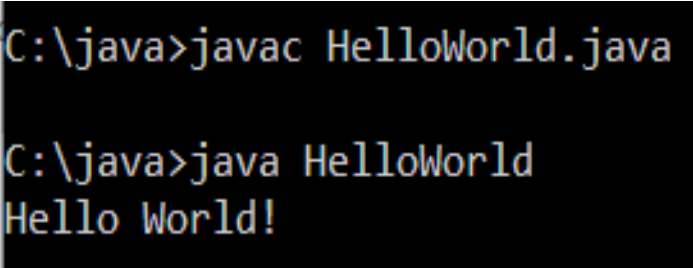


1. Java Program to print "Hello World".

```
class HelloWorld  
{  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Output:



```
C:\java>javac HelloWorld.java  
  
C:\java>java HelloWorld  
Hello World!
```

2. Write a program to check a number is even or odd.

```
import java.util.Scanner;

public class OddEven{

    public static void main(String[] args) {

        int number;

        Scanner myObj = new Scanner(System.in);

        System.out.print("Enter a Number to check odd or even: ");

        number = myObj.nextInt();

        if (number == 0)

            {

                System.out. println ("The Number is 0 ");

            }

            else if(number % 2 == 0){

                System.out. println ("The Number is even ");

            }

            else {

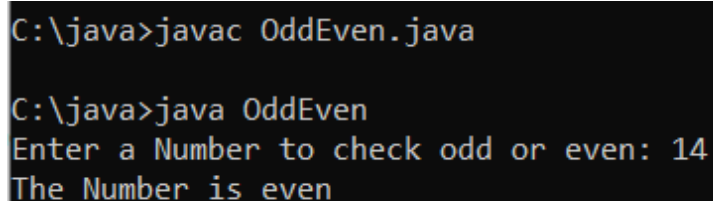
                System.out. println ("The Number is odd ");

            }

        }

    }
```

Output:



```
C:\java>javac OddEven.java

C:\java>java OddEven
Enter a Number to check odd or even: 14
The Number is even
```

3. Java program of ATM.

```
import java.util.Scanner;

public class ATM
{
    public static void main(String args[] )
    {
        int balance = 100000, withdraw, deposit;
        Scanner sc = new Scanner(System.in);
        while(true)
        {
            System.out.println("Automated Teller Machine");
            System.out.println("Choose 1 for Withdraw");
            System.out.println("Choose 2 for Deposit");
            System.out.println("Choose 3 for Check Balance");
            System.out.println("Choose 4 for EXIT");
            System.out.print("Choose the operation you want to perform:");

            int choice = sc.nextInt();
            switch(choice)
            {
                case 1:
                    System.out.print("Enter money to be withdrawn:");
                    withdraw = sc.nextInt();
                    if(balance >= withdraw)
                    {
                        balance = balance - withdraw;
                        System.out.println("Please collect your money");
                    }
                }
            }
        }
```

```
        else
        {
System.out.println("Insufficient Balance");
        }
System.out.println("");
        break;

        case 2:
System.out.print("Enter money to be deposited:");
        deposit = sc.nextInt();
        balance = balance + deposit;
System.out.println("Your Money has been successfully deposite");
System.out.println("");
        break;

        case 3:
System.out.println("Balance : "+balance);
System.out.println("");
        break;

        case 4:
System.exit(0);
        }
    }
}
```

Output:

```
C:\java>javac ATM.java

C:\java>java ATM
Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:1
Enter money to be withdrawn:4000
Please collect your money

Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:2
Enter money to be deposited:1000
Your Money has been successfully deposite

Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:3
Balance : 97000

Automated Teller Machine
Choose 1 for Withdraw
Choose 2 for Deposit
Choose 3 for Check Balance
Choose 4 for EXIT
Choose the operation you want to perform:4
```

4. Write a program in java to check valid password.

```
import java.util.Scanner;

public class PassWord {

    public static final int PASSWORD_LENGTH = 8;

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.print(

            "1. A password must have at least eight characters.\n" +

            "2. A password consists of only letters and digits.\n" +

            "3. A password must contain at least two digits \n" +

            "Input a password : ");

        String s = input.nextLine();

        if (is_Valid_Password(s)) {

            System.out.println("Password is valid: " + s);

        } else {

            System.out.println("Not a valid password: " + s);

        }

    }

    public static boolean is_Valid_Password(String password) {

        if (password.length() < PASSWORD_LENGTH) return false;

        int charCount = 0;

        int numCount = 0;

        for (int i = 0; i < password.length(); i++) {

            char ch = password.charAt(i);

            if (is_Numeric(ch)) numCount++;

            else if (is_Letter(ch)) charCount++;

            else return false;

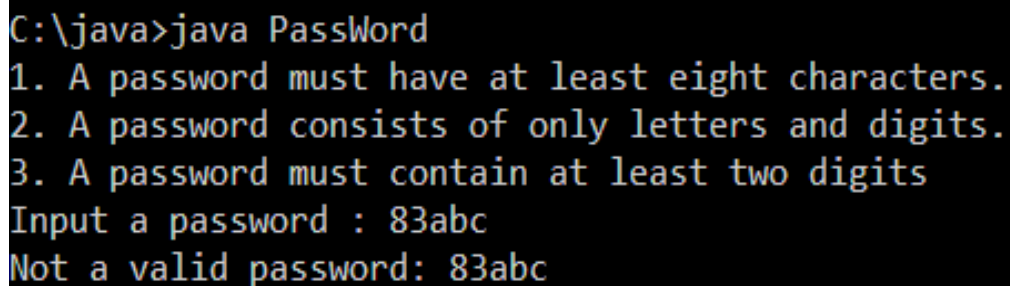
        }

    }

}
```

```
    }  
    return (charCount>= 2 &&numCount>= 2);  
}  
  
public static boolean is_Letter(char ch) {  
    ch = Character.toUpperCase(ch);  
    return (ch>= 'A' &&ch<= 'Z');  
}  
public static boolean is_Numeric(char ch) {  
    return (ch>= '0' &&ch<= '9');  
}  
}
```

Output:

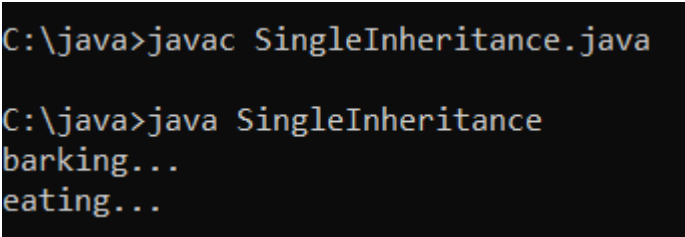


```
C:\java>java Password  
1. A password must have at least eight characters.  
2. A password consists of only letters and digits.  
3. A password must contain at least two digits  
Input a password : 83abc  
Not a valid password: 83abc
```

5. Write a program of single inheritance.

```
class Animal
{
void eat()
{
System.out.println("eating...");
}
}
class Dog extends Animal
{
void bark(){
System.out.println("barking...");
}
}
class SingleInheritance
{
public static void main(String args[])
{
Dog d=new Dog();
d.bark();
d.eat();
}
}
```

Output:



```
C:\java>javac SingleInheritance.java

C:\java>java SingleInheritance
barking...
eating...
```


6. Write a program of multiple inheritance.

```
class A
{
    public void perform()
    {
        System.out.println("Hello.. from Super Class");
    }
}

class B extends A {
}

public class Main
{
    public static void main(String[] args)
    {
        B obj = new B();
        obj.perform();
    }
}
```

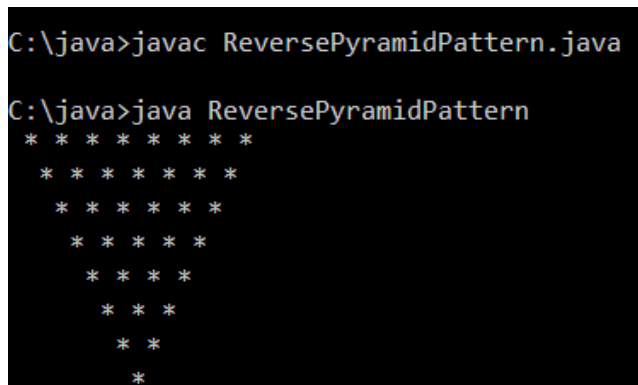
Output:

```
C:\SY BCA\JAVA>java Multilevel
The animal is eating
The dog is barking
The puppy is playing
```

7. Write a program to print pyramid.

```
public class ReversePyramidPattern
{
    public static void main(String[] args)
    {
        int rows=8;
        for (int i= 0; i<= rows-1; i++)
        {
            for (int j=0; j<=i; j++)
            {
                System.out.print(" ");
            }
            for (int k=0; k<=rows-1-i; k++)
            {
                System.out.print("*" + " ");
            }
            System.out.println();
        }
    }
}
```

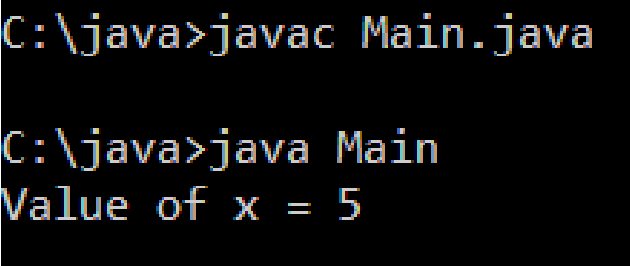
Output:



```
C:\java>javac ReversePyramidPattern.java
C:\java>java ReversePyramidPattern
      * * * * * * *
     * * * * * *
    * * * * *
   * * * *
  * * *
 * *
*
```

8. Write a program using "This" keyword.

```
public class Main
{
    int x;
    public Main(int x)
    {
        this.x = x;
    }
    public static void main(String[] args)
    {
        Main myObj = new Main(5);
        System.out.println("Value of x = " + myObj.x);
    }
}
```

Output:

```
C:\java>javac Main.java

C:\java>java Main
Value of x = 5
```

9. Write a program of if else in java.

```
public class IfElse
{
    public static void main(String[] args)
    {
        int number=13;
        if(number%2==0)
        {
            System.out.println("even number");
        }
        Else
        {
            System.out.println("odd number");
        }
    }
}
```

Output:

```
C:\java>javac IfElse.java
C:\java>java IfElse
odd number
```

10. Write a program of String method.

```
public class Stringoperation
{
    public static void main(String ar[])
    {
        String s="Hello World";
        System.out.println(s.toUpperCase());
        System.out.println(s.toLowerCase());
        System.out.println(s);
    }
}
```

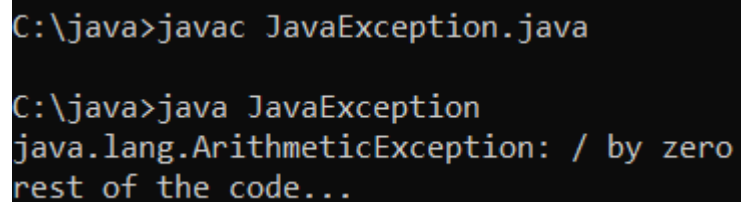
Output:

```
The length of the string is: 13
The index of the first 'o' is: 4
The substring is: world
The string in uppercase is: HELLO, WORLD!
```

11. Write a program of exception handling.

```
public class JavaException
{
    public static void main(String args[])
    {
        Try
        {
            int data=100/0;
        }
        catch(ArithmeticException e)
        {
            System.out.println(e);
        }
        System.out.println("rest of the code...");
    }
}
```

Output:

A screenshot of a Windows command prompt window with a black background and white text. It shows the compilation and execution of a Java program. The first line is 'C:\java>javac JavaException.java'. The second line is 'C:\java>java JavaException'. The output of the program is displayed on the next two lines: 'java.lang.ArithmeticException: / by zero' and 'rest of the code...'.

```
C:\java>javac JavaException.java

C:\java>java JavaException
java.lang.ArithmeticException: / by zero
rest of the code...
```

12. Write a program of threading in java.

```
class Multi extends Thread
{
    public void run()
    {
        System.out.println("thread is running...");
    }

    public static void main(String args[])
    {
        Multi t1=new Multi();
        t1.start();
    }
}
```

Output:

```
Thread started running..
Sum of two numbers is: 22
```

13. Write a program of multi-threading in java.

```
class TestMultitasking1 extends Thread
{
    public void run()
    {
        System.out.println("khushbu");
    }

    public static void main(String args[])
    {
        TestMultitasking1 t1=new TestMultitasking1();
        TestMultitasking1 t2=new TestMultitasking1();
        TestMultitasking1 t3=new TestMultitasking1();

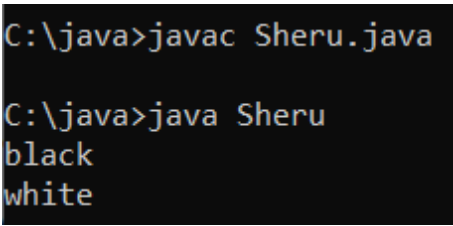
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output:

```
Thread Thread-0 is running
Thread Thread-1 is running
```


14. Write a program using "Super" keyword.

```
class Animal
{
String color="white";
}
class Dog extends Animal
{
String color="black";
void printColor()
{
System.out.println(color);
System.out.println(super.color);
}
}
class Sheru
{
public static void main(String args[])
{
Dog d=new Dog();
d.printColor();
}
}
```

Output:

```
C:\java>javac Sheru.java
C:\java>java Sheru
black
white
```

15. Write a program of single linked list.

Algorithm:

1. Create a class Node which has two attributes: data and next. Next is a pointer to the next node.
2. Create another class which has two attributes: head and tail.
3. addNode() will add a new node to the list:
 - a. Create a new node.
 - b. It first checks, whether the head is equal to null which means the list is empty.
 - c. If the list is empty, both head and tail will point to the newly added node.
 - d. If the list is not empty, the new node will be added to end of the list such that tail's next will point to the newly added node. This new node will become the new tail of the list.
4. display() will display the nodes present in the list:
 - a. Define a node current which initially points to the head of the list.
 - b. Traverse through the list till current points to null.
 - c. Display each node by making current to point to node next to it in each iteration.

Program:

```
public class LinkedList {  
    Node head;  
    static class Node  
    {  
        int data;  
        Node next;  
        Node(int d)  
        {  
            data = d;  
            next=null;  
        }  
    }  
}
```

```
    }

    /* This function prints contents of the linked list starting from head */

    public void display()
    {
        Node n = head;
        while (n != null)

        {
            System.out.print(n.data+" \n");
            n = n.next;
        }
    }

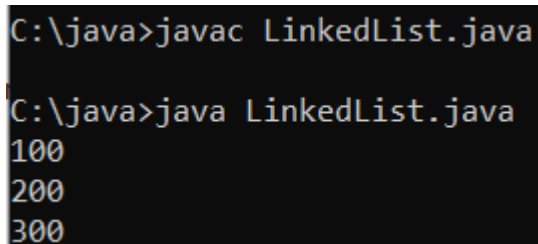
    /* method to create a simple linked list with 3 nodes*/

    public static void main(String[] args)
    {
        /* Start with the empty list. */

        LinkedList list = new LinkedList();
        list.head = new Node(100);
        Node second= new Node(200);
        Node third = new Node(300);

        list.head.next = second;
        second.next = third;
        list.display();
    }
}
```

Output:



```
C:\java>javac LinkedList.java
C:\java>java LinkedList.java
100
200
300
```

16. Write a program of circular linked list.

Algorithm:

1. Define a Node class which represents a node in the list. It has two properties data and next which will point to the next node.
2. Define another class for creating the circular linked list and it has two nodes: head and tail. It has two methods: add() and display() .
3. add() will add the node to the list:
 - a. It first checks whether the head is null, then it will insert the node as the head.
 - b. Both head and tail will point to the newly added node.
 - c. If the head is not null, the new node will be the new tail, and the new tail will point to the head as it is a circular linked list.
4. display() will show all the nodes present in the list.
 - a. Define a new node 'current' that will point to the head.
 - b. Print current.data till current will points to head
 - c. Current will point to the next node in the list in each iteration.

Program:

```
public class CreateList
{
    public class Node
    {
        int data;
        Node next;
        public Node(int data)
        {
            this.data = data;
        }
    }

    public Node head = null;
    public Node tail = null;
```

```
    public void add(int data)
    {
        Node newNode = new Node(data);
        if(head == null)
        {
            head = newNode;
            tail = newNode;
            newNode.next = head;
        }
        else
        {
            tail.next = newNode;
            tail = newNode;
            tail.next = head;
        }
    }

    public void display()
    {
        Node current = head;
        if(head == null)
        {
            System.out.println("List is empty");
        }
        Else
        {
            System.out.println("Nodes of the circular linked list: ");
            do
            {
                System.out.print(" "+ current.data);
                current = current.next;
            }
            while(current != head);
            System.out.println();
        }
    }

    public static void main(String[] args)
```

```
{  
CreateList cl = new CreateList();  
cl.add(1);  
cl.add(2);  
cl.add(3);  
cl.add(4);  
cl.display();  
}  
}
```

Output:

```
C:\java>javac CreateList.java  
  
C:\java>java CreateList.java  
Nodes of the circular linked list:  
1 2 3 4
```

17. Draw an smiley in java using applet.

```
import java.applet.Applet;

import java.awt.*;

public class Smiley extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.yellow);
        g.fillOval(20,20,150,150); // For face

        g.setColor(Color.black);
        g.fillOval(50,60,15,25); // Left Eye
        g.fillOval(120,60,15,25); // Right Eye

        int x[] = {95,85,106,95};
        int y[] = {85,104,104,85};

        g.drawArc(55,95,78,50,0,-180); // Smile
        g.drawLine(50,126,60,116); // Smile arc1
        g.drawLine(128,115,139,126); // Smile arc2

    }

}

/* <applet code="Smiley.class" width="200" height="200">

    </applet>

*/
```

Output:

18. Write a program in java, applet using parameters.

```
import java.awt.*;
import java.applet.*;
public class MyApplet extends Applet
{
    String n;
    String a;
    public void init()
    {
        n = getParameter("name");
        a = getParameter("age");
    }
    public void paint(Graphics g)
    {
        g.drawString("Name is: " + n, 20, 20);
        g.drawString("Age is: " + a, 20, 40);
    }
}
/*
<applet code="MyApplet" height="300" width="500">
    <param name="name" value="Sumit" />
    <param name="age" value="20" />
</applet>
*/
```

Output:

```
Name is: Sumit  
Age is :18
```