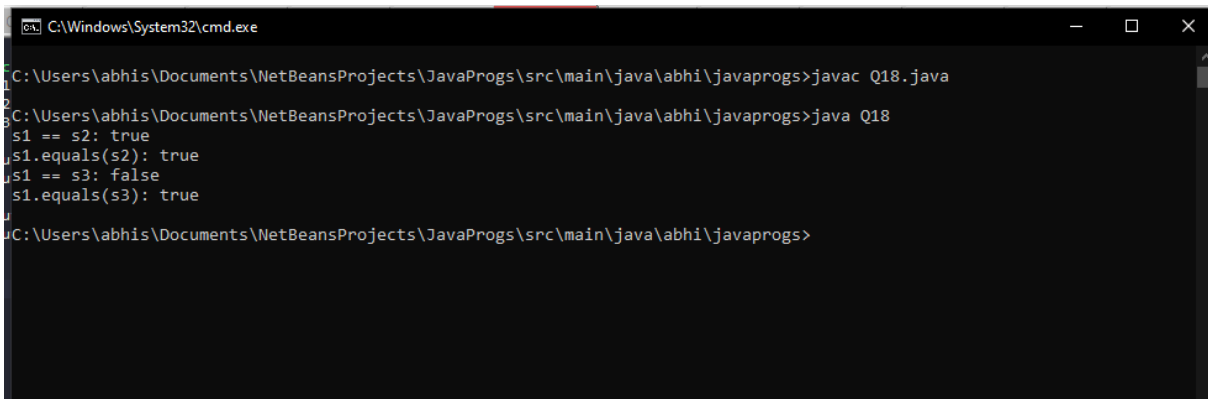
In this program, three **String** objects are created with different values. The **equals()** method is used to compare two strings for equality. The **trim()** method is used to remove leading and trailing whitespace from a string. The **length()** method is used to get the length of a string. The **substring()** method is used to get a part of a string. The **compareTo()** method is used to compare two strings lexicographically.

18.Write a Java program to demonstrate the use of equals() and == in Java.

CODE:

```
public class Q18 {

   public static void main(String[] args) {

       String s1 = "Hello";

       String s2 = "Hello";

       String s3 = new String("Hello");


       System.out.println("s1 == s2: " + (s1 == s2));

       System.out.println("s1.equals(s2): " + s1.equals(s2));


       System.out.println("s1 == s3: " + (s1 == s3));

       System.out.println("s1.equals(s3): " + s1.equals(s3));

   }

}
```

OUTPUT:



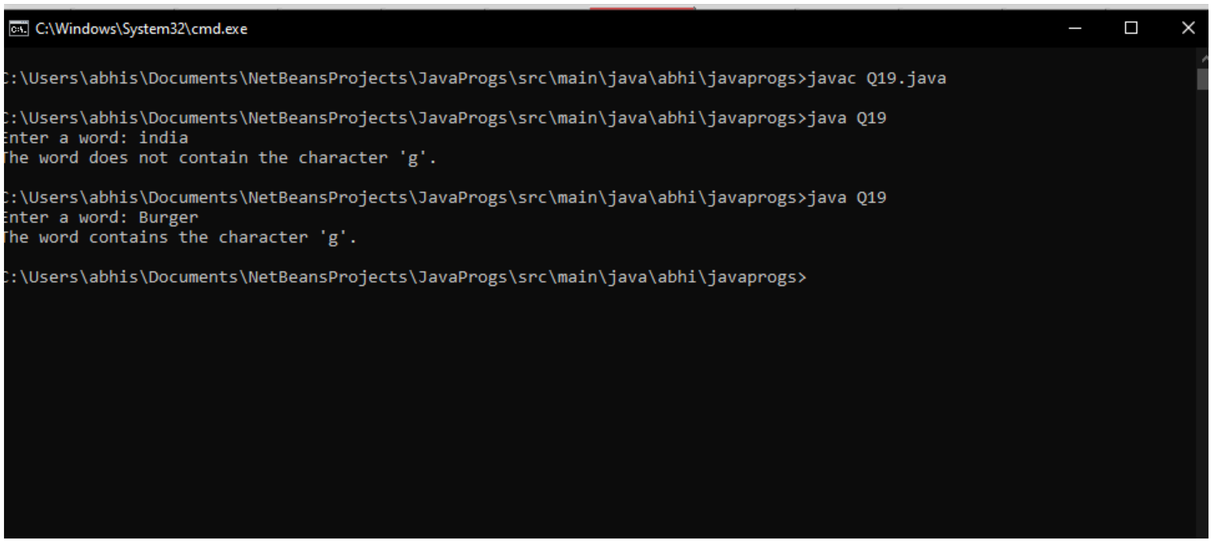
```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q18.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q18
s1 == s2: true
s1.equals(s2): true
s1 == s3: false
s1.equals(s3): true

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

19. Write a Java program to check a word contains the character 'g' in a given string.

CODE:

```
import java.util.Scanner;


public class Q19 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a word: ");
        String word = input.nextLine();
        if (word.contains("g")) {
            System.out.println("The word contains the character 'g'.");
        } else {
            System.out.println("The word does not contain the character 'g'.");
        }
    }
}
```

OUTPUT:



```
:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q19.java

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q19
nter a word: india
he word does not contain the character 'g'.

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q19
nter a word: Burger
he word contains the character 'g'.

:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

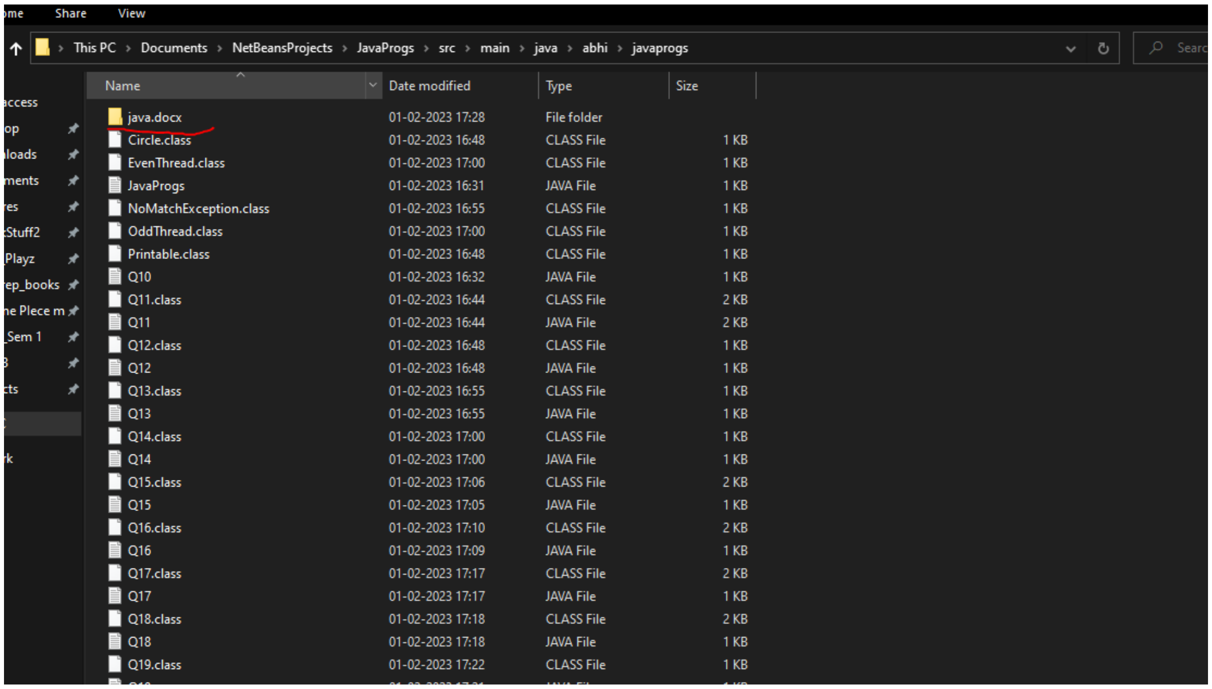
In this program, the **Scanner** class is used to get input from the user. The user is prompted to enter a word. The **contains()** method of the **String** class is used to check if the word contains the character 'g'. If the word contains the character 'g', a message is printed indicating that the word contains the character 'g'. If the word does not contain the character 'g', a message is printed indicating that the word does not contain the character 'g'.

20. Write a java program to create a folder named “java.docx” in which you have to make a file as abc.java and other 3 text files as,total_char, total_lines and total_words. Read the data from the abc.java file and write the values to the 3 files respectively as per the name . eg: write the total no of words in the abc.java file in total_words.txt.

CODE:

```
import java.io.*;


public class Q20 {

    public static void main(String[] args) {

        File javaDocx = new File("java.docx");

        javaDocx.mkdir();


        File abcJava = new File(javaDocx, "abc.java");

        File totalChar = new File(javaDocx, "total_char.txt");

        File totalLines = new File(javaDocx, "total_lines.txt");

        File totalWords = new File(javaDocx, "total_words.txt");


        try {

            abcJava.createNewFile();

            totalChar.createNewFile();

            totalLines.createNewFile();

            totalWords.createNewFile();


            BufferedReader reader = new BufferedReader(new FileReader(abcJava));

            BufferedWriter charWriter = new BufferedWriter(new FileWriter(totalChar));

            BufferedWriter lineWriter = new BufferedWriter(new FileWriter(totalLines));

            BufferedWriter wordWriter = new BufferedWriter(new FileWriter(totalWords));
            int charCount = 0;
            int lineCount = 0;
            int wordCount = 0;

            String line;
            while ((line = reader.readLine()) != null) {
                lineCount++;
                charCount += line.length();
                String[] words = line.split(" ");
                wordCount += words.length;
            }

            charWriter.write("Total characters: " + charCount);
            lineWriter.write("Total lines: " + lineCount);
            wordWriter.write("Total words: " + wordCount);

            reader.close();
            charWriter.close();
            lineWriter.close();
            wordWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

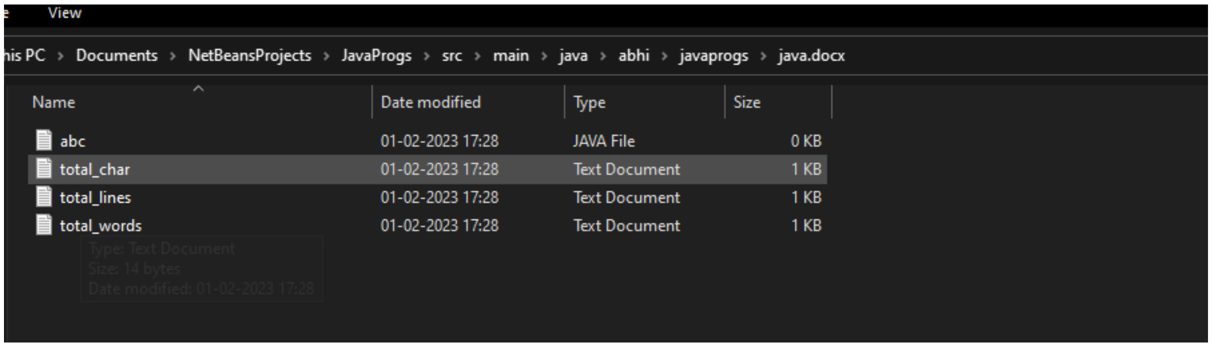
OUTPUT:



ome Share View

This PC › Documents › NetBeansProjects › JavaProgs › src › main › java › abhi › javaprogs

Name	Date modified	Type	Size
java.docx	01-02-2023 17:28	File folder	
Circle.class	01-02-2023 16:48	CLASS File	1 KB
EvenThread.class	01-02-2023 17:00	CLASS File	1 KB
JavaProgs	01-02-2023 16:31	JAVA File	1 KB
NoMatchException.class	01-02-2023 16:55	CLASS File	1 KB
OddThread.class	01-02-2023 17:00	CLASS File	1 KB
Printable.class	01-02-2023 16:48	CLASS File	1 KB
Q10	01-02-2023 16:32	JAVA File	1 KB
Q11.class	01-02-2023 16:44	CLASS File	2 KB
Q11	01-02-2023 16:44	JAVA File	2 KB
Q12.class	01-02-2023 16:48	CLASS File	1 KB
Q12	01-02-2023 16:48	JAVA File	1 KB
Q13.class	01-02-2023 16:55	CLASS File	1 KB
Q13	01-02-2023 16:55	JAVA File	1 KB
Q14.class	01-02-2023 17:00	CLASS File	1 KB
Q14	01-02-2023 17:00	JAVA File	1 KB
Q15.class	01-02-2023 17:06	CLASS File	2 KB
Q15	01-02-2023 17:05	JAVA File	1 KB
Q16.class	01-02-2023 17:10	CLASS File	2 KB
Q16	01-02-2023 17:09	JAVA File	1 KB
Q17.class	01-02-2023 17:17	CLASS File	2 KB
Q17	01-02-2023 17:17	JAVA File	1 KB
Q18.class	01-02-2023 17:18	CLASS File	2 KB
Q18	01-02-2023 17:18	JAVA File	1 KB
Q19.class	01-02-2023 17:22	CLASS File	1 KB

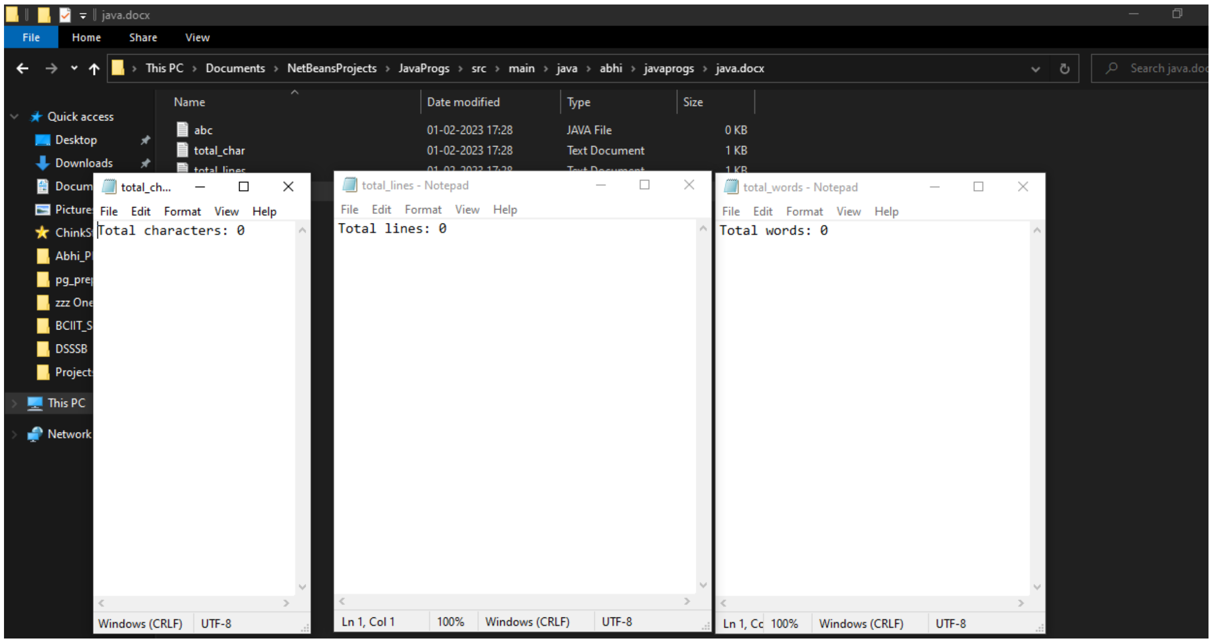


View

his PC › Documents › NetBeansProjects › JavaProgs › src › main › java › abhi › javaprogs › java.docx

Name	Date modified	Type	Size
abc	01-02-2023 17:28	JAVA File	0 KB
total_char	01-02-2023 17:28	Text Document	1 KB
total_lines	01-02-2023 17:28	Text Document	1 KB
total_words	01-02-2023 17:28	Text Document	1 KB

Type: Text Document
Size: 14 bytes
Date modified: 01-02-2023 17:28

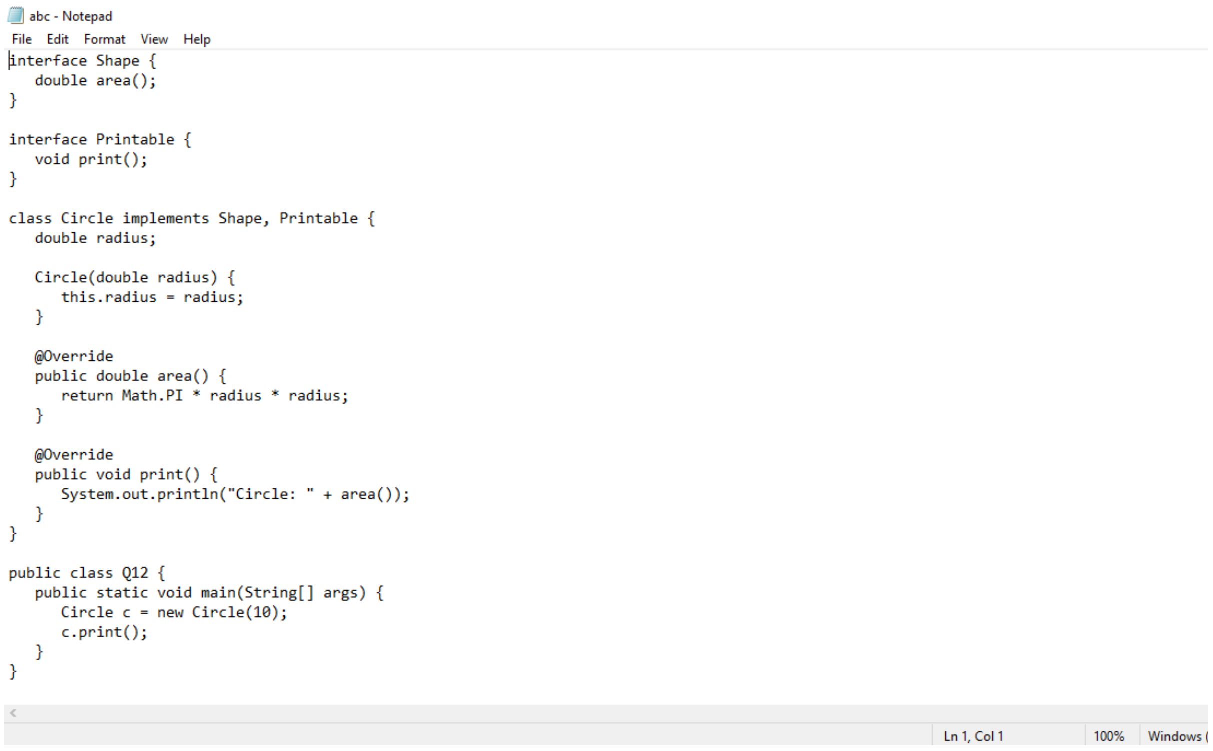

java.docx
File Home Share View
This PC > Documents > NetBeansProjects > JavaProgs > src > main > java > abhi > javaprogs > java.docx
Search java.doc
Quick access
Desktop
Downloads
This PC
Network
Name Date modified Type Size
abc 01-02-2023 17:28 JAVA File 0 KB
total_char 01-02-2023 17:28 Text Document 1 KB
total_ch...
total_lines - Notepad
total_words - Notepad
File Edit Format View Help
Windows (CRLF) UTF-8
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
Total characters: 0
```

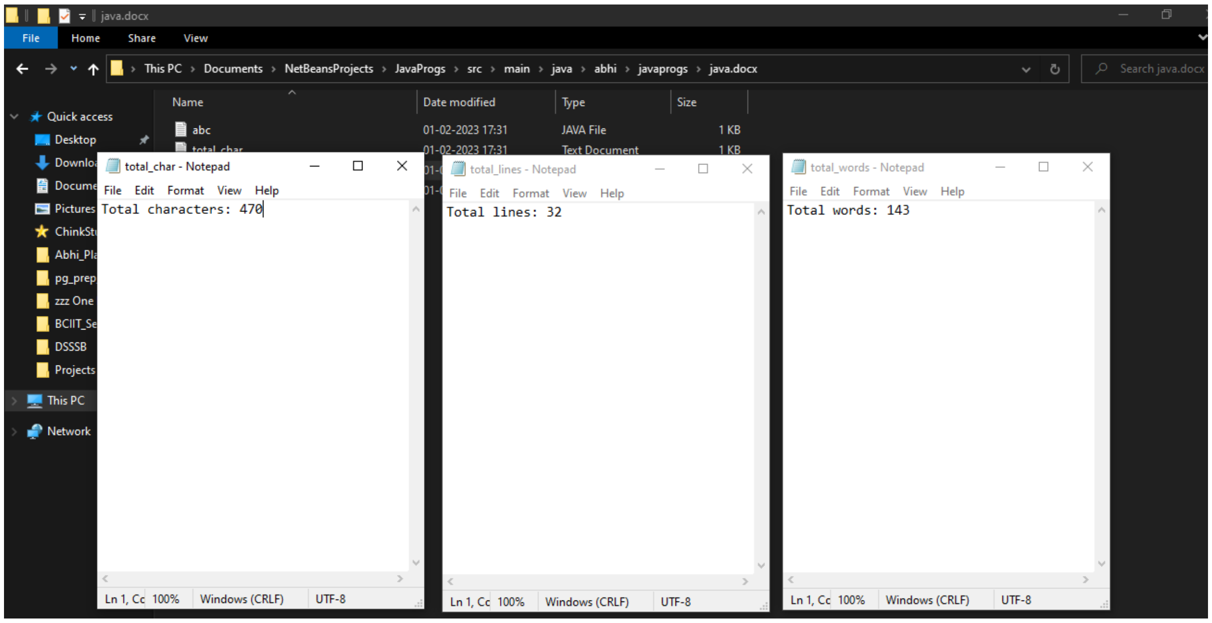
```
Total lines: 0
```

```
Total words: 0
```

Now, filling abc.java with sample contents from a previous program:



```
interface Shape {
   double area();
}

interface Printable {
   void print();
}

class Circle implements Shape, Printable {
   double radius;

   Circle(double radius) {
      this.radius = radius;
   }

   @Override
   public double area() {
      return Math.PI * radius * radius;
   }

   @Override
   public void print() {
      System.out.println("Circle: " + area());
   }
}

public class Q12 {
   public static void main(String[] args) {
      Circle c = new Circle(10);
      c.print();
   }
}
```



```
Total characters: 470
```

```
Total lines: 32
```

```
Total words: 143
```

In this program, the **File** class is used to create the folder **java.docx** and the four files. The **BufferedReader** and **BufferedWriter** classes are used to read and write data to the files. The program keeps track of the total number of characters, lines, and words in the **abc.java** file. The values are written to the **total_char.txt**, **total_lines.txt**, and **total_words.txt** files respectively. Note that the contents of the **abc.java** file are assumed to be in the form of a string. The program assumes that there is at least one line of text in the file.

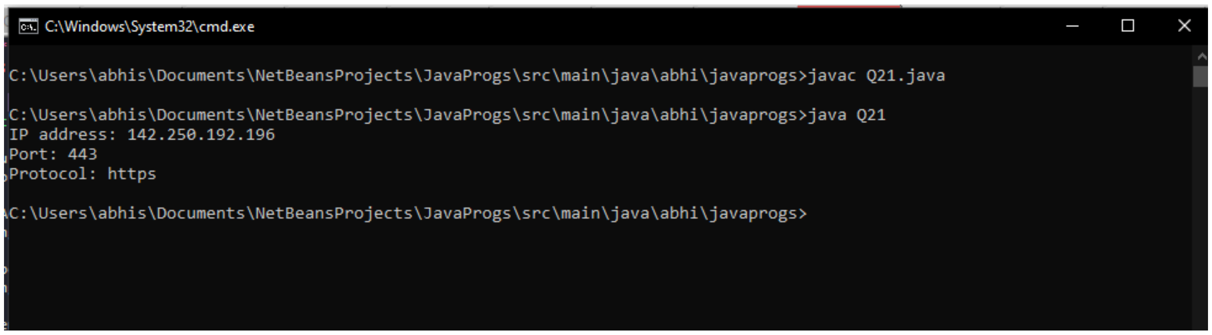
21.write a java program to get a websites detail such as its Ip address, port number, protocols etc.

CODE:

```
import java.net.*;

import java.io.*;


public class Q21 {

   public static void main(String[] args) {

      try {

         URL url = new URL("https://www.google.com");

         URLConnection conn = url.openConnection();


         InetAddress address = InetAddress.getByName(url.getHost());

         String ip = address.getHostAddress();


         int port = url.getDefaultPort();

         String protocol = url.getProtocol();


         System.out.println("IP address: " + ip);

         System.out.println("Port: " + port);

         System.out.println("Protocol: " + protocol);

      } catch (MalformedURLException e) {

         System.out.println("Invalid URL");

      } catch (UnknownHostException e) {

         System.out.println("Unknown host");

      } catch (IOException e) {

         System.out.println("IOException");

      }
```

```
    }
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q21.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q21
IP address: 142.250.192.196
Port: 443
Protocol: https

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

In this program, the **URL** class is used to represent the website, and the **URLConnection** class is used to open a connection to the website. The **InetAddress** class is used to retrieve the IP address of the website, and the **getHostAddress** method is used to retrieve the IP address as a string. The **getDefaultPort** method is used to retrieve the port number, and the **getProtocol** method is used to retrieve the protocol used by the website.

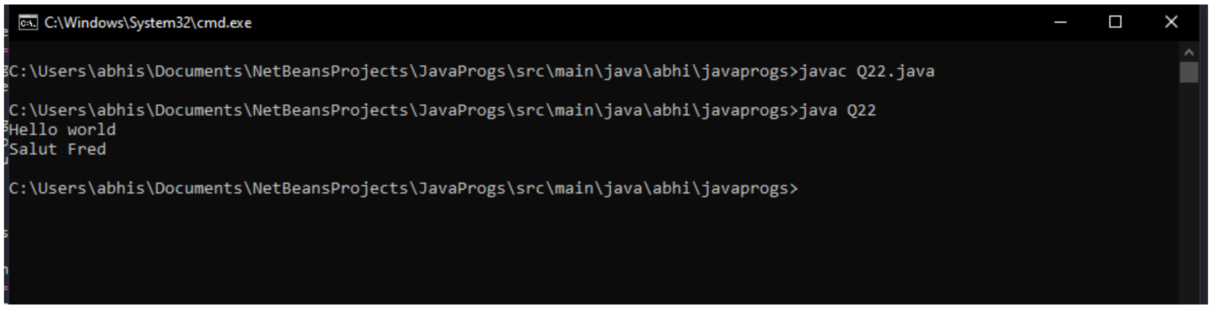
22. Write a java program to implement anonymous class.

CODE:

```
interface HelloWorld {
   void greet();
   void greetSomeone(String someone);
}

public class Q22 {
   public void sayHello() {

       class EnglishGreeting implements HelloWorld {
          String name = "world";
          public void greet() {
             greetSomeone("world");
          }
          public void greetSomeone(String someone) {
             name = someone;
             System.out.println("Hello " + name);
          }
       }

       HelloWorld englishGreeting = new EnglishGreeting();

       HelloWorld frenchGreeting = new HelloWorld() {
          String name = "Bonjour";
          public void greet() {
             greetSomeone("Bonjour");
          }
```

```
            public void greetSomeone(String someone) {
                name = someone;
                System.out.println("Salut " + name);
            }
        };

        englishGreeting.greet();
        frenchGreeting.greetSomeone("Fred");
    }

    public static void main(String... args) {
        Q22 myApp = new Q22();
        myApp.sayHello();
    }
}
```

OUTPUT:



```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q22.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q22
Hello world
Salut Fred

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

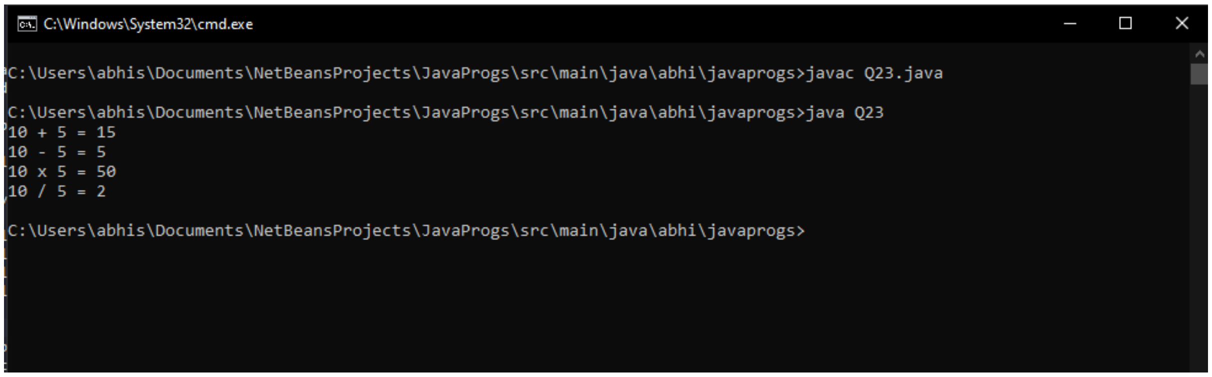
In this program, an interface **HelloWorld** is defined with two methods **greet** and **greetSomeone**. The main class **Q22** implements a method **sayHello** that creates two instances of **HelloWorld**. The first instance is a regular inner class named **EnglishGreeting**, while the second instance is an anonymous class. Both classes implement the **HelloWorld** interface and override the **greet** and **greetSomeone** methods. The **main** method creates an instance of the **Q22** class and calls the **sayHello** method, which in turn creates the instances of the **HelloWorld** interface and calls the **greet** and **greetSomeone** methods on each instance.

23. Write a java program to implement lambda expressions.

CODE:

```
interface MathOperation {
    int operation(int a, int b);
}

public class Q23 {
    public static void main(String[] args) {
        MathOperation addition = (a, b) -> a + b;

        MathOperation subtraction = (int a, int b) -> a - b;

        MathOperation multiplication = (int a, int b) -> { return a * b; };

        MathOperation division = (a, b) -> a / b;

        System.out.println("10 + 5 = " + operate(10, 5, addition));
        System.out.println("10 - 5 = " + operate(10, 5, subtraction));
        System.out.println("10 x 5 = " + operate(10, 5, multiplication));
        System.out.println("10 / 5 = " + operate(10, 5, division));
    }

    private static int operate(int a, int b, MathOperation mathOperation) {
        return mathOperation.operation(a, b);
    }
}
```

OUTPUT:



```
C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>javac Q23.java

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>java Q23
10 + 5 = 15
10 - 5 = 5
10 x 5 = 50
10 / 5 = 2

C:\Users\abhis\Documents\NetBeansProjects\JavaProgs\src\main\java\abhi\javaprogs>
```

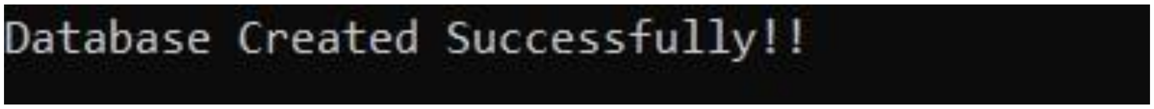
In this program, an interface **MathOperation** is defined with a single method **operation**. The main class **Q23** implements a method **operate** that takes two **int** values and an instance of **MathOperation** as arguments, and returns the result of the **operation** method. The **main** method defines four instances of **MathOperation** using lambda expressions. Each lambda expression defines a different mathematical operation (addition, subtraction, multiplication, division). The **main** method then calls the **operate** method for each **MathOperation** instance and prints the results.

24. Write a java problem to show java database connectivity.

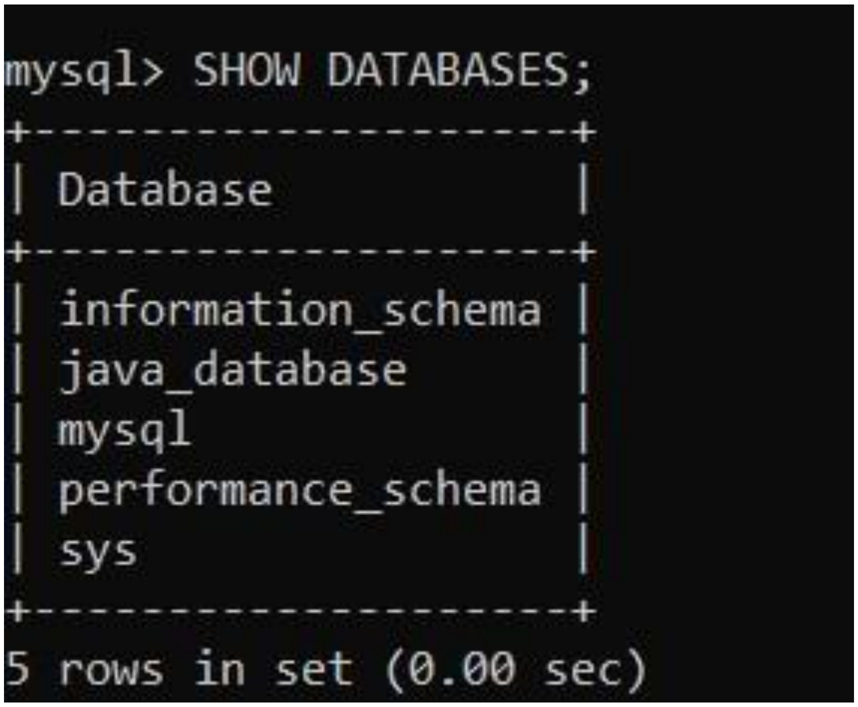
CODE:

```
import java.sql.*;

class Q24

{

public static void main(String[] args)

{

Connection connection = null;

String query;

ResultSet result; try

{

Class.forName("com.mysql.cj.jdbc.Driver");

connection = DriverManager.getConnection("jdbc:mysql://localhost:3308/",

"12345",  "12345");

Statement stmt = connection.createStatement();

query = "CREATE DATABASE java_database";

stmt.executeUpdate(query);

System.out.println("Database Created Successfully!!");

connection.close();

}

catch (Exception exception)

{

System.out.println(exception);

}

}

}
```

OUTPUT:



```
Database Created Successfully!!
```



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| java_database      |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

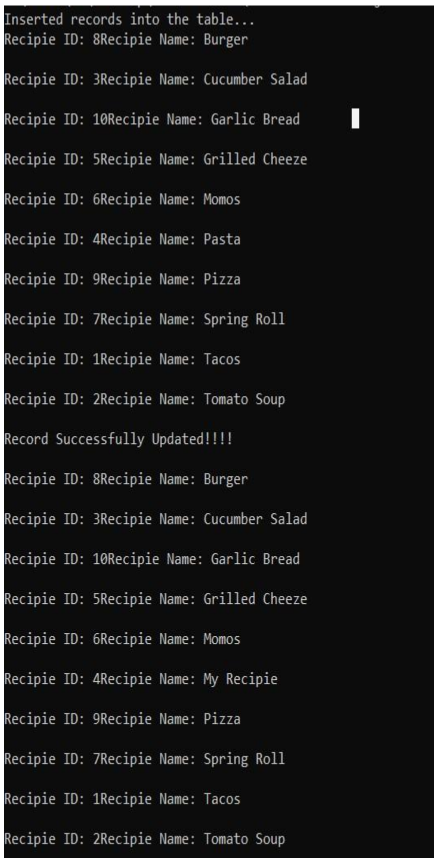
QUESTION 25: Write a Java program in Java to insert, update, delete & select records.

CODE:

```
import java.sql.*;

class Program37

{

    public static void main(String[] args)

    {

        Connection connection = null;

        String query;

        ResultSet result; try

        {
            Class.forName("com.mysql.cj.jdbc.Driver"); connection =

DriverManager.getConnection("jdbc:mysql://localhost:3308/java_database","12345", "12345");

            Statement stmt = connection.createStatement();


            query = "INSERT INTO Recipies(recipe_id, recipe_name)"

                    + " VALUES"

                    + "(6,\"Momos\"), "

                    + "(7,\"Spring Roll\"), "

                    + "(8,\"Burger\"), "

                    + "(9,\"Pizza\"), "

                    + "(10,\"Garlic Bread\"); ";

             stmt.executeUpdate(query); System.out.println("Inserted records
             into the table...");

             query = "SELECT * FROM Recipies;"; result =
             stmt.executeQuery(query); while(result.next()){

                 System.out.print("Recipie ID: " + result.getInt("recipe_id"));
```

```
         System.out.print("Recipie Name: " + result.getString("recipe_name"));

         System.out.println("\n");

    }

    query = "UPDATE Recipies " +

    "SET recipe_name = 'My Recipie' WHERE recipe_id = 4"; stmt.executeUpdate(query);

    System.out.println("Record Successfully Updated!!!!\n");
    query = "SELECT * FROM Recipies;"; result =
    stmt.executeQuery(query);

    while(result.next()){

           System.out.print("Recipie ID: " + result.getInt("recipe_id"));
        System.out.print("Recipie Name: " + result.getString("recipe_name"));
        System.out.println("\n");

     }


     query = "DELETE FROM Recipies where recipe_id = 3"; stmt.executeUpdate(query);

     System.out.println("Record Successfully Deleted!!!!\n");
     query = "SELECT * FROM Recipies;"; result =
     stmt.executeQuery(query); while(result.next()){

           System.out.print("Recipie ID: " + result.getInt("recipe_id"));
        System.out.print("|Recipie Name: " + result.getString("recipe_name"));
     System.out.println("\n"); }

    connection.close();

  } catch (Exception exception)

  {

    System.out.println(exception);

  }

}

}
```

OUTPUT:



```
Inserted records into the table...
Recipie ID: 8Recipie Name: Burger

Recipie ID: 3Recipie Name: Cucumber Salad

Recipie ID: 10Recipie Name: Garlic Bread

Recipie ID: 5Recipie Name: Grilled Cheeze

Recipie ID: 6Recipie Name: Momos

Recipie ID: 4Recipie Name: Pasta

Recipie ID: 9Recipie Name: Pizza

Recipie ID: 7Recipie Name: Spring Roll

Recipie ID: 1Recipie Name: Tacos

Recipie ID: 2Recipie Name: Tomato Soup

Record Successfully Updated!!!!

Recipie ID: 8Recipie Name: Burger

Recipie ID: 3Recipie Name: Cucumber Salad

Recipie ID: 10Recipie Name: Garlic Bread

Recipie ID: 5Recipie Name: Grilled Cheeze

Recipie ID: 6Recipie Name: Momos

Recipie ID: 4Recipie Name: My Recipie

Recipie ID: 9Recipie Name: Pizza

Recipie ID: 7Recipie Name: Spring Roll

Recipie ID: 1Recipie Name: Tacos

Recipie ID: 2Recipie Name: Tomato Soup
```