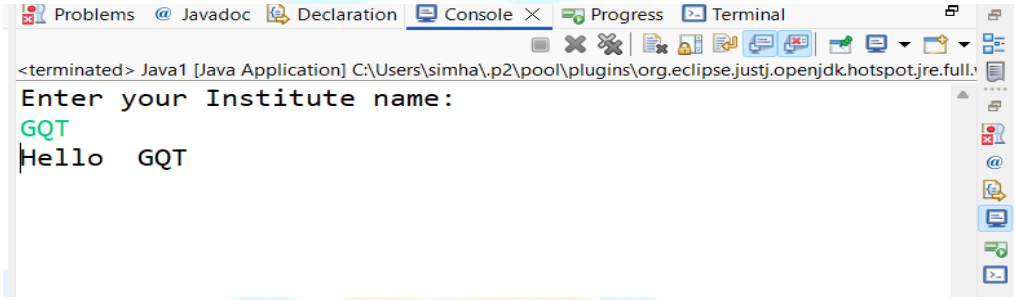
Core Java Programs

Introduction:

1) Write a java program to print your name.

```
package Introduction;
import java.util.*;
public class Program1 {
   public static void main(String[] args) {
      System.out.println("Enter your Institute
      name");
    Scanner s=new Scanner(System.in);
   String name=s.next();
      System.out.println("Hello "+name);
   }
}
```

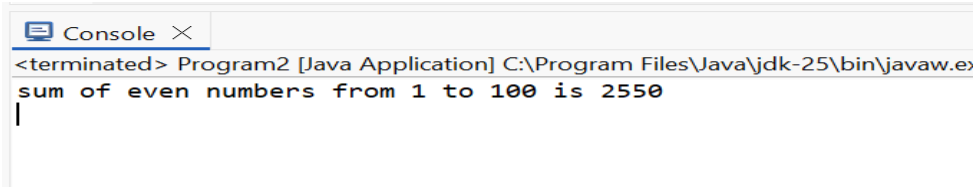
Output:



2) Write a program to print the sum of all even numbers from 1 to 100.

```
// program to print sum of even numbers between 1 to 100
public class Evensum {
   public static void main(String[] args) {
      int sum=0;
      for(int i=1;i<=100;i++){
         if(i%2==0){
            sum+=i;          }
      }
      System.out.println("sum of even numbers from 1 to 100 is "+sum);
   }
}
```

Output:

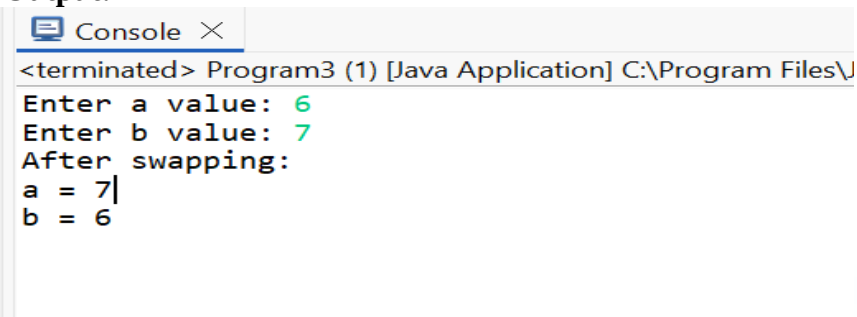


3) Write a program to swap two numbers without using a temporary variable.

```
import java.util.*;
public class Program3 {
        public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a value: ");
        int a = sc.nextInt();
        System.out.print("Enter b value: ");
        int b = sc.nextInt();
        a = a + b;
        b = a - b;
        a = a - b;
        System.out.println("After swapping:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
          }
    }
```

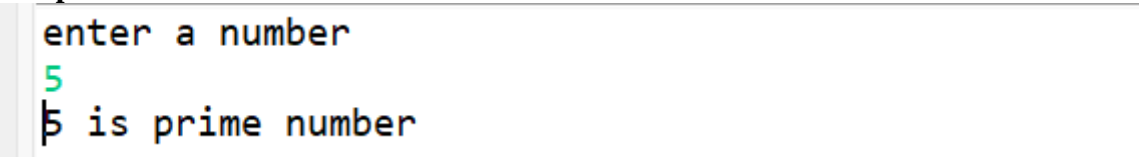
Output:



4) Write java program to check given number is prime or not.

```
    import java.util.*;
    public class Program4 {
        public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("enter a number ");
        int a=s.nextInt();
        int count=0;
        for(int i=1;i<=a;i++){
          if(a%i==0){
            count++;          }
          if(count>2){     break;          }
        }
        if(count>2){
          System.out.print(a+" is not prime number");
        }
        else{
          System.out.println(a+" is prime number");
        }
    }
    }
```

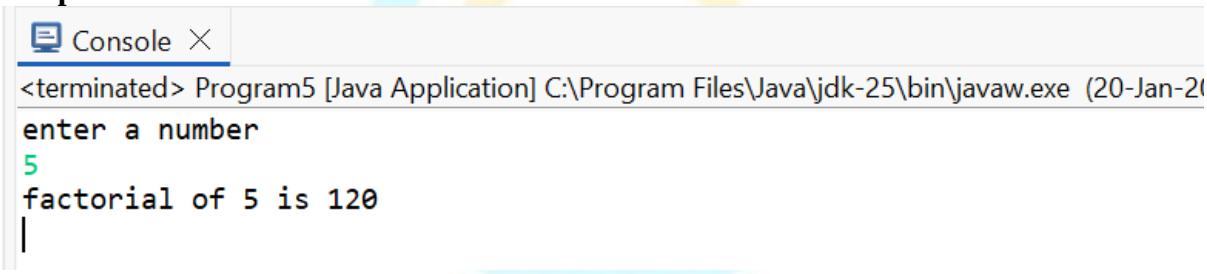
Output:



5) **write a program to find factorial of a number using recursion.**

```
//program to find factorial of a number
import java.util.*;
public class Program5 {
        public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("enter a number");
    int n=s.nextInt();
    System.out.println("factorial of "+n+" is "+ factorial(n));
  }
  public static int  factorial(int n){
    if(n<=1){
      return 1;
    }
    else{
      return n*factorial(n-1);
    }
  }
}
```

Output:



6) **Write a program to find the roots of a quadratic equation.**

```
package Introduction;
import java.util.*;
public class Program6 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter coefficients a, b and c:");
    double a = sc.nextDouble();
    double b = sc.nextDouble();
    double c = sc.nextDouble();
    double discriminant = b * b - 4 * a * c;
    if (discriminant > 0) {
      double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
      double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
      System.out.println("Two distinct real roots:");
      System.out.println("Root 1 = " + root1);
      System.out.println("Root 2 = " + root2);
    }
    else if (discriminant == 0) {
      double root = -b / (2 * a);
      System.out.println("Two equal real roots:");
      System.out.println("Root = " + root);
    }
```

```
            else {
                double realPart = -b / (2 * a);
                double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
                System.out.println("Complex roots:");
                System.out.println("Root 1 = " + realPart + " + " + imaginaryPart + "i");
                System.out.println("Root 2 = " + realPart + " - " + imaginaryPart + "i");
            }
        }
    }
```

Output:

```
<terminated> Program6 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
Enter coefficients a, b and c:
1 5 6
Two distinct real roots:
Root 1 = -2.0
Root 2 = -3.0
```

7) Write a program to find area of triangle.

```
import java.util.*;
public class Program7 {
        public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("Enter base of the triangle:");
    double base = s.nextDouble();
    System.out.println("Enter height of the triangle:");
    double height = s.nextDouble();
    double area = 0.5 * base * height;
    System.out.println("Area of the triangle = " + area);
        }
}
```

Output:

```
Console
<terminated> Program7 (1) [Java Application] C:\Program Files\Java\jdk-25\bi
Enter base of the triangle:
5
Enter height of the triangle:
7
Area of the triangle = 17.5
```

8) Write a program to print fibanacci series up to given number of terms.

```
import java.util.*;
public class Program8 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
                System.out.println("Enter a number");
            int n=s.nextInt();
            int first=0;
            int second=1;
```

```
                System.out.print(first+" "+second+" ");
                for(int i=1;i<=n-2;i++){
                    int next=first+second;
                    first=second;
                    second=next;
                    System.out.print(next+" ");
                }
            }
        }
```

Output:

```
<terminated> Program8 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (20-Jan-2026, 9:2
Enter a number
6
0 1 1 2 3 5
```

9) Write a program to find second largest element in an array.

```
// Second largest element
import java.util.*;
public class SecondLargest {
    public static void main(String[] args) {
        Scanner  s=new Scanner(System.in);
        System.out.println("Enter array size");
        int n=s.nextInt();
        int array[]=new int[n];
        System.out.println("Enter elemnts");
        for (int i= 0; i<n; i++) {
            array[i]=s.nextInt();
        }
        int f=Integer.MIN_VALUE;
        int sc=Integer.MIN_VALUE;
        for(int i=0;i<n;i++){
            if(array[i]>f){
                sc=f;
                f=array[i];
            }
            else if(array[i]>sc && array[i]!=f){
                sc=array[i];
            }
        }
        System.out.println("Secnd Largest elemnt "+ sc);
    }   }
```

Output:

```
Console
<terminated> Program9 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (20-Jan-2026, 9:27:57
Enter array size
5
Enter elemnts
3 5 1 7 6
Secnd Largest elemnt 6
```

10) Write a program to reverse a string.

```
//program to reverseing a string
import java.util.*;
public class StringReverse {
   public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      String str=s.nextLine();
      StringBuffer st=new StringBuffer(str);
      st.reverse();
      String sb=st.toString();
      System.out.println("Reversed string "+sb);
   }
}
```

Output:

```
<terminated> Program10 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (20-Jan-2026, 9:
enter a string
Global quest technologies
Reversed string seigolonhcet tseuq labolG
```

11) write a program to check a string palindrome or not.

```
//program to check string is pallindrome or not
import java.util.*;
public class Pallindrome {
   public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      String str=s.nextLine();
      String temp=str;
      StringBuffer st=new StringBuffer(str);
      st.reverse();
      String sb=st.toString();
      System.out.println("Reversed string "+sb);
      if(sb.equals(temp)){
         System.out.println("String is palindrome");
      }
      else{
         System.out.println("String is not pallindrome");
      }
   }
}
```

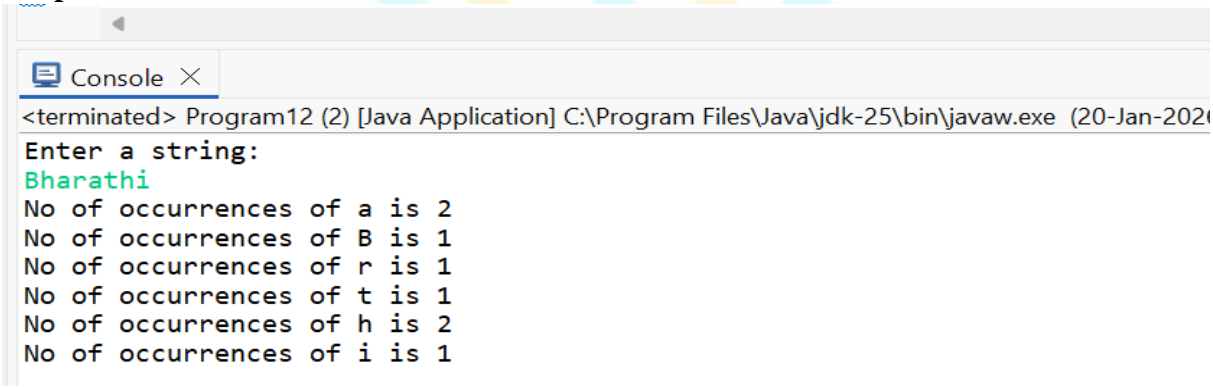
Output:

```
Console
<terminated> Program11 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (20-Jan-2026, 9:32:07 p
Enter a string
MADAM
Reversed string MADAM
This string is palindrome
```

12) Write a program to count the no of occurences of a character in a string.

```
// Program to count number of occurrences in a string
import java.util.*;
public class Numberofoccurences {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter a string:");
      String str = s.nextLine();
      HashMap<Character, Integer> map = new HashMap<>();
      for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);
         if (map.containsKey(ch)) {
            map.put(ch, map.get(ch) + 1);
         } else {
            map.put(ch, 1);
         }
      }
      for (char ch : map.keySet()) {
         System.out.println("No of occurrences of " + ch + " is " + map.get(ch));
      }
   }}
```

Output:



13) Write a program to find the GCD of two numbers.

```
import java.util.*;
public class Program13 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
            System.out.println("Enter two numbers");
            int n=s.nextInt();
            int m=s.nextInt();
            while (m!= 0) {
               int temp = m;
               m = n % m;
               n = temp;
            }
            System.out.println("GCD = " + n);
        }}
```

Output:

```
<terminated> Program13 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
Enter two numbers
9 6
GCD = 3
```

14) Write a program to convert decimal number to binary.

```
import java.util.*;
public class Program14 {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("enter a number ");
            int num = sc.nextInt();
            String binary = "";
            while (num > 0) {
               binary = (num % 2) + binary;
               num = num / 2;
            }
            System.out.println("Binary = " + binary);
        }
}
```

Output:

```
Console
<terminated> Program14 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (20-Jan-2
enter a number
10
Binary = 1010
```

15) Write a program to calculate the sum of digits of a given number.

```
import java.util.*;
public class Program15 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
            System.out.println("enter number ");
            int n=s.nextInt();
            int sum=0;
            while(n>0){
               sum+=n%10;
               n=n/10;
            }
            System.out.println("sum of the digits is "+sum);
        }
```

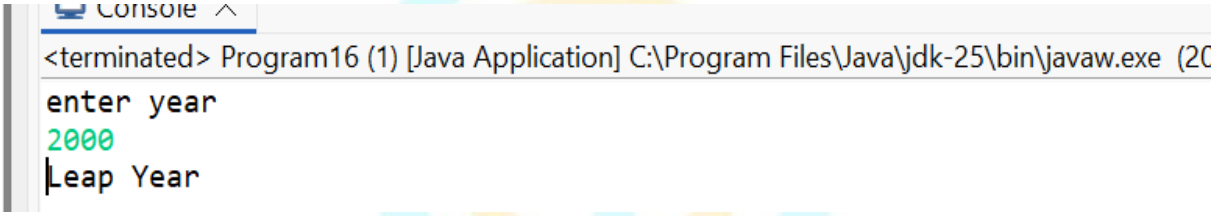
Output:

```
<terminated> Program15 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (20-Jan
enter number
685
sum of the digits is 19
```

16) Write a program to check given year is leap year or not.

```
package Introduction;
import java.util.*;
public class Program16 {
        public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("enter year");
    int n=s.nextInt();
    if((n%400==0)||(n%4==0&&!(n%100==0))){
       System.out.println("Leap Year");
    }
    else{
       System.out.println("Not a leap year");
    }
       }
}
```

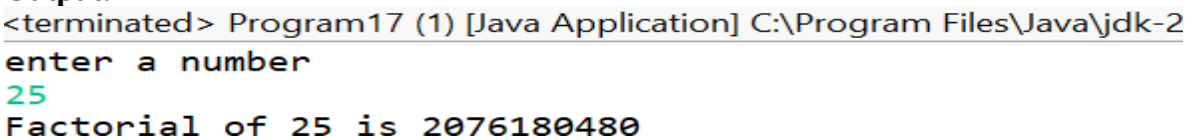
Output:



17) Write a program to find the factorial of a number using iteration.

```
package Introduction;
import java.util.*;
public class Program17 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
    System.out.println("enter a number");
    int n=s.nextInt();
    int fact=1;
    for(int i=1;i<=n;i++){
       fact=fact*i;
    }
    System.out.println("Factorial of "+n+" is "+fact);
       }}
```

Output:



18) Write a program to print the pascal's triangle.

```
package Introduction;
import java.util.*;
public class Program18 {
```

```
    public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
for (int i = 0; i < n; i++) {
    int num = 1;
    for (int space = 1; space <= n - i; space++) {
        System.out.print(" ");
    }
    for (int j = 0; j <= i; j++) {
        System.out.print(num + " ");
        num = num * (i - j) / (j + 1);
    }
    System.out.println();
}
    }}
```

Output:

```
<terminated> Program18 (1) [Java Application] C:\Program Files\Java\jdk-25\bin
5
      1
    1 1
  1 2 1
1 3 3 1
1 4 6 4 1
```

19) Write a program to check if a number is Armstrong or not.

```
import java.util.*;
public class Program19 {
        public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter a number ");
int num = sc.nextInt();
int original = num;
int sum = 0;
int digits = String.valueOf(num).length();
while (num > 0) {
    int digit = num % 10;
    sum += Math.pow(digit, digits);
    num /= 10;
}
if (sum == original) {
    System.out.println("Armstrong Number");
} else {
    System.out.println("Not an Armstrong Number");
}  }}
```

Output:

```
<terminated> Program19 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (20-Jan-2026,
Enter a number
153
Armstrong Number
```

20) Write a program to find the area and perimeter of a circle.

```
package Introduction;
import java.util.*;
public class Program20 {
	public static void main(String[] args) {
		Scanner s=new Scanner(System.in);
		System.out.println("enter radius");
	 double r = s.nextDouble();
	double pi = 3.14159;
	double area = pi * r * r;
	double perimeter = 2 * pi * r;
	System.out.println("Area of Circle = " + area);
	System.out.println("Perimeter of Circle = " + perimeter);
	    }
}
```

Output:

```
<terminated> Program20 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.ex
enter radius
6
Area of Circle = 113.09723999999999
Perimeter of Circle = 37.699079999999995
```

21) Write a program to print the multiplication table of a given number.

```
import java.util.*;
public class Program21 {
	public static void main(String[] args) {
		Scanner s=new Scanner(System.in);
	   System.out.println("enter a number");
	   int n=s.nextInt();
	   for(int i=1;i<=10;i++){
	     System.out.println(n+" * "+i+" = "+(n*i));
	   }
	}}
```

Output:

```
<terminated> Program21 (1) [Java Application] C:\Program Files\Jav
5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

22) Write a program to find the sum of all elements in an array.

```
package Introduction;
import java.util.*;
```

```
public class Program22 {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
            System.out.println("enter array size");
            int n = sc.nextInt();
            int[] arr = new int[n];
            int sum = 0;
            for (int i = 0; i < n; i++) {
               arr[i] = sc.nextInt();
               sum += arr[i];
            }
            System.out.println("Sum of array = " + sum);    }}
```

Output:

```
<terminated> Program22 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.ex
enter array size
5
1 2 3 4 5
Sum of array = 15
```

23) Write a program to check if a given number is a perfect number.

```
package Introduction;
import java.util.*;
public class Program23 {
        public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num = sc.nextInt();
      int sum = 0;
      for (int i = 1; i <= num / 2; i++) {
         if (num % i == 0) {
            sum += i;
         }
      }
      if (sum == num) {
         System.out.println("Perfect Number");
      } else {
         System.out.println("Not a Perfect Number");
      }
         }
```

Output:

```
<terminated> Program23 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (2
144
Not a Perfect Number
```

```
<terminated> Program23 [Java Application] C:\Program Files\Java\jdk-25\bin\ja
28
Perfect Number
```

24) Write a program to find the ASCII value of a character.

```
package Introduction;
import java.util.*;
public class Program24 {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter a character");
      char ch = sc.next().charAt(0);
      int ascii = ch;
      System.out.println("ASCII value of " + ch + " = " + ascii);
        }
}
```

Output:

```
<terminated> Program24 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.e
Enter a character
g
ASCII value of g = 103
```

25) Write a program to calculate the power if a number.

```
package Introduction;
import java.util.*;
public class Program25 {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.println("enter the base ");
            int base = sc.nextInt();
            System.out.println("enter the exponent");
            int exp = sc.nextInt();
            long result = 1;
            for (int i = 1; i <= exp; i++) {
               result *= base;
            }
            System.out.println("Result = " + result);}}
```

Output:

```
<terminated> Program25 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (20-Jar
enter the base
6
enter the exponent
3
Result = 216
```

OPERATORS:

1) Write a program to perform arithmetic operators on two numbers.

```
package Operators;
import java.util.*;
public class Program1 {
        public static void main(String[] args) {
      System.out.println("enter two numbers ");
```

```
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        int m=s.nextInt();
        System.out.println("Addition is "+(n+m));
        System.out.println("substraction is "+(n-m));
        System.out.println("multiplication is "+(n*m));
        System.out.println("division is "+(n/m));
          }
}
```

OUTPUT:

```
<terminated> Program1 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (2
enter two numbers
5 7
Addition is 12
substraction is -2
multiplication is 35
division is 0
```

2) **Write a program to perform bitwise AND, OR, and XOR operations on two integers.**

```
package Operators;
import java.util.*;
public class Program2 {
        public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
     System.out.println("Enter first integer:");
    int a = sc.nextInt();
    System.out.println("Enter second integer:");
    int b = sc.nextInt();
    int andResult = a & b;
    int orResult  = a | b;
    int xorResult = a ^ b;
    System.out.println("Bitwise AND (a & b) = " + andResult);
    System.out.println("Bitwise OR  (a | b) = " + orResult);
    System.out.println("Bitwise XOR (a ^ b) = " + xorResult);
          }
}
```

OUTPUT:

```
<terminated> Program2 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.
Enter first integer:
6
Enter second integer:
9
Bitwise AND (a & b) = 0
Bitwise OR  (a | b) = 15
Bitwise XOR (a ^ b) = 15
```

3) Write a program to check whether a given number is positive, negative, or zero.

```
package Operators;
import java.util.*;
public class Program3 {
        public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("enter a number");
    int n=s.nextInt();
    if(n>0){
        System.out.println("Positive Number");
    }
    else if(n<0){
        System.out.println("Negative Number");
    }
    else{
        System.out.println("Zero");
    } }
```

OUTPUT:

```
<terminated> Program3 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-2026, 9:18:03 pm
enter a number
1
Positive Number
```

```
<terminated> Program3 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (2
enter a number
0
Zero
```

```
<terminated> Program3 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-2026, 9:18:29
enter a number
-1
Negative Number
```

4) Write a program to swap two numbers using bitwise XOR operator.

```
package Operators;
import java.util.*;
public class Program4 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
            System.out.println("Enter 2 values ");
            int n=s.nextInt();
            int m=s.nextInt();
            System.out.println("Before swap: n = " + n + ", m = " + m);
            n=n^m;
            m=n^m;
            n=n^m;
            System.out.println("n = "+n+" m = "+m);
        }}
```

OUTPUT:

```
<terminated> Program4 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-202
Enter 2 values
5 8
Before swap: n = 5, m = 8
n = 8 m = 5
```

5) Write a program to calculate the area of a circle using the radius entered by the user.

```
package Operators;
import java.util.*;
public class Program5 {
        public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("enter radius ");
   double r = s.nextDouble();
   double pi = 3.14159;
   double area = pi * r * r;
   System.out.println("Area of circle is "+ area);
        }
}
```

OUTPUT:

```
<terminated> Program5 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
enter radius
6
Area of circle is 113.09723999999999
```

6) Write a program to convert temperature from Fahrenheit to Celsius.

```
package Operators;
import java.util.*;
public class Program6 {
     public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("enter temperature in fahrenheit ");
      double t=s.nextDouble();
      double c=(t-32)*(5.0/9.0);
      System.out.println("temperature in celcius "+c);
        }
}
```

OUTPUT:

```
<terminated> Program6 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.ex
enter temperature in fahrenheit
68
temperature in celcius 20.0
```

7) Write a program to check if a given number is divisible by both 5 and 7.

```
package Operators;
import java.util.*;
public class Program7 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
      System.out.println("Enter a number");
      int n=s.nextInt();
      if((n%5==0) && (n%7==0)){
        System.out.println("Yes "+n+" divisible by both 5 and 7");
```

```
        }
        else{
           System.out.println("No "+n+" does not divisible by both 5 and 7");
        }
           }}
```

OUTPUT:

```
<terminated> Program7 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-2
Enter a number
35
Yes 35 divisible by both 5 and 7
```

8) Write a program to calculate the compound interest.

```
package Operators;
import java.util.*;
public class Program8 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
      System.out.println("enter pricipal amount, rate of interest, and time ");
      double p=s.nextDouble();
      double i=s.nextDouble();
      double t=s.nextDouble();
      double amount=p*Math.pow((1+(i/100)),t);
      double ci=amount-p;
      System.out.println("Compound interest "+ci);
        }
}
```

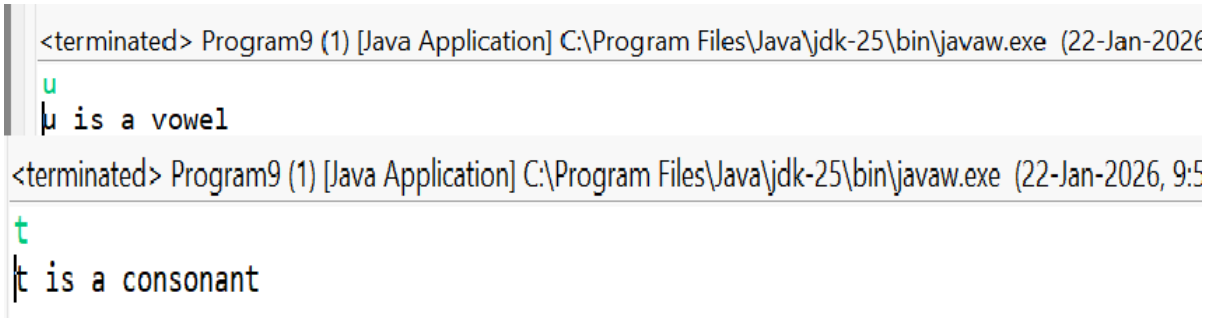
OUTPUT:

```
<terminated> Program8 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-2026, 9:49:39 pm – 9:
enter pricipal amount, rate of interest, and time
10000 20 4
Compound interest 10736.0
```

9) Write a program to check whether a given character is a vowel or consonant.

```
package Operators;
import java.util.*;
public class Program9 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
      char c=s.next().charAt(0);
      String vowel="aeiouAEIOU";
      if(vowel.contains(String.valueOf(c))){
         System.out.println(c+" is a vowel");
      }
      else{
         System.out.println(c+" is a consonant");
      }
         }
}
```

OUTPUT:



10) Write a program to find the maximum of three numbers using conditional operator.

```
package Operators;
import java.util.*;
public class Program10 {
        public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("enter three numbers ");
      int a=s.nextInt();
      int b=s.nextInt();
      int c=s.nextInt();
      if(a>b){
         if(a>c){
            System.out.println(a+" is greater ");
         }
         else{
            System.out.println(c+" is greater ");
         }
      }
      else if(b>c){
         System.out.println(b+" is greater ");
      }
      else{
         System.out.println(c+" is greater ");
      }
   }
}
```

OUTPUT:

11) Write a program to find the sum of digits of a number using while loop.

```
package Operators;
import java.util.*;
public class Program11 {
        public static void main(String[] args) {
```

```
        Scanner s=new Scanner(System.in);
    System.out.println("enter number ");
    int n=s.nextInt();
    int sum=0;
    while(n>0){
       sum+=n%10;
       n=n/10;
    }
    System.out.println("sum of the digits is "+sum);          }}
```

OUTPUT:

```
<terminated> Program11 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-202
enter number
256
sum of the digits is 13
```

12) Write a program to check whether a given number is palindrome or not using recursion.

```
package Operators;
import java.util.*;
public class Program12 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
                System.out.println("enter number ");
            int n=s.nextInt();
            int rev=pallindrome(n,0);
            if(rev==n){
               System.out.println("Pallindrome");
            }
            else{
               System.out.println("not pallindrome");
            }
        }
       static int pallindrome(int n,int rev){
          if (n == 0) return rev;
          return pallindrome(n / 10, rev * 10 + (n % 10));
        }
}
```

OUTPUT:

```
<terminated> Program12 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.e
enter number
262
Pallindrome
```

13) Write a program to check whether a given number is prime or not using for loop.

```
package Operators;
import java.util.*;
public class Program13 {
        public static void main(String[] args) {
```

```
        Scanner s=new Scanner(System.in);
    System.out.println("Enter a number ");
    int n=s.nextInt();
    int count=0;
    for(int i=1;i<=n;i++){
       if(5%i==0){
          count++;
       }
    }
    if(count<=2){
       System.out.println("It is a Prime number");
    }
    else{
       System.out.println("It is not a prime number");
    } }
  }
```

OUTPUT:

```
<terminated> Program13 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-2026, 10:00:40 pr
Enter a number
67
It is a Prime number
```

14) Write a program to find the factorial of a number using recursion.

```
package Operators;
import java.util.*;
public class Program14 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
      System.out.println("Enetr a number ");
      int n=s.nextInt();
      System.out.println("Factorial of "+n+" :"+Factorial(n));
    }
    static int Factorial(int n){
      if(n<=1){
         return 1;
      }
      else{
        return  n*Factorial(n-2);
      }
        }
}
```

OUTPUT:

```
Enetr a number
5
Factorial of 5 is  : 120
```

15) Write a program to calculate the power of a number using recursion.

```
package Operators;
import java.util.*;
public class Program15 {
   public static void main(String[] args) {
```

```
        Scanner s=new Scanner(System.in);
        System.out.println("Enter a base ");
      int n=s.nextInt();
      System.out.println("enter exponent");
      int m=s.nextInt();
      System.out.println("Result  "+power(n,m));
      }
      static int power(int base,int exp){
         if(exp==0){
            return 1;
         }
         else{
            return base*power(base,exp-1);
         }
      }
   }
```

OUTPUT:

```
<terminated> Program15 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (22-Jan-2026,
Enter a base
6
enter exponent
3
Result  216
```

16) Write a program to print the Fibonacci series using recursion.

```
package Operators;
import java.util.*;
public class Program166 {
        static int fibonacci(int n) {
           if (n == 0)
              return 0;
            if (n == 1)
              return 1;
           return fibonacci(n - 1) + fibonacci(n - 2);
        }
              public static void main(String[] args) {
                 Scanner sc = new Scanner(System.in);

                 System.out.println("Enter number of terms:");
                 int n = sc.nextInt();

                 System.out.println("Fibonacci Series:");
                 for (int i = 0; i < n; i++) {
                    System.out.print(fibonacci(i) + " ");
                 }    }}
```

OUTPUT:

```
<terminated> Program166 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (23-Ja
Enter number of terms:
5
Fibonacci Series:
0 1 1 2 3
```

17) Write a program to reverse a string using recursion.

```
package Operators;
import java.util.*;
public class Program17 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
      System.out.println("enter a string");
      String st=s.nextLine();
      System.out.println("Reversed string is "+stringReverse(st));
   }
   static String stringReverse(String str){
      if(str.isEmpty()){
         return str;
      }
      else{
         return stringReverse(str.substring(1))+str.charAt(0);
      }
         }
}
```

OUTPUT:

```
<terminated> Program17 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\java
enter a string
Bharathi
Reversed string is ihtarahB
```

18) Write a program to calculate the sum of natural numbers up to a given term.

```
package Operators;
import java.util.*;
public class Program18 {
   public static void main(String args[]){
      Scanner s=new Scanner(System.in);
      System.out.println("Enter a number");
      int n=s.nextInt();
      int sum=0;
      for(int i=1;i<=n;i++){
         sum+=i;
      }
      System.out.println("Sum of natural numbers is "+sum);
   }}
```

OUTPUT:

```
<terminated> Program18 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (23
Enter a number
9
Sum of natural numbers is 45
```

19) Write a program to check whether a given year is leap year or not using conditional operator.

```
package Operators;
import java.util.*;
```

```
public class Program19 {
        public static void main(String args[]) {
                Scanner s=new Scanner(System.in);
            System.out.println("Enter a year ");
            int n=s.nextInt();
            if((n%400==0)||((n%4==0)&&(n%100==0))){
               System.out.println("Given year is leap year");
            }
            else{
               System.out.println("Given year not q leap year");
            }
        }
}
```

OUTPUT:

```
Enter a year
2005
Given year not q leap year
```

20) Write a program to find the LCM (Least Common Multiple) of two numbers.

```
package Operators;
import java.util.*;
public class Program20 {
   public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("Enter two numbers ");
      int n=s.nextInt();
      int m=s.nextInt();
       int lcm = (n * m) / gcd(n, m);
      System.out.println("LCM = " + lcm);
   }
    static int gcd(int a, int b) {
      if (b == 0)
         return a;
      return gcd(b, a % b);
   }
}
```

OUTPUT:

```
Problems  Javadoc  Declaration  Console  Eclipse IDE
<terminated> Program20 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (23-Jan-2026, 9:50:10 pm
Enter two numbers
6 8
LCM = 24
```

21) Write a program to calculate the area of a triangle using Heron's formula.

```
package Operators;
import java.util.*;
public class Program21 {
   public static void main(String[] args) {
```

```
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter side a: ");
        double a = sc.nextDouble();
        System.out.print("Enter side b: ");
        double b = sc.nextDouble();
        System.out.print("Enter side c: ");
        double c = sc.nextDouble();
        double s = (a + b + c) / 2;
        double area = Math.sqrt(s * (s - a) * (s - b) * (s - c));
        System.out.println("Area of the triangle = " + area);

    }
}
```

OUTPUT:

```
<terminated> Program21 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (23-Jan-2026, 9:51:2
Enter side a: 6
Enter side b: 8
Enter side c: 3
Area of the triangle = 7.644442425710328
```

22) Write a program to find the sum of all even numbers between 1 to 100 using for loop.

```
package Operators;
import java.util.*;
public class Program22 {
        public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("printing numbers from 1 to 100");
    for(int i=1;i<=100;i++){
        System.out.print(i+" ");
    }
        }}
```

OUTPUT:

```
<terminated> Program22 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (23-Jan-2026, 9:52:53 pm – 9
printing numbers from 1 to 100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
```

23) Write a program to calculate the simple interest.

```
package Operators;
import java.util.*;
public class Program23 {
        public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("enter pricipal amount, year and rate");
    int a=s.nextInt();
    int y=s.nextInt();
    int r=s.nextInt();
    double SI=(a*y*r)/100;
    System.out.println("simple interest "+SI);        }
```

}

OUTPUT:

```
<terminated> Program23 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (23-
enter pricipal amount, year and rate
10000 5 20
simple interest 10000.0
```

24) Write a program to print the multiplication table of a given number using for loop.

```
package Operators;
import java.util.*;
public class Program24 {
	public static void main(String[] args) {
		Scanner s=new Scanner(System.in);
		  System.out.println("Enter a number");
		  int n=s.nextInt();
		  for(int i=1;i<=10;i++){
		     System.out.println(n+" * "+i+" = "+(n*i));
		  }
	}
}
```

OUTPUT:

```
Enter a number
5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

25) Write a program to check whether a given number is Armstrong or not using while loop.

```
package Operators;
import java.util.*;
public class Program25 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
	System.out.println("enter a number");
	int num = sc.nextInt();
	int original = num;
	int sum = 0;
	int digits = String.valueOf(num).length();
	while (num > 0) {
	   int digit = num % 10;
	   sum += Math.pow(digit, digits);
	   num /= 10;
	   }
```

```
        if (sum == original) {
           System.out.println("Armstrong Number");
           }
        else {
                System.out.println("Not an Armstrong Number");
                }
           }
    }
```

OUTPUT:

```
<terminated> Program25 (1) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
enter a number
153
Armstrong Number
```

Arrays:

1) Write a program to find the sum of all elements in an array.

```
package Arrays;
import java.util.*;
public class Program1 {
        public static void main(String[] args) {
Scanner s = new Scanner(System.in);
      System.out.println("Enter array size");
      int n = s.nextInt();
      int array[] = new int[n];
      int sum = 0;
      System.out.println("Enter array elements");
      for(int i = 0; i < n; i++){
         array[i] = s.nextInt();
         sum = sum + array[i];
      }
      System.out.println("Sum = " + sum);
         }
}
```

OUTPUT:

```
<terminated> Program1 (6) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 9:0
Enter array size
5
Enter array elements
1 2 3 4 5
Sum = 15
```

2) Write a program to find the largest and smallest elements in an array.

```
package Arrays;
import java.util.*;
public class Program2 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
```

```
                System.out.println("Enter array size ");
                int n=s.nextInt();
                int array[] = new int[n];
                System.out.println("Enter array elements:");
                for (int i = 0; i < n; i++) {
                   array[i] = s.nextInt();
                }
                 int largest = array[0];
                for (int i = 1; i < n; i++) {
                   if (array[i] > largest) {
                      largest = array[i];
                   }
                }
                System.out.println("Largest: " + largest);
                // Smallest element
                int smallest = array[0];
                for (int i = 1; i < n; i++) {
                   if (array[i] < smallest) {
                      smallest = array[i];
                   }
                }
                System.out.println("Smallest: " + smallest);
        }
    }
```

OUTPUT:

```
Enter array size
5
Enter array elements:
1 2 6 8 2
Largest: 8
Smallest: 1
```

3) Write a program to copy elements from one array to another.

```
package Arrays;
import java.util.Scanner;
public class Program3 {
        public static void main(String[] args) {
                 Scanner s=new Scanner(System.in);
                System.out.println("Enter array size ");
                int n=s.nextInt();
                int array[] = new int[n];
                System.out.println("Enter array elements:");
                for (int i = 0; i < n; i++) {
                   array[i] = s.nextInt();
                }
                int a2[]=new int[n];
                for(int i=0;i<n;i++) {
                   a2[i]=array[i];
                }
```

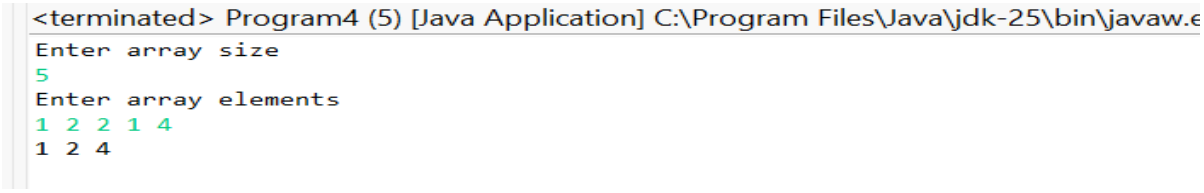
```
System.out.println("Array2 elements are");
for(int i=0;i<n;i++) {
   System.out.print(a2[i]+" ");
}
}}
```

OUTPUT:

4) Write a program to remove duplicate elements from an array.

```
package Arrays;
import java.util.*;
public class Program4 {
        public static void main(String[] args) {
                Scanner s = new Scanner(System.in);
      System.out.println("Enter array size");
      int n = s.nextInt();
      int array[] = new int[n];
      System.out.println("Enter array elements");
      for(int i = 0; i < n; i++){
         array[i] = s.nextInt();
      }
      int u[]=Arrays.stream(array).distinct().toArray();
      for(int i=0;i<u.length;i++){
         System.out.print(u[i]+" ");
      }
         }}
```

OUTPUT:



5) Write a program to reverse an array.

```
package Arrays;
import java.util.*;
public class Program5 {
        public static void main(String[] args) {
      Scanner s=new Scanner(System.in);
      System.out.println("Enter array size ");
      int n=s.nextInt();
      int array[] = new int[n];
      System.out.println("Enter array elements");
```

```
        for(int i = 0; i < n; i++){
           array[i] = s.nextInt();
        }
        for(int i=0;i<n/2;i++){
           int t=array[i];
           array[i]=array[n-1-i];
           array[n-1-i]=t;
        }
        System.out.println("Array after reversing ");
         for(int i = 0; i < n; i++){
           System.out.print(array[i]+" ");
        }
            }
    }
```

OUTPUT:

```
Enter array size
5
Enter array elements
1 2 3 4 5
Array after reversing
5 4 3 2 1
```

6) Write a program to sort an array in ascending and descending order.

```
package Arrays;
import java.util.*;
public class Program6 {
        public static void main(String[] args) {
       Scanner s= new Scanner(System.in);
       System.out.println("enter array size");
       int n=s.nextInt();
       int array[]=new int[n];
       for (int i = 0; i <n; i++) {
          array[i]=s.nextInt();
       }
       for(int i=1;i<n;i++){
             if(array[i]<array[i-1]){
               int t=array[i];
               array[i]=array[i-1];
               array[i-1]=t;
             }
       }
       System.out.println("Sorted array is ");
       for(int i=0;i<n;i++) {
          System.out.print(array[i]+" ");
       }
          }
}
```

OUTPUT:

```
<terminated> Program6 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 
enter array size
5
1 5 3 2 6
Sorted array is
1 3 2 5 6
```

7) Write a program to find the frequency of each element in an array.

```
package Arrays;
    import java.util.*;
    public class Program7 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
         System.out.println("enter array size ");
         int n=s.nextInt();
         int array[]=new int[n];
         System.out.println("Enter array elements");
         for(int i=0;i<n;i++){
            array[i]=s.nextInt();
         }
         Map <Integer,Integer> map=new HashMap<>();
         for(int value: array){
            map.put(value,map.getOrDefault(value,0)+1);
         }
         System.out.println("Frequency of each element is ");
         for(Map.Entry<Integer,Integer> entry: map.entrySet()){
            System.out.println(entry.getKey()+"->"+entry.getValue());
         }
        }
    }
```

OUTPUT:

```
<terminated> Program7 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe 
enter array size
5
Enter array elements
1 2 3 2 3
Frequency of each element is
1->1
2->2
3->2
```

8) Write a program to merge two sorted arrays.

```
package Arrays;
import java.util.*;
public class Program8 {
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
      System.out.println("Enter size of 2 arrays");
      int n=s.nextInt();
      int m=s.nextInt();
      int array[]=new int[n];
      System.out.println("Enter elements of array1");
      for(int i=0;i<n;i++){
         array[i]=s.nextInt();
```

```
        }
        int array2[]=new int[m];
        System.out.println("Enter elements of array2");
        for(int i=0;i<m;i++){
           array2[i]=s.nextInt();
        }
        int newarray[]=new int[n+m];
        System.arraycopy(array,0, newarray, 0, n);
        System.arraycopy(array2, 0, newarray, n, m);
        System.out.println("Resultant array is ");
        for(int i=0;i<newarray.length;i++){
           System.out.print(newarray[i]+" ");

        }
           }
    }
```

OUTPUT:

```
Enter size of 2 arrays
5
6
Enter elements of array1
1 2 3 4 5
Enter elements of array2
6 7 8 9 0 1
Resultant array is
1 2 3 4 5 6 7 8 9 0 1
```

9) Write a program to find the intersection of two arrays.

```
package Arrays;
import java.util.*;
public class Program9 {
        public static void main(String[] args) {
              Scanner s=new Scanner(System.in);
           System.out.println("enter array sizes");
           int n=s.nextInt();
           int m=s.nextInt();
           int array1[]=new int[n];
           int array2[]=new int[m];
           System.out.println("enter array1 elements ");
           for(int i=0;i<n;i++){
              array1[i]=s.nextInt();
           }
           System.out.println("enter array2 elements ");
           for(int i=0;i<m;i++){
              array2[i]=s.nextInt();
           }
           Set <Integer> set1=new HashSet<>();
           for(int v:array1){
              set1.add(v);
           }
           Set <Integer> intersection=new HashSet<>();
           for(int v:array2){
              if(set1.contains(v)){
```

```
                    intersection.add(v);
                }
            }
            int array[]=new int[intersection.size()];
            int i=0;
            for(int v:intersection){
                array[i++]=v;
            }
            System.out.println("Intersection of arrays :"+ Arrays.toString(array));
        }
    }
```

OUTPUT:

```
<terminated> Program9 (2) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jar
enter array sizes
5 6
enter array1 elements
12 4 5 6 2
enter array2 elements
7 8 4 5 3 2
Intersection of arrays :[2, 4, 5]
```

10) Write a program to check whether an array is palindrome or not.

```
package Arrays;
import java.util.Scanner;
public class Program10 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of array:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        boolean isPalindrome = true;
        for (int i = 0; i < n / 2; i++) {
            if (arr[i] != arr[n - 1 - i]) {
                isPalindrome = false;
                break;
            }
        }
        if (isPalindrome) {
            System.out.println("Array is Palindrome");
        } else {
            System.out.println("Array is NOT Palindrome");
        }   }}
```

OUTPUT:

```
<terminated> Program10 (4) [Java Application] C:\Progr
Enter size of array:
5
Enter array elements:
1 2 5 2 1
Array is Palindrome
```

11) Write a program to find the sum of all positive numbers in an array.

```
package Arrays;
import java.util.*;
public class Program11 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter size of array:");
		int n = sc.nextInt();
		int[] arr = new int[n];
		System.out.println("Enter array elements:");
		for (int i = 0; i < n; i++) {
		   arr[i] = sc.nextInt();
		}
		int sum = 0;
		for (int i = 0; i < n; i++) {
		   if (arr[i] > 0) {
		      sum = sum + arr[i];
		   }		}
		System.out.println("Sum of positive numbers: " + sum);
	   }
	}
```

OUTPUT:

```
<terminated> Program11 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
Enter size of array:
5
Enter array elements:
1 2 -7 4 -1
Sum of positive numbers: 7
```

12) Write a program to find the sum of all negative numbers in an array.

```
package Arrays;
import java.util.*;
public class Program12 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter size of array:");
		int n = sc.nextInt();
		int[] arr = new int[n];
		System.out.println("Enter array elements:");
		for (int i = 0; i < n; i++) {
		   arr[i] = sc.nextInt();
		}
		int sum = 0;
		for (int i = 0; i < n; i++) {
		   if (arr[i] < 0) {
		      sum = sum + arr[i];
		   }
		}
		System.out.println("Sum of negative numbers: " + sum);
	}
```

```
}
```

OUTPUT:

```
<terminated> Program12 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\ja
Enter size of array:
5
Enter array elements:
1 -2 -4 2 3
Sum of negative numbers: -6
```

13) Write a program to find the product of all elements in an array.

```
package Arrays;
import java.util.*;
public class Program13 {
        public static void main(String[] args) {
                Scanner s = new Scanner(System.in);
    System.out.println("Enter array size");
    int n = s.nextInt();
    int array[] = new int[n];
    int sum = 1;
    System.out.println("Enter array elements");
    for(int i = 0; i < n; i++){
       array[i] = s.nextInt();
       sum = sum * array[i];
    }
    System.out.println("Product = " + sum);
       }}
```

OUTPUT:

```
<terminated> Program13 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
Enter array size
5
Enter array elements
1 2 3 4 5
Product = 120
```

14) Write a program to find the second largest and second smallest elements in an array.

```
package Arrays;
import java.util.*;
public class Program14 {
        public static void main(String[] args) {
    Scanner  s=new Scanner(System.in);
    System.out.println("Enter array size");
    int n=s.nextInt();
    int array[]=new int[n];
    System.out.println("Enter elemnts");
    for (int i= 0; i<n; i++) {
       array[i]=s.nextInt();
    }
    int f=Integer.MIN_VALUE;
    int sc=Integer.MIN_VALUE;
    for(int i=0;i<n;i++){
       if(array[i]>f){
```

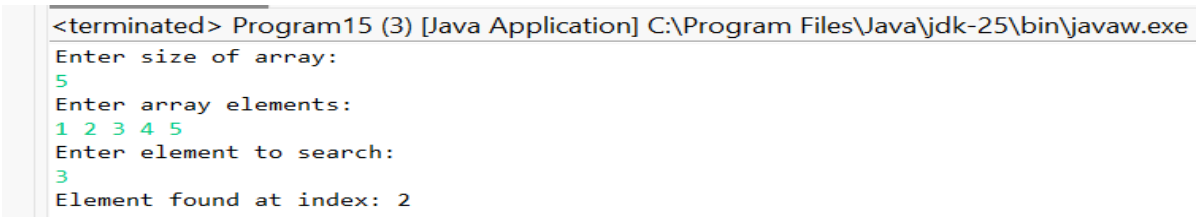
```
            sc=f;
            f=array[i];
        }
        else if(array[i]>sc && array[i]!=f){
            sc=array[i];
        }
    }
    System.out.println("Secnd Largest elemnt "+ sc);
    }}
```

OUTPUT:

15) Write a program to find the index of a given element in an array.

```
package Arrays;
import java.util.*;
public class Program15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of array:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter element to search:");
        int key = sc.nextInt();
        int index = -1;
        for (int i = 0; i < n; i++) {
            if (arr[i] == key) {
                index = i;
                break;
            }           }
        if (index != -1) {
            System.out.println("Element found at index: " + index);
        } else {
            System.out.println("Element not found in array");
        } }}
```

OUTPUT:



16) Write a program to rotate an array to the left or right.

```
package Arrays;
import java.util.*;
import java.util.Scanner;
public class Program16 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter size of array:");
      int n = sc.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter array elements:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
      System.out.println("Enter number of rotations:");
      int r = sc.nextInt();
      System.out.println("Enter direction (L for Left, R for Right):");
      char dir = sc.next().charAt(0);
      r = r % n;
      if (dir == 'L' || dir == 'l') {
         for (int i = 0; i < r; i++) {
            int first = arr[0];
            for (int j = 0; j < n - 1; j++) {
               arr[j] = arr[j + 1];
            }
            arr[n - 1] = first;
         }
      } else if (dir == 'R' || dir == 'r') {
         for (int i = 0; i < r; i++) {
            int last = arr[n - 1];
            for (int j = n - 1; j > 0; j--) {
               arr[j] = arr[j - 1];
            }
            arr[0] = last;
         }       }
      System.out.println("Rotated Array:");
      for (int i = 0; i < n; i++) {
         System.out.print(arr[i] + " ");
      }    }}
```

OUTPUT:

```
<terminated> Program16 (3) [Java Application] C:\Program Files\Java\jdk-
Enter size of array:
5
Enter array elements:
1 2 3 4 5
Enter number of rotations:
2
Enter direction (L for Left, R for Right):
L
Rotated Array:
3 4 5 1 2
```

17) Write a program to print the elements of a 2D array in spiral order.

```
package Arrays;
import java.util.Scanner;
public class Program17 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number of rows:");
      int r = sc.nextInt();
      System.out.println("Enter number of columns:");
      int c = sc.nextInt();
      int[][] arr = new int[r][c];
      System.out.println("Enter matrix elements:");
      for (int i = 0; i < r; i++) {
         for (int j = 0; j < c; j++) {
            arr[i][j] = sc.nextInt();
         }
      }
      int top = 0, bottom = r - 1;
      int left = 0, right = c - 1;
      System.out.println("Spiral Order:");
      while (top <= bottom && left <= right) {
         for (int i = left; i <= right; i++) {
            System.out.print(arr[top][i] + " ");
         }
         top++;
         for (int i = top; i <= bottom; i++) {
            System.out.print(arr[i][right] + " ");
         }
         right--;
         if (top <= bottom) {
            for (int i = right; i >= left; i--) {
               System.out.print(arr[bottom][i] + " ");
            }
            bottom--;
         }
        if (left <= right) {
            for (int i = bottom; i >= top; i--) {
               System.out.print(arr[i][left] + " ");
            }
            left++;
         }      }
   }
}
```

OUTPUT:

```
<terminated> Program17 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe
Enter number of rows:
3
Enter number of columns:
3
Enter matrix elements:
1 2 3 4 5 6 1 2 3
Spiral Order:
1 2 3 6 3 2 1 4 5
```

18) Write a program to check whether two arrays are equal or not.

```
package Arrays;
import java.util.Scanner;
public class Program18 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter size of arrays:");
      int n = sc.nextInt();
      int[] arr1 = new int[n];
      int[] arr2 = new int[n];
      System.out.println("Enter elements of first array:");
      for (int i = 0; i < n; i++) {
         arr1[i] = sc.nextInt();
      }
      System.out.println("Enter elements of second array:");
      for (int i = 0; i < n; i++) {
         arr2[i] = sc.nextInt();
      }
      boolean isEqual = true;
      for (int i = 0; i < n; i++) {
         if (arr1[i] != arr2[i]) {
            isEqual = false;
            break;
         }
      }
      if (isEqual) {
         System.out.println("Arrays are Equal");
      } else {
         System.out.println("Arrays are NOT Equal");
      }
   }
}
```

OUTPUT:

```
Enter size of arrays:
5
Enter elements of first array:
1 2 3 4 5
Enter elements of second array:
1 2 3 4 5 6
Arrays are Equal
```

19) Write a program to find the sum of elements in the upper triangle of a matrix.

```
package Arrays;
import java.util.Scanner;
public class Program19 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter order of square matrix:");
      int n = sc.nextInt();
      int[][] mat = new int[n][n];
```

```
            System.out.println("Enter matrix elements:");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    mat[i][j] = sc.nextInt();
                }
                System.out.println();
            }
            int sum = 0;
            for (int i = 0; i < n; i++) {
                for (int j = i; j < n; j++) {
                    sum = sum + mat[i][j];
                }
            }
            System.out.println("Sum of upper triangle elements: " + sum);
        }
    }
```

OUTPUT:

```
Enter order of square matrix:
3
Enter matrix elements:
1 2 3

4 5 6

1 2 3

Sum of upper triangle elements: 20
```

20) Write a program to find the sum of elements in the lower triangle of a matrix.

```
package Arrays;
import java.util.Scanner;
public class Program20 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter order of square matrix:");
        int n = sc.nextInt();
        int[][] mat = new int[n][n];
        System.out.println("Enter matrix elements:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                mat[i][j] = sc.nextInt();
            }
        }
        int sum = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j <= i; j++) {
                sum = sum + mat[i][j];
            }
        }
        System.out.println("Sum of lower triangle elements: " + sum);
    }
}
```

OUTPUT:

```
<terminated> Program20 (3) [Java Application] C:\Program Files\Java\jdk-2
Enter order of square matrix:
3
Enter matrix elements:
1 2 3
4 5 6
1 2 3
Sum of lower triangle elements: 16
```

21) Write a program to find the sum of elements in each row and column of a matrix.

```
package Arrays;
import java.util.Scanner;
public class Program21 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of rows:");
        int r = sc.nextInt();
        System.out.println("Enter number of columns:");
        int c = sc.nextInt();
        int[][] mat = new int[r][c];
        System.out.println("Enter matrix elements:");
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                mat[i][j] = sc.nextInt();
            }
        }
        for (int i = 0; i < r; i++) {
            int rowSum = 0;
            for (int j = 0; j < c; j++) {
                rowSum += mat[i][j];
            }
            System.out.println("Sum of row " + (i + 1) + " = " + rowSum);
        }
         for (int j = 0; j < c; j++) {
            int colSum = 0;
            for (int i = 0; i < r; i++) {
                colSum += mat[i][j];
            }
            System.out.println("Sum of column " + (j + 1) + " = " + colSum);
        }   }}
```

OUTPUT:

```
Enter number of rows:
2
Enter number of columns:
3
Enter matrix elements:
2 3 4
1 2 3
Sum of row 1 = 9
Sum of row 2 = 6
Sum of column 1 = 3
Sum of column 2 = 5
Sum of column 3 = 7
```

22) Write a program to check whether a given matrix is symmetric or not.

```
package Arrays;
import java.util.Scanner;
public class Program22 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter order of square matrix:");
      int n = sc.nextInt();
      int[][] mat = new int[n][n];
      System.out.println("Enter matrix elements:");
      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
            mat[i][j] = sc.nextInt();
         }
      }
      boolean isSymmetric = true;
      for (int i = 0; i < n; i++) {
         for (int j = i + 1; j < n; j++) {
            if (mat[i][j] != mat[j][i]) {
               isSymmetric = false;
               break;
            }
         }
      }
      if (isSymmetric) {
         System.out.println("Matrix is Symmetric");
      } else {
         System.out.println("Matrix is NOT Symmetric");
      }
   }
}
```

OUTPUT:

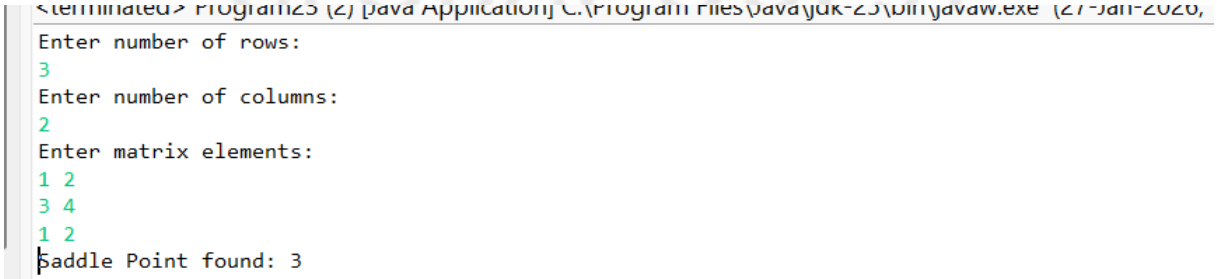
```
Enter order of square matrix:
3
Enter matrix elements:
1 2 3
1 2 3
1 2 3
Matrix is NOT Symmetric
```

23) Write a program to find the saddle point of a matrix.

```
package Arrays;
import java.util.Scanner;
public class Program23 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number of rows:");
      int r = sc.nextInt();
      System.out.println("Enter number of columns:");
      int c = sc.nextInt();
```

```
int[][] mat = new int[r][c];
System.out.println("Enter matrix elements:");
for (int i = 0; i < r; i++) {
    for (int j = 0; j < c; j++) {
        mat[i][j] = sc.nextInt();
    }
}
boolean found = false;
for (int i = 0; i < r; i++) {
    int min = mat[i][0];
    int colIndex = 0;
    for (int j = 1; j < c; j++) {
        if (mat[i][j] < min) {
            min = mat[i][j];
            colIndex = j;
        }
    }
    int k;
    for (k = 0; k < r; k++) {
        if (mat[k][colIndex] > min) {
            break;
        }
    }
    if (k == r) {
        System.out.println("Saddle Point found: " + min);
        found = true;
    }
}
if (!found) {
    System.out.println("No Saddle Point found");
}   }}
```

OUTPUT:



24) Write a program to find the kth smallest and kth largest elements in an array.

```
package Arrays;
import java.util.Scanner;
public class Program24 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of array:");
        int n = sc.nextInt();
```

```
        int[] arr = new int[n];
        System.out.println("Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter value of k:");
        int k = sc.nextInt();
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }       }
        if (k > 0 && k <= n) {
            System.out.println(k + "th Smallest element: " + arr[k - 1]);
            System.out.println(k + "th Largest element: " + arr[n - k]);
        } else {
            System.out.println("Invalid value of k");
        }
    }
}
```

OUTPUT:

```
Enter size of array:
5
Enter array elements:
1 2 3 4 5
Enter value of k:
4
4th Smallest element: 4
4th Largest element: 2
```

25) Write a program to count the number of negative, positive, and zero elements in an array.

```
package Arrays;
import java.util.Scanner;
public class Program25 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter size of array:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        int positive = 0, negative = 0, zero = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] > 0) {
                positive++;
```

```
            } else if (arr[i] < 0) {
                negative++;
            } else {
                zero++;
            }       }
        System.out.println("Positive numbers count: " + positive);
        System.out.println("Negative numbers count: " + negative);
        System.out.println("Zero count: " + zero);
    }
}
```

OUTPUT:

```
Enter size of array:
5
Enter array elements:
1 -2 0 3 -4
Positive numbers count: 2
Negative numbers count: 2
Zero count: 1
```

Data Types:

1) Write a program to demonstrate the use of primitive data types in Java.

```
public class Dt1 {
    public static void main(String[] args) {
        int a = 10;
        float b = 5.5f;
        double c = 25.75;
        char d = 'A';
        boolean e = true;
        System.out.println("int: " + a);
        System.out.println("float: " + b);
        System.out.println("double: " + c);
        System.out.println("char: " + d);
        System.out.println("boolean: " + e);
    }
}
```

Output:

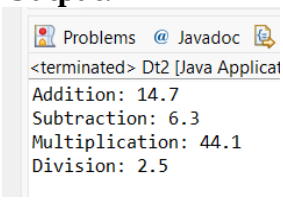
```
Problems  @ Java
<terminated> Dt1 [Java
int: 10
float: 5.5
double: 25.75
char: A
boolean: true
```

2) Write a program to perform arithmetic operations using float and double data types.

```
public class Dt2 {
    public static void main(String[] args) {
        float a = 10.5f;
        double b = 4.2;
        System.out.println("Addition: " + (a + b));
        System.out.println("Subtraction: " + (a - b));
        System.out.println("Multiplication: " + (a * b));
        System.out.println("Division: " + (a / b));
    }
```

```
    }
```

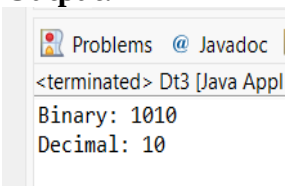
Output:



3) Write a program to convert an integer to binary and vice versa.

```
public class Dt3 {
    public static void main(String[] args) {
        int num = 10;
        String binary = Integer.toBinaryString(num);
        int decimal = Integer.parseInt(binary, 2);
        System.out.println("Binary: " + binary);
        System.out.println("Decimal: " + decimal);
    }
}
```

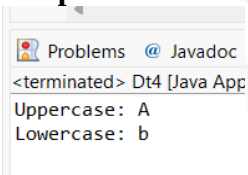
Output:



4) Write a program to perform operations on characters such as converting lowercase to uppercase and vice versa.

```
public class Dt4 {
    public static void main(String[] args) {
        char ch = 'a';
        System.out.println("Uppercase: " + Character.toUpperCase(ch));
        System.out.println("Lowercase: " + Character.toLowerCase('B'));
    }
}
```

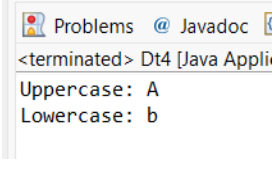
Output:



5) Write a program to demonstrate the use of boolean data type

```
public class Dt5{
    public static void main(String[] args) {
        int a = 10, b = 20;
        boolean result = a < b;
       System.out.println("Result: " + result);
    }}
```

Output:



6) Write a program to demonstrate the use of arrays in Java.

```
public class Dt6 {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40};
       for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
    }}
```

Output:

```
Problems @ Javadoc
<terminated> Dt6 [Java Applica
10 20 30 40
```

7) Write a program to find the maximum and minimum values in an array of integers.

```
public class Dt7 {
    public static void main(String[] args) {
        int[] arr = {10, 5, 25, 3};
        int max = arr[0], min = arr[0];
        for (int i : arr) {
            if (i > max) max = i;
            if (i < min) min = i;
        }
        System.out.println("Maximum: " + max);
        System.out.println("Minimum: " + min);
    }}
```

Output:

```
<terminated> Dt7 [Java Application] C
Maximum: 25
Minimum: 3
```

8) Write a program to calculate the average of elements in an array.

```
public class Dt8 {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30};
        int sum = 0;
        for (int i : arr)
            sum += i;
        System.out.println("Average: " + (sum / arr.length));
    }
}
```

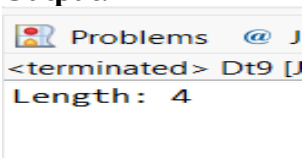
Output:

```
Problems @ Jav
<terminated> Dt8 [Jav
Average: 20
```

9) Write a program to find the length of a string.

```
public class Dt9 {
    public static void main(String[] args) {
        String str = "Java";
        System.out.println("Length: " + str.length());
    }
}
```

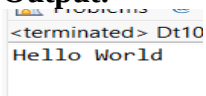
Output:



10) Write a program to concatenate two strings.

```
public class Dt10 {
 public static void main(String[] args) {
    String a = "Hello";
    String b = "World";
    System.out.println(a + " " + b);
 }}
```

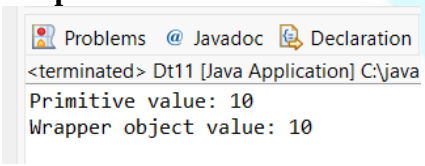
Output:



11) Write a program to demonstrate the use of wrapper classes in Java.

```
public class Dt11 {
   public static void main(String[] args) {
      int a = 10;
      Integer obj = Integer.valueOf(a);
      System.out.println("Primitive value: " + a);
      System.out.println("Wrapper object value: " + obj);
   }}
```

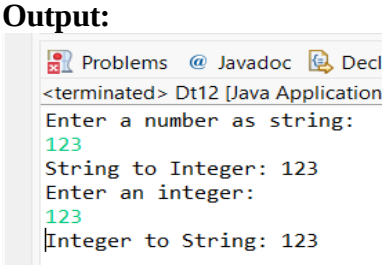
Output:



12) Write a program to convert a string to integer and vice versa.

```
import java.util.Scanner;
public class Dt12 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a number as string:");
      String str = sc.nextLine();
      int num = Integer.parseInt(str);
      System.out.println("String to Integer: " + num);
      System.out.println("Enter an integer:");
      int n = sc.nextInt();
      String s = Integer.toString(n);
      System.out.println("Integer to String: " + s);
   }}
```

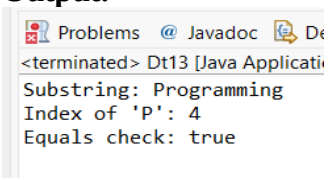
Output:



13) Write a program to demonstrate string methods such as substring, indexOf, and equals.

```
public class Dt13 {
    public static void main(String[] args) {
        String str = "JavaProgramming";
        System.out.println("Substring: " + str.substring(4));
        System.out.println("Index of 'P': " + str.indexOf('P'));
        System.out.println("Equals check: " + str.equals("JavaProgramming"));
    }}
```

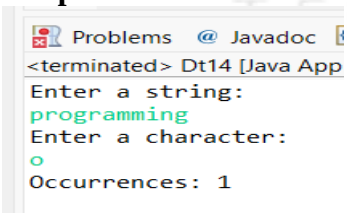
Output:



14) Write a program to count the occurrences of a character in a string.
Program.

```
package com.gqt_java.java_gqt.programs;
import java.util.Scanner;
public class Dt14 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.nextLine();
        System.out.println("Enter a character:");
        char ch = sc.next().charAt(0);
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i) == ch)
                count++;
        }
        System.out.println("Occurrences: " + count);
    }
}
```

Output:

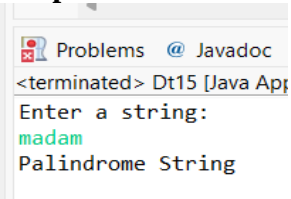


15) Write a program to check whether a given string is palindrome or not.

```
import java.util.Scanner;
public class Dt15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.next();
        String rev = "";
        for (int i = str.length() - 1; i >= 0; i--)
            rev += str.charAt(i);
```

```
        if (str.equals(rev))
            System.out.println("Palindrome String");
        else
            System.out.println("Not a Palindrome String");
    }}
```

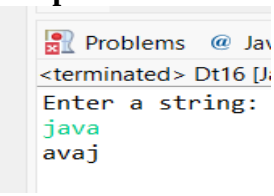
Output:



16) Write a program to reverse a string without using any built-in methods.

```
import java.util.Scanner;
public class Dt16 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.next();
        char[] arr = str.toCharArray();
        for (int i = arr.length - 1; i >= 0; i--)
            System.out.print(arr[i]);
    }}
```

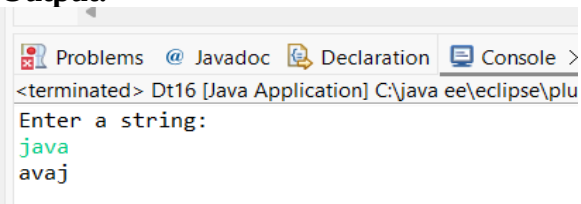
Output:



17) Write a program to convert a string to uppercase and vice versa.

```
import java.util.Scanner;
public class Dt16 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.next();
        char[] arr = str.toCharArray();
        for (int i = arr.length - 1; i >= 0; i--)
            System.out.print(arr[i]);
    }}
```

Output:

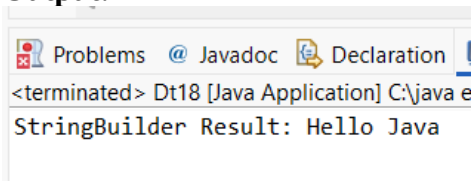


18) Write a program to demonstrate the use of StringBuilder class.

```
public class Dt18 {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("Hello");
```

```
        sb.append(" Java");
        System.out.println("StringBuilder Result: " + sb);
    }}
```

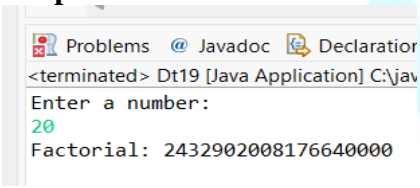
Output:



19) Write a program to find the factorial of a number using BigInteger class.

```
import java.math.BigInteger;
import java.util.Scanner;
public class Dt19 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number:");
        int n = sc.nextInt();
        BigInteger fact = BigInteger.ONE;
        for (int i = 1; i <= n; i++)
            fact = fact.multiply(BigInteger.valueOf(i));
        System.out.println("Factorial: " + fact);
    }}
```

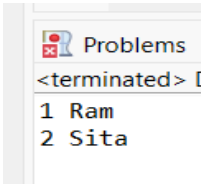
Output:



20) Write a program to demonstrate the use of arrays of objects.

```
class Student {
    int id;
    String name;
    Student(int id, String name) {
        this.id = id;
        this.name = name;
    }}
public class Dt20 {
    public static void main(String[] args) {
        Student[] s = new Student[2];
        s[0] = new Student(1, "Ram");
        s[1] = new Student(2, "Sita");
        for (Student st : s)
            System.out.println(st.id + " " + st.name);
    }}
```

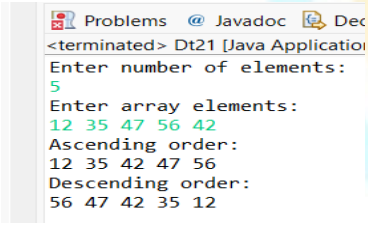
Output:



21) Write a program to sort an array of integers in ascending and descending order.
Program.

```
import java.util.Arrays;
import java.util.Scanner;
public class Dt21 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number of elements:");
      int n = sc.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter array elements:");
      for (int i = 0; i < n; i++)
         arr[i] = sc.nextInt();
      Arrays.sort(arr);
      System.out.println("Ascending order:");
      for (int i : arr)
         System.out.print(i + " ");
      System.out.println("\nDescending order:");
      for (int i = n - 1; i >= 0; i--)
         System.out.print(arr[i] + " ");
   }}
```

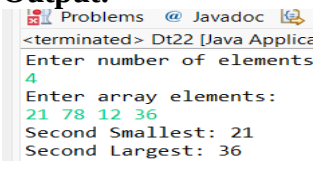
Output:



22) Write a program to find the second largest and second smallest elements in an array.

```
import java.util.Arrays;
import java.util.Scanner;
public class Dt22 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number of elements:");
      int n = sc.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter array elements:");
      for (int i = 0; i < n; i++)
         arr[i] = sc.nextInt();
      Arrays.sort(arr);
      System.out.println("Second Smallest: " + arr[1]);
      System.out.println("Second Largest: " + arr[n - 2]);
   }}
```

Output:

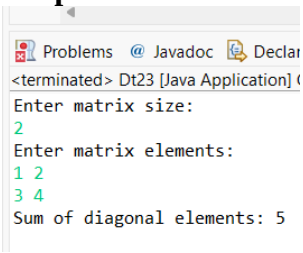


23) Write a program to find the sum of diagonal elements of a matrix.

Program.

```
import java.util.Scanner;
public class Dt23 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter matrix size:");
      int n = sc.nextInt();
      int[][] mat = new int[n][n];
      System.out.println("Enter matrix elements:");
      for (int i = 0; i < n; i++)
         for (int j = 0; j < n; j++)
            mat[i][j] = sc.nextInt();
      int sum = 0;
      for (int i = 0; i < n; i++)
         sum += mat[i][i];
      System.out.println("Sum of diagonal elements: " + sum);
   }}
```

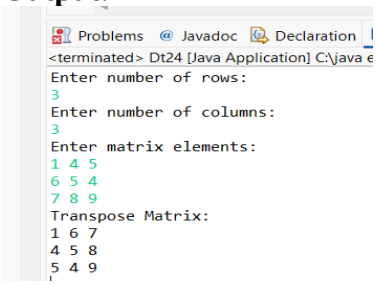
Output:



24) Write a program to transpose a matrix.

```
import java.util.Scanner;
public class Dt24 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number of rows:");
      int r = sc.nextInt();
      System.out.println("Enter number of columns:");
      int c = sc.nextInt();
      int[][] mat = new int[r][c];
      System.out.println("Enter matrix elements:");
      for (int i = 0; i < r; i++)
         for (int j = 0; j < c; j++)
            mat[i][j] = sc.nextInt();
      System.out.println("Transpose Matrix:");
      for (int j = 0; j < c; j++) {
         for (int i = 0; i < r; i++)
            System.out.print(mat[i][j] + " ");
         System.out.println();
      }}}
```

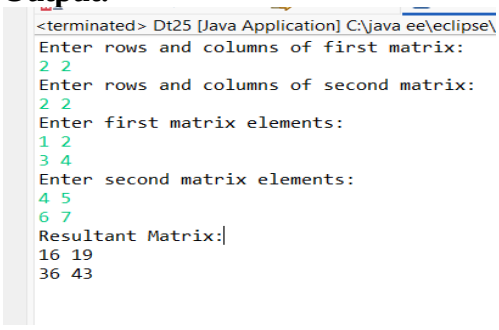
Output:



25) Write a program to multiply two matrices.

```
import java.util.Scanner;
public class Dt25 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter rows and columns of first matrix:");
      int r1 = sc.nextInt();
      int c1 = sc.nextInt();
      System.out.println("Enter rows and columns of second matrix:");
      int r2 = sc.nextInt();
      int c2 = sc.nextInt();
      int[][] a = new int[r1][c1];
      int[][] b = new int[r2][c2];
      int[][] mul = new int[r1][c2];
      System.out.println("Enter first matrix elements:");
      for (int i = 0; i < r1; i++)
         for (int j = 0; j < c1; j++)
            a[i][j] = sc.nextInt();
      System.out.println("Enter second matrix elements:");
      for (int i = 0; i < r2; i++)
         for (int j = 0; j < c2; j++)
            b[i][j] = sc.nextInt();
      for (int i = 0; i < r1; i++)
         for (int j = 0; j < c2; j++)
            for (int k = 0; k < c1; k++)
               mul[i][j] += a[i][k] * b[k][j];
      System.out.println("Resultant Matrix:");
      for (int i = 0; i < r1; i++) {
         for (int j = 0; j < c2; j++)
            System.out.print(mul[i][j] + " ");
         System.out.println();
      }}}
```

Output:

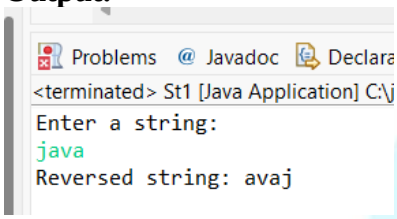


Strings.

1) Write a program to reverse a string.

```
import java.util.Scanner;
public class St1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a string:");
      String str = sc.nextLine();
      String rev = "";
      for (int i = str.length() - 1; i >= 0; i--)
         rev += str.charAt(i);
      System.out.println("Reversed string: " + rev);
   }}
```

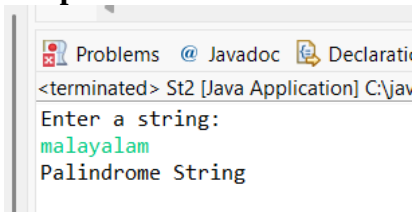
Output:



2) Write a program to check whether a given string is palindrome or not.

```
import java.util.Scanner;
public class St2 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a string:");
      String str = sc.next();
      String rev = "";
      for (int i = str.length() - 1; i >= 0; i--)
         rev += str.charAt(i);
      if (str.equals(rev))
         System.out.println("Palindrome String");
      else
         System.out.println("Not a Palindrome String");
   }
}
```

Output:

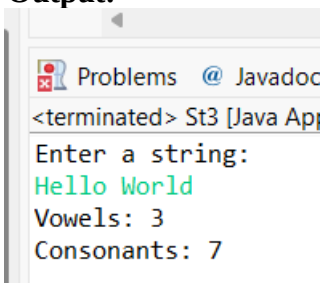


3) Write a program to count the number of vowels and consonants in a string.
Program.

```
import java.util.Scanner;
public class St3 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a string:");
```

```
        String str = sc.nextLine().toLowerCase();
        int vowels = 0, consonants = 0;
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
                    vowels++;
                else
                    consonants++;
            }
    }
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
    }
}
```

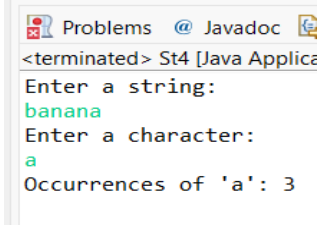
Output:



4) Write a program to count the occurrences of a character in a string.

```
import java.util.Scanner;
public class St4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.nextLine();
        System.out.println("Enter a character:");
        char ch = sc.next().charAt(0);
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i) == ch)
                count++;
        }
        System.out.println("Occurrences of '" + ch + "': " + count);
    }
}
```

Output:

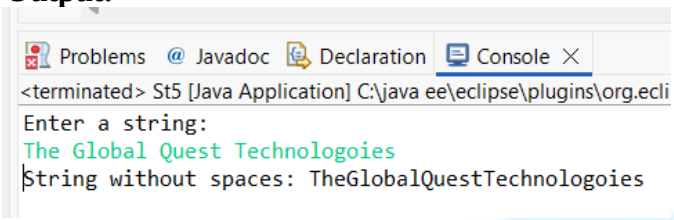


5) Write a program to remove all white spaces from a string.

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

```
public class St5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.nextLine();
        str = str.replaceAll("\\s+", "");
       System.out.println("String without spaces: " + str);
    }
}
```

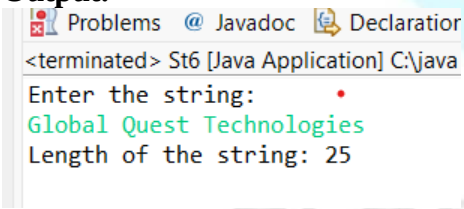
Output:



6) Write a program to find the length of a string.

```
import java.util.Scanner;
public class St6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string:");
        String str = sc.nextLine();
        System.out.println("Length of the string: " + str.length());
}}
```

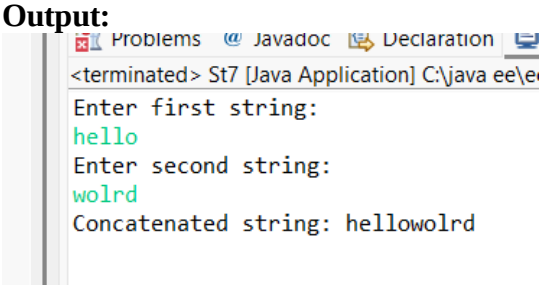
Output:



7) Write a program to concatenate two strings.

```
import java.util.Scanner;
public class St7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter first string:");
        String s1 = sc.nextLine();
        System.out.println("Enter second string:");
        String s2 = sc.nextLine();
        System.out.println("Concatenated string: " + s1 + s2);
    }}
```

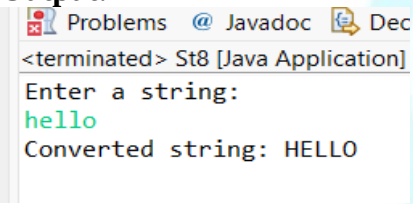
Output:



8) Write a program to convert lowercase letters to uppercase and vice versa.

```
import java.util.Scanner;
public class St8 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a string:");
      String str = sc.nextLine();
      String result = "";
      for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);
         if (Character.isUpperCase(ch))
            result += Character.toLowerCase(ch);
         else if (Character.isLowerCase(ch))
            result += Character.toUpperCase(ch);
         else
            result += ch;
      }
      System.out.println("Converted string: " + result);
   }}
```

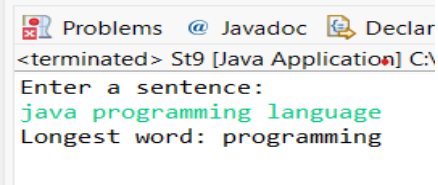
Output:



9) Write a program to find the longest word in a string.

```
import java.util.Scanner;
public class St9 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a sentence:");
      String str = sc.nextLine();
      String[] words = str.split(" ");
      String longest = words[0];
      for (String word : words) {
         if (word.length() > longest.length())
            longest = word;
      }
      System.out.println("Longest word: " + longest);
   }}
```

Output:

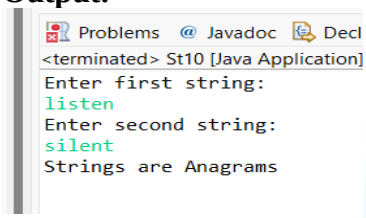


10) Write a program to check whether two strings are anagrams or not.

```
import java.util.Arrays;
import java.util.Scanner;
public class St10 {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter first string:");
        String s1 = sc.next().toLowerCase();
        System.out.println("Enter second string:");
        String s2 = sc.next().toLowerCase();
        char[] a = s1.toCharArray();
        char[] b = s2.toCharArray();
        Arrays.sort(a);
        Arrays.sort(b);
        if (Arrays.equals(a, b))
            System.out.println("Strings are Anagrams");
        else
            System.out.println("Strings are not Anagrams");
    }}
```

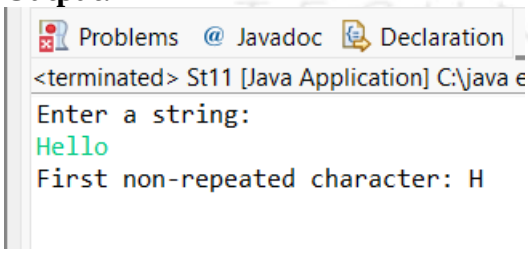
Output:



11) Write a program to find the first non-repeated character in a string.

```
import java.util.Scanner;
public class St11 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.nextLine();
        for (int i = 0; i < str.length(); i++) {
            if (str.indexOf(str.charAt(i)) == str.lastIndexOf(str.charAt(i))) {
                System.out.println("First non-repeated character: " + str.charAt(i));
                break;
            }}}}
```

Output:

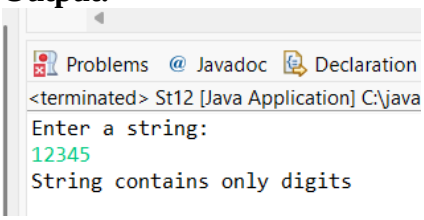


12) Write a program to check whether a string contains only digits or not.

```
import java.util.Scanner;
public class St12 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = sc.nextLine();
        boolean isDigit = str.matches("\\d+");
```

```
        if (isDigit)
            System.out.println("String contains only digits");
        else
            System.out.println("String does not contain only digits");
    }}
```

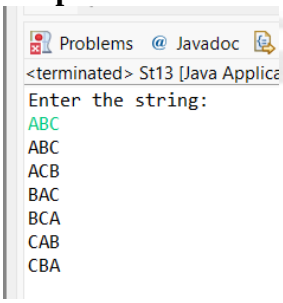
Output:



13) Program to find all permutations of a string.

```
import java.util.Scanner;
public class St13 {
    static void permute(String str, String ans) {
        if (str.length() == 0) {
            System.out.println(ans);
            return;
        }
        for (int i = 0; i < str.length(); i++) {
            permute(str.substring(0, i) + str.substring(i + 1), ans + str.charAt(i));
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string:");
        String s = sc.next();
        permute(s, "");
    }}
```

Output:



14) Program to find frequency of each character.

```
import java.util.*;
public class St14 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String s = sc.nextLine();
        Map<Character, Integer> map = new HashMap<>();
        for (char c : s.toCharArray())
            map.put(c, map.getOrDefault(c, 0) + 1);
        System.out.println("Character frequencies:");
```

```
        for (char c : map.keySet())
            System.out.println(c + " : " + map.get(c));
    }}
```

Output:

```
<terminated> St14 [Java App
Enter a string:
java
Character frequencies
a : 2
v : 1
j : 1
```

15) Write a program to reverse words in a sentence.

```
import java.util.Scanner;
public class St15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a sentence:");
        String[] words = sc.nextLine().split(" ");
        for (int i = words.length - 1; i >= 0; i--)
            System.out.print(words[i] + " ");
    }}
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> St15 [Java Application] C:\java ee\eclipse\plugi
Enter a sentence:
the group of sentence is known as sentence
sentence as known is sentence of group the
```

16) Write a program to find the duplicate characters in a string.

```
import java.util.*;
public class St16 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the string:");
        String s = sc.nextLine();
        HashSet<Character> set = new HashSet<>();
        System.out.println("Duplicate characters:");
        for (char c : s.toCharArray()) {
            if (!set.add(c))
                System.out.println(c);
        }}}
```

Output:

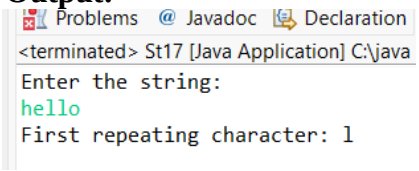
```
Problems  Javadoc  Dec
<terminated> St16 [Java Application
Enter the string:
Programming
Duplicate characters:
r
m
g
```

17) Write a program to find the first repeating character in a string.

```
import java.util.*;
public class St17 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
            System.out.println("Enter the string:");
            String s = sc.nextLine();
            HashSet<Character> set = new HashSet<>();
            for (char c : s.toCharArray()) {
               if (!set.add(c)) {
                  System.out.println("First repeating character: " + c);
                  return;
               }}
            System.out.println("No repeating characters found");
         }
      }
```

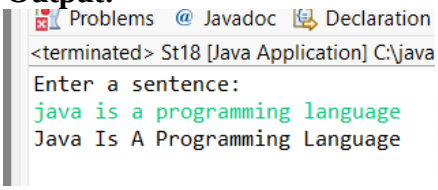
Output:



18) Write a program to capitalize the first letter of each word in a sentence.

```
import java.util.Scanner;
public class St18 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a sentence:");
      String[] words = sc.nextLine().split(" ");
      for (String w : words)
         System.out.print(w.substring(0,1).toUpperCase() + w.substring(1) + " ");
   }
}
```

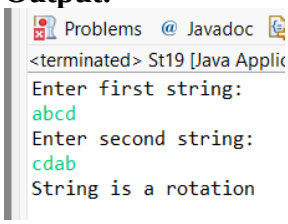
Output:



19) Write a program to check whether a string is a rotation of another string.

```
import java.util.Scanner;
public class St19 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter first string:");
      String s1 = sc.next();
      System.out.println("Enter second string:");
      String s2 = sc.next();
      if ((s1 + s1).contains(s2))
         System.out.println("String is a rotation");
      else
         System.out.println("String is not a rotation");
   }
}
```

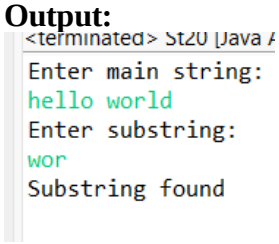
Output:



20) Write a program to check whether a string is a substring of another string.

```
import java.util.Scanner;
public class St20 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter main string:");
      String s1 = sc.nextLine();
      System.out.println("Enter substring:");
      String s2 = sc.nextLine();
      if (s1.contains(s2))
         System.out.println("Substring found");
      else
         System.out.println("Substring not found");
   }}
```

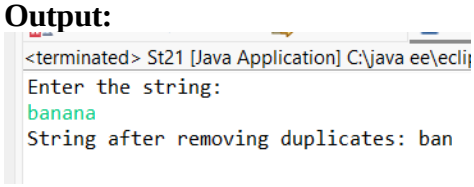
Output:



21) Write a program to remove duplicates from a string.

```
public class St21 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the string:");
      String s = sc.nextLine();
      String result = "";
      for (char c : s.toCharArray())
         if (!result.contains(String.valueOf(c)))
            result += c;
      System.out.println("String after removing duplicates: " + result);
   }}
```

Output:



22) Write a program to find the common characters in two given strings.

```
import java.util.*;
public class St22 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter first string:");
        String s1 = sc.next();
        System.out.println("Enter second string:");
        String s2 = sc.next();
        HashSet<Character> set = new HashSet<>();
        for (char c : s1.toCharArray())
            if (s2.indexOf(c) != -1)
                set.add(c);
        System.out.println("Common characters: " + set);
    }}
```

Output:

```
<terminated> St22 [Java Application]
Enter first string:
hello
Enter second string:
world
Common characters: [l, o]
```

23) Write a program to check whether a given string is a pangram or not.

```
import java.util.*;
public class St23{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a sentence:");
        String s = sc.nextLine().toLowerCase();
        HashSet<Character> set = new HashSet<>();
        for (char c : s.toCharArray())
            if (c >= 'a' && c <= 'z')
                set.add(c);
        if (set.size() == 26)
            System.out.println("Pangram");
        else
            System.out.println("Not a Pangram");
    }}
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> St23 [Java Application] C:\java ee\eclipse\plu
Enter a sentence:
The quick brown fox jumps over the lazy dog
Pangram
```

24) Write a program to find the smallest window in a string containing all characters of another string.

```
import java.util.Scanner;
public class St24 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter main string:");
        String s = sc.next();
        System.out.println("Enter pattern string:");
        String p = sc.next();
        int minLen = Integer.MAX_VALUE;
```

```
            String result = "";
            for (int i = 0; i < s.length(); i++) {
                for (int j = i + 1; j <= s.length(); j++) {
                    String sub = s.substring(i, j);
                    boolean found = true;
                    for (char c : p.toCharArray())
                        if (sub.indexOf(c) == -1)
                            found = false;
                    if (found && sub.length() < minLen) {
                        minLen = sub.length();
                        result = sub;
                    }}}
                System.out.println("Smallest window: " + result);
        }
    }
```

Output:

```
<terminated> St24 [Java Appli
Enter main string:
thisisateststring
Enter pattern string:
 tist
Smallest window: this
```

25) Write a program to find the longest common prefix among an array of strings.

```
import java.util.Scanner;
public class St25 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of strings:");
        int n = sc.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the strings:");
        for (int i = 0; i < n; i++)
            arr[i] = sc.next();
        String prefix = arr[0];
        for (int i = 1; i < n; i++)
            while (!arr[i].startsWith(prefix))
                prefix = prefix.substring(0, prefix.length() - 1);
        System.out.println("Longest common prefix: " + prefix);
    }}
```

Output:

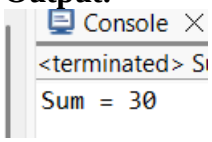
```
Problems @ Javadoc
<terminated> St25 [Java Applica
Enter number of strings:
3
Enter the strings:
flow
flight
flower
Longest common prefix: fl
```

OverLoading

1) Write a program to find the sum of two integers.

```
class SumInt {
   int add(int a, int b) {
      return a + b;
   }
   public static void main(String[] args) {
      SumInt s = new SumInt();
      System.out.println("Sum = " + s.add(10, 20));
   }}
```

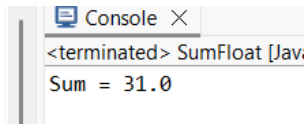
Output:



2) Write a program to find the sum of two floats.

```
class SumFloat {
   float add(float a, float b) {
      return a + b;
   }
   public static void main(String[] args) {
      SumFloat s = new SumFloat();
      System.out.println("Sum = " + s.add(10.5f, 20.5f));
   }
}
```

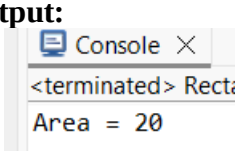
Output:



3) Write a program to find the area of a rectangle.

```
class Rectangle {
   int area(int l, int b) {
      return l * b;
   }
   public static void main(String[] args) {
      Rectangle r = new Rectangle();
      System.out.println("Area = " + r.area(5, 4));
   }}
```

Output:

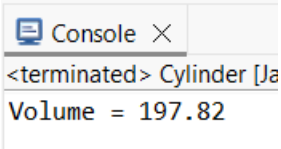


4) Write a program to find the volume of a cylinder.

```
class Cylinder {
   double volume(double r, double h) {
      return 3.14 * r * r * h;
   }
   public static void main(String[] args) {
```

```
        Cylinder c = new Cylinder();
        System.out.println("Volume = " + c.volume(3, 7));
    }}
```

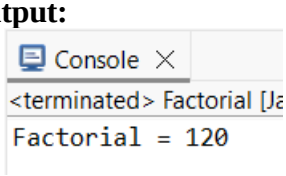
Output:



5) Write a program to calculate the factorial of a number.

```
class Factorial {
    int fact(int n) {
        int f = 1;
        for(int i=1;i<=n;i++) f *= i;
        return f;
    }
    public static void main(String[] args) {
        Factorial f = new Factorial();
        System.out.println("Factorial = " + f.fact(5));
    }}
```

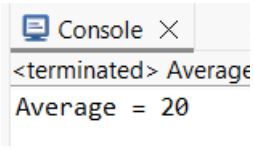
Output:



6) Write a program to find the average of three numbers.

```
class Average {
    int avg(int a, int b, int c) {
        return (a + b + c) / 3;
    }
    public static void main(String[] args) {
        Average a = new Average();
        System.out.println("Average = " + a.avg(10, 20, 30));
    }}
```

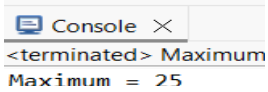
Output:



7) Write a program to find the maximum of two numbers.

```
class Maximum {
    int max(int a, int b) {
        return a > b ? a : b;    }
    public static void main(String[] args) {
        Maximum m = new Maximum();
        System.out.println("Maximum = " + m.max(10, 25));
    }}
```

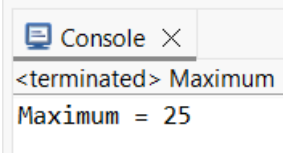
Output:



8) Write a program to find the minimum of two numbers.

```
class Minimum {
   int min(int a, int b) {
      return a < b ? a : b;
   }
   public static void main(String[] args) {
      Minimum m = new Minimum();
      System.out.println("Minimum = " + m.min(10, 25));
   }}
```

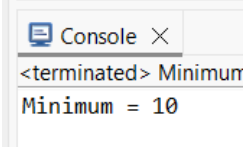
Output:



9) Write a program to calculate the power of a number.

```
class Minimum {
   int min(int a, int b) {
      return a < b ? a : b;
   }
   public static void main(String[] args) {
      Minimum m = new Minimum();
      System.out.println("Minimum = " + m.min(10, 25));
}}
```

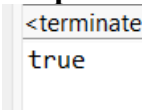
Output:



10) Write a program to check whether a number is prime or not.

```
class Prime {
   boolean check(int n) {
      if(n<=1) return false;
      for(int i=2;i<=n/2;i++)
         if(n%i==0) return false;
      return true;
   }
   public static void main(String[] args) {
      Prime p = new Prime();
      System.out.println(p.check(7));
   }}
```

Output:

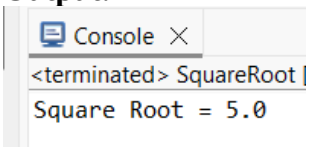


11) Write a program to find the square root of a number.

```
class SquareRoot {
   double root(double n) {
      return Math.sqrt(n);
   }
   public static void main(String[] args) {
```

```
        SquareRoot s = new SquareRoot();
        System.out.println("Square Root = " + s.root(25));
    }}
```

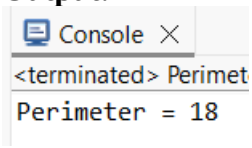
Output:



12) Write a program to calculate the perimeter of a rectangle.

```
class Perimeter {
    int peri(int l, int b) {
        return 2 * (l + b);
    }
    public static void main(String[] args) {
        Perimeter p = new Perimeter();
        System.out.println("Perimeter = " + p.peri(5, 4));
    }}
```

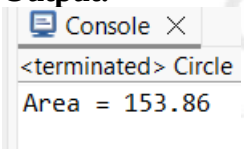
Output:



13) Write a program to find the area of a circle.

```
class Circle {
    double area(double r) {
        return 3.14 * r * r;
    }
    public static void main(String[] args) {
        Circle c = new Circle();
        System.out.println("Area = " + c.area(7));
    }}
```

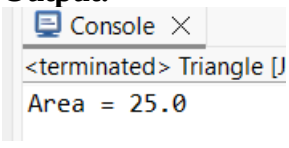
Output:



14) Write a program to find the area of a triangle.

```
class Triangle {
    double area(double b, double h) {
        return 0.5 * b * h;
    }
    public static void main(String[] args) {
        Triangle t = new Triangle();
        System.out.println("Area = " + t.area(10, 5));
    }}
```

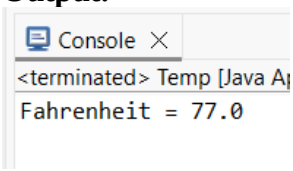
Output:



15) Write a program to convert Celsius to Fahrenheit.

```
class Temp {
   double convert(double c) {
      return (c * 9/5) + 32;
   }
   public static void main(String[] args) {
      Temp t = new Temp();
      System.out.println("Fahrenheit = " + t.convert(25));
   }}
```

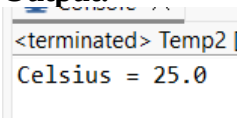
Output:



16) Write a program to convert Fahrenheit to Celsius.

```
class Temp2 {
   double convert(double f) {
      return (f - 32) * 5/9;
   }
   public static void main(String[] args) {
      Temp2 t = new Temp2();
      System.out.println("Celsius = " + t.convert(77));
   }}
```

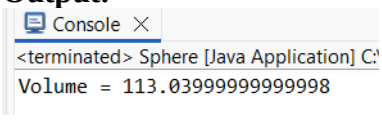
Output:



17) Write a program to find the volume of a sphere.

```
class Sphere {
   double volume(double r) {
      return (4.0/3) * 3.14 * r * r * r;
   }
   public static void main(String[] args) {
      Sphere s = new Sphere();
      System.out.println("Volume = " + s.volume(3));
   }}
```

Output:

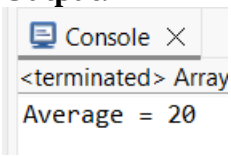


18) Write a program to find the average of an array of integers.

```
class ArrayAvg {
   int avg(int[] a) {
      int sum = 0;
      for(int i : a) sum += i;
      return sum / a.length;
   }
   public static void main(String[] args) {
      ArrayAvg a = new ArrayAvg();
      System.out.println("Average = " + a.avg(new int[]{10,20,30}));
```

```
}}
```

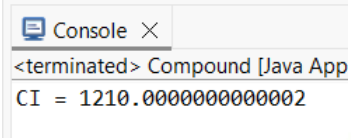
Output:



19) Write a program to calculate compound interest.

```
class Compound {
   double ci(double p, double r, double t) {
      return p * Math.pow(1 + r/100, t);
   }
   public static void main(String[] args) {
      Compound c = new Compound();
      System.out.println("CI = " + c.ci(1000, 10, 2));
   }}
```

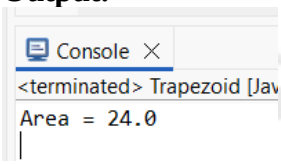
Output:



20) Write a program to find the area of a trapezoid.

```
class Trapezoid {
   double area(double a, double b, double h) {
      return 0.5 * (a + b) * h;
   }
   public static void main(String[] args) {
      Trapezoid t = new Trapezoid();
      System.out.println("Area = " + t.area(5, 7, 4));
   }}
```

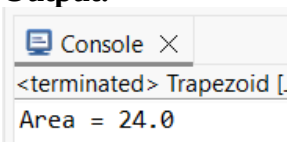
Output:



21) Write a program to find the area of a parallelogram.

```
class Parallelogram {
   double area(double b, double h) {
      return b * h;
   }
   public static void main(String[] args) {
      Parallelogram p = new Parallelogram();
      System.out.println("Area = " + p.area(6, 5));
   }}
```

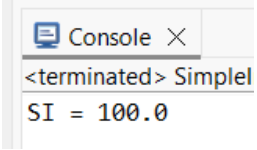
Output:



22) Write a program to calculate simple interest.

```
class SimpleInterest {
   double si(double p, double r, double t) {
      return (p * r * t) / 100;
   }
   public static void main(String[] args) {
      SimpleInterest s = new SimpleInterest();
      System.out.println("SI = " + s.si(1000, 5, 2));
   }}
```

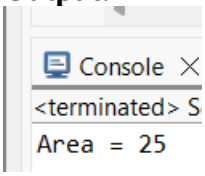
Output:



23) Write a program to find the area of a square.

```
class Square {
   int area(int s) {
      return s * s;
   }
   public static void main(String[] args) {
      Square s = new Square();
      System.out.println("Area = " + s.area(5));
   }}
```

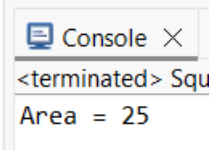
Output:



24) Write a program to find the area of a rhombus.

```
class Rhombus {
   double area(double d1, double d2) {
      return 0.5 * d1 * d2;
   }
   public static void main(String[] args) {
      Rhombus r = new Rhombus();
      System.out.println("Area = " + r.area(10, 8));
   }}
```

Output:

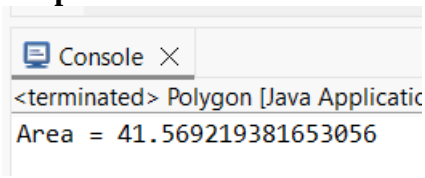


25) Write a program to find the area of a regular polygon.

```
class Polygon {
   double area(double n, double s) {
      return (n * s * s) / (4 * Math.tan(Math.PI/n));
   }
   public static void main(String[] args) {
      Polygon p = new Polygon();
```

```
        System.out.println("Area = " + p.area(6, 4));
    }}
```

Output:

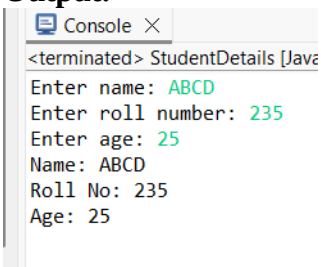


Encapsulation

1) Write a Java program to demonstrate encapsulation by creating a Student class with private data members name, roll number and age. Use public methods to set and display values.

```
import java.util.Scanner;
class Student1 {
    private String name;
    private int rollNo;
    private int age;
    public Student1() {
    }
    public void setData(String n, int r, int a) {
        name = n;
        rollNo = r;
        age = a;
    }
    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Roll No: " + rollNo);
        System.out.println("Age: " + age);
    }}
public class StudentDetails {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student1 s = new Student1();
        System.out.print("Enter name: ");
        String n = sc.nextLine();
        System.out.print("Enter roll number: ");
        int r = sc.nextInt();
        System.out.print("Enter age: ");
        int a = sc.nextInt();
        s.setData(n, r, a);
        s.display();
    }}
```

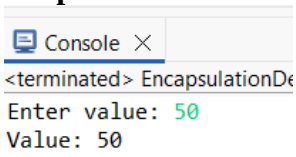
Output:



2) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class DemoEncap {
   private int value;
   public void setValue(int v) {
      value = v;
   }
   public int getValue() {
      return value;
   }}
public class EncapsulationDemo1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      DemoEncap d = new DemoEncap();
      System.out.print("Enter value: ");
      d.setValue(sc.nextInt());
      System.out.println("Value: " + d.getValue());
   }}
```

Output:

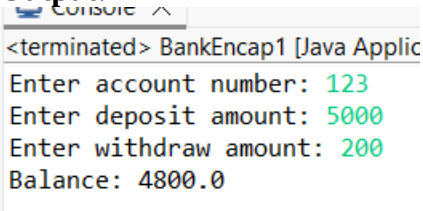


3) Create a class representing a bank account with private member variables (account number, balance) and public methods (deposit, withdraw).

```
import java.util.Scanner;
class BankEncap {
   private int accNo;
   private double balance;
   public void setAccount(int a) {
      accNo = a;
      balance = 0;
   }
   public void deposit(double amt) {
      balance += amt;
   }
   public void withdraw(double amt) {
      balance -= amt;
   }
   public double getBalance() {
      return balance;
   }}
public class BankEncap1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      BankEncap b = new BankEncap();
      System.out.print("Enter account number: ");
      b.setAccount(sc.nextInt());
```

```
            System.out.print("Enter deposit amount: ");
            b.deposit(sc.nextDouble());
            System.out.print("Enter withdraw amount: ");
            b.withdraw(sc.nextDouble());
           System.out.println("Balance: " + b.getBalance());
        }}
```

Output:

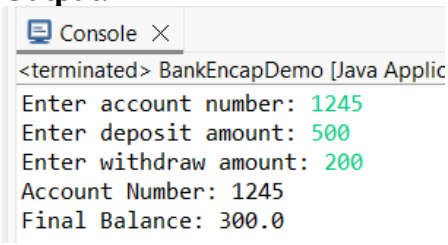


4) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class BankAccount {
   private int accountNumber;
   private double balance;
   public void setAccountNumber(int accNo) {
      accountNumber = accNo;
   }
   public void deposit(double amount) {
      balance = balance + amount;
   }
   public void withdraw(double amount) {
      balance = balance - amount;
   }
   public int getAccountNumber() {
      return accountNumber;
   }
  public double getBalance() {
      return balance;
   }}
public class BankEncapDemo {
   public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     BankAccount b = new BankAccount();
     System.out.print("Enter account number: ");
     b.setAccountNumber(sc.nextInt());
     System.out.print("Enter deposit amount: ");
     b.deposit(sc.nextDouble());
      System.out.print("Enter withdraw amount: ");
      b.withdraw(sc.nextDouble());
      System.out.println("Account Number: " + b.getAccountNumber());
      System.out.println("Final Balance: " + b.getBalance());
   }

}
```

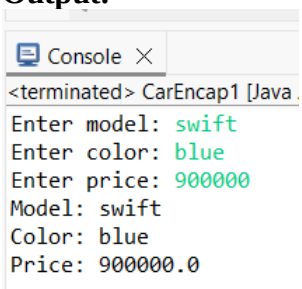
Output:



5) Create a class representing a car with private member variables (model, color, price) and public methods (getters and setters).

```
import java.util.Scanner;
class CarEncap {
   private String model, color;
   private double price;
   public void setCar(String m, String c, double p) {
      model = m;
      color = c;
      price = p;
   }
   public String getModel() { return model; }
   public String getColor() { return color; }
   public double getPrice() { return price; }
}
public class CarEncap1 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      CarEncap c = new CarEncap();
      System.out.print("Enter model: ");
      String m = sc.nextLine();
      System.out.print("Enter color: ");
      String col = sc.nextLine();
      System.out.print("Enter price: ");
      double p = sc.nextDouble();
      c.setCar(m, col, p);
      System.out.println("Model: " + c.getModel());
      System.out.println("Color: " + c.getColor());
      System.out.println("Price: " + c.getPrice());
   }}
```

Output:



6) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

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

```
class Car {
    private String model;
    private String color;
    private double price;
    public void setModel(String m) {
        model = m;
    }
    public void setColor(String c) {
        color = c;
    }
    public void setPrice(double p) {
        price = p;
    }
    public String getModel() {
        return model;
    }
    public String getColor() {
        return color;
    }
    public double getPrice() {
        return price;
    }}
public class CarEncapsulationDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Car c = new Car();
        System.out.print("Enter model: ");
        c.setModel(sc.nextLine());
        System.out.print("Enter color: ");
        c.setColor(sc.nextLine());
        System.out.print("Enter price: ");
        c.setPrice(sc.nextDouble());
       System.out.println("Model: " + c.getModel());
        System.out.println("Color: " + c.getColor());
        System.out.println("Price: " + c.getPrice());
    }}
```

Output:

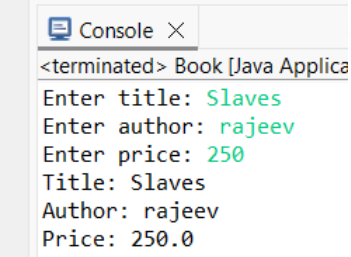
```
Console ×
<terminated> CarEncap1 [Java
Enter model: swift
Enter color: blue
Enter price: 900000
Model: swift
Color: blue
Price: 900000.0
```

7) Create a class representing a book with private member variables (title, author, price) and public methods (getters and setters).

```
import java.util.Scanner;
public class Book {
    private String title;
```

```
    private String author;
    private double price;
    public void setTitle(String t) {
        title = t;
    }
    public void setAuthor(String a) {
        author = a;
    }
    public void setPrice(double p) {
        price = p;
    }
    public String getTitle() {
        return title;
    }
    public String getAuthor() {
        return author;
    }
    public double getPrice() {
        return price;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Book b = new Book();
        System.out.print("Enter title: ");
        b.setTitle(sc.nextLine());
        System.out.print("Enter author: ");
        b.setAuthor(sc.nextLine());
        System.out.print("Enter price: ");
        b.setPrice(sc.nextDouble());
        System.out.println("Title: " + b.getTitle());
        System.out.println("Author: " + b.getAuthor());
        System.out.println("Price: " + b.getPrice());
    }}
```

Output:

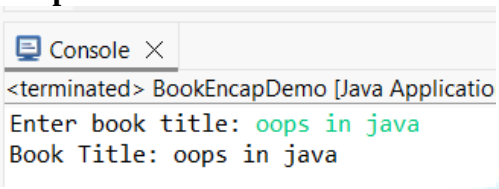


8) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class Book {
    private String title;
   public void setTitle(String t) {
        title = t;
    }
    public String getTitle() {
```

```
        return title;
    }}
public class BookEncapDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Book b = new Book();
        System.out.print("Enter book title: ");
        b.setTitle(sc.nextLine());
        System.out.println("Book Title: " + b.getTitle());
    }}
```

Output:

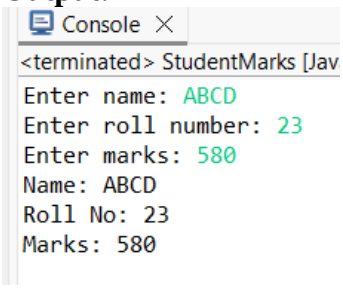


9) Create a class representing a student with private member variables (name, roll number, marks) and public methods (getters and setters).

```
import java.util.Scanner;
public class StudentMarks {
    private String name;
    private int rollNumber;
    private int marks;
    public void setName(String n) {
        name = n;
    }
    public void setRollNumber(int r) {
        rollNumber = r;
    }
    public void setMarks(int m) {
        marks = m;
    }
    public String getName() {
        return name;
    }
    public int getRollNumber() {
        return rollNumber;
    }
    public int getMarks() {
        return marks;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        StudentMarks s = new StudentMarks();
        System.out.print("Enter name: ");
        s.setName(sc.nextLine());
        System.out.print("Enter roll number: ");
        s.setRollNumber(sc.nextInt());
        System.out.print("Enter marks: ");
        s.setMarks(sc.nextInt());
```

```
        System.out.println("Name: " + s.getName());
        System.out.println("Roll No: " + s.getRollNumber());
        System.out.println("Marks: " + s.getMarks());
    }}
```

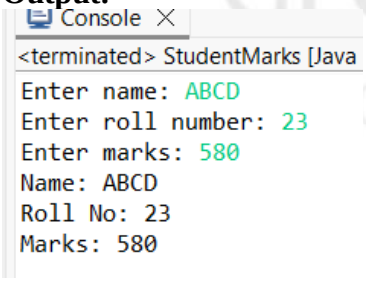
Output:



10) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class Student {
    private int marks;
    public void setMarks(int m) {
        marks = m;
    }
    public int getMarks() {
        return marks;
    }}
public class StudentMarksDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student s = new Student();
        System.out.print("Enter marks: ");
        s.setMarks(sc.nextInt());
        System.out.println("Marks: " + s.getMarks());
    }}
```

Output:

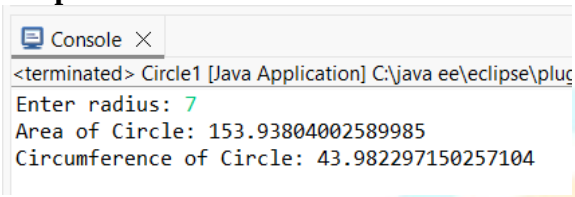


11) Create a class representing a circle with private member variables (radius, area, circumference) and public methods (getters and setters)

```
import java.util.Scanner;
public class Circle1 {
    private double radius;
    public void setRadius(double r) {
        radius = r;
    }
    public double getArea() {
        return Math.PI * radius * radius;
```

```
}
public double getCircumference() {
   return 2 * Math.PI * radius;
}
public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   Circle1 c = new Circle1();
   System.out.print("Enter radius: ");
   c.setRadius(sc.nextDouble());
   System.out.println("Area of Circle: " + c.getArea());
   System.out.println("Circumference of Circle: " + c.getCircumference());
}}
```

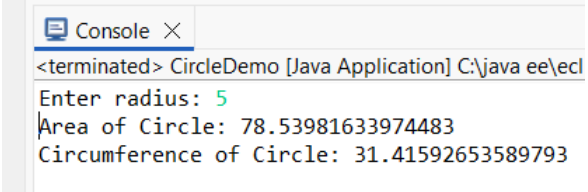
Output:



12) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
public class CircleDemo {
   private double radius;
   public void setRadius(double r) {
      radius = r;
   }
   public double getArea() {
      return Math.PI * radius * radius;
   }
   public double getCircumference() {
      return 2 * Math.PI * radius;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      CircleDemo c = new CircleDemo();
      System.out.print("Enter radius: ");
      c.setRadius(sc.nextDouble());
      System.out.println("Area of Circle: " + c.getArea());
      System.out.println("Circumference of Circle: " + c.getCircumference());
   }}
```

Output:

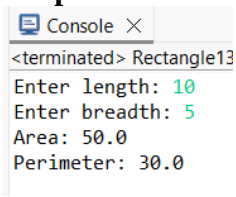


13) Create a class representing a Rectangle with private member variables (length, breadth) and public methods to calculate area and perimeter.

```
import java.util.Scanner;
public class Rectangle13 {
```

```
    private double length;
    private double breadth;
    public void setDimensions(double l, double b) {
        length = l;
        breadth = b;
    }
    public double getArea() {
        return length * breadth;
    }
    public double getPerimeter() {
        return 2 * (length + breadth);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Rectangle13 r = new Rectangle13();
        System.out.print("Enter length: ");
        double l = sc.nextDouble();
        System.out.print("Enter breadth: ");
        double b = sc.nextDouble();
        r.setDimensions(l, b);
        System.out.println("Area: " + r.getArea());
        System.out.println("Perimeter: " + r.getPerimeter());
    }}
```

Output:

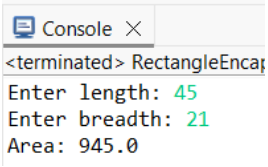


14) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods (Rectangle).

```
import java.util.Scanner;
public class RectangleEncap14 {
    private double length;
    private double breadth;
    public void setLength(double l) {
        length = l;
    }
    public void setBreadth(double b) {
        breadth = b;
    }
    public double getArea() {
        return length * breadth;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        RectangleEncap14 r = new RectangleEncap14();
        System.out.print("Enter length: ");
        r.setLength(sc.nextDouble());
        System.out.print("Enter breadth: ");
```

```
        r.setBreadth(sc.nextDouble());
        System.out.println("Area: " + r.getArea());
    }}
```

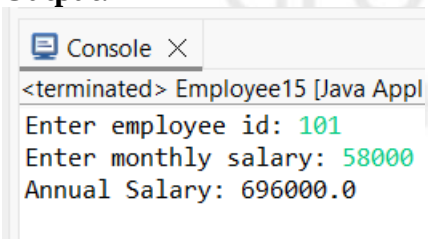
Output:



15) Create a class representing an Employee with private member variables (id, salary) and public methods to calculate annual salary.

```
import java.util.Scanner;
public class Employee15 {
    private int id;
    private double salary;
    public void setDetails(int i, double s) {
        id = i;
        salary = s;
    }
    public double getAnnualSalary() {
        return salary * 12;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Employee15 e = new Employee15();
        System.out.print("Enter employee id: ");
        int id = sc.nextInt();
        System.out.print("Enter monthly salary: ");
        double sal = sc.nextDouble();
        e.setDetails(id, sal);
        System.out.println("Annual Salary: " + e.getAnnualSalary());
    }}
```

Output:

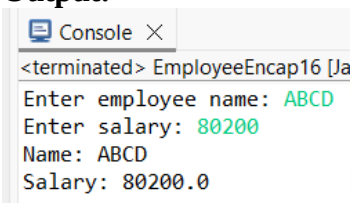


16) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods (Employee).

```
import java.util.Scanner;
public class EmployeeEncap16 {
    private String name;
    private double salary;
    public void setName(String n) {
        name = n;
    }
    public void setSalary(double s) {
        salary = s;
```

```
    }
    public String getName() {
        return name;
    }
    public double getSalary() {
        return salary;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        EmployeeEncap16 e = new EmployeeEncap16();
        System.out.print("Enter employee name: ");
        e.setName(sc.nextLine());
        System.out.print("Enter salary: ");
        e.setSalary(sc.nextDouble());
        System.out.println("Name: " + e.getName());
        System.out.println("Salary: " + e.getSalary());
    }}
```

Output:

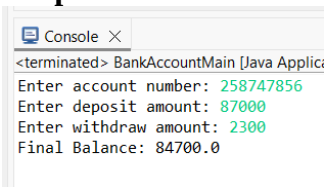


17) Create a class representing a Bank Account with private member variables (account number, balance) and public methods (deposit, withdraw).

```
import java.util.Scanner;
public class BankAccountMain {
    private int accountNumber;
    private double balance;
    public void setAccountNumber(int accNo) {
        accountNumber = accNo;
    }
    public void deposit(double amount) {
        balance += amount;
    }
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Insufficient Balance");
        }}
    public double getBalance() {
        return balance;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        BankAccountMain acc = new BankAccountMain();
        System.out.print("Enter account number: ");
        acc.setAccountNumber(sc.nextInt());
        System.out.print("Enter deposit amount: ");
```

```
        acc.deposit(sc.nextDouble());
        System.out.print("Enter withdraw amount: ");
        acc.withdraw(sc.nextDouble());
        System.out.println("Final Balance: " + acc.getBalance());
    }}
```

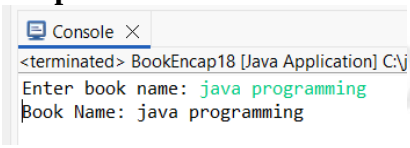
Output:



18) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
public class BookEncap18 {
    private String bookName;
    public void setBookName(String n) {
        bookName = n;
    }
    public String getBookName() {
        return bookName;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        BookEncap18 b = new BookEncap18();
        System.out.print("Enter book name: ");
        b.setBookName(sc.nextLine());
        System.out.println("Book Name: " + b.getBookName());
    }}
```

Output:

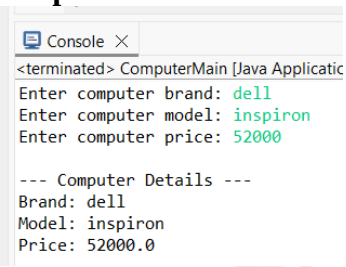


19) Create a class representing a Computer with private member variables (brand, model, price) and public methods (getters and setters).

```
import java.util.Scanner;
public class ComputerMain {
    private String brand;
    private String model;
    private double price;
    public void setBrand(String b) {
        brand = b;
    }
    public void setModel(String m) {
        model = m;
    }
    public void setPrice(double p) {
        price = p;
    }
```

```
    public String getBrand() {
        return brand;
    }
    public String getModel() {
        return model;
    }
    public double getPrice() {
        return price;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ComputerMain c = new ComputerMain();
        System.out.print("Enter computer brand: ");
        c.setBrand(sc.nextLine());
        System.out.print("Enter computer model: ");
        c.setModel(sc.nextLine());
        System.out.print("Enter computer price: ");
        c.setPrice(sc.nextDouble());
        System.out.println("\n--- Computer Details ---");
        System.out.println("Brand: " + c.getBrand());
        System.out.println("Model: " + c.getModel());
        System.out.println("Price: " + c.getPrice());
    }}
```

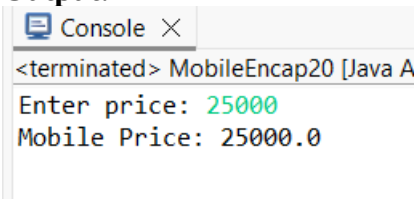
Output:



20) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods (Mobile).

```
import java.util.Scanner;
public class MobileEncap20 {
    private double price;
    public void setPrice(double p) {
        price = p;
    }
    public double getPrice() {
        return price;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        MobileEncap20 m = new MobileEncap20();
        System.out.print("Enter price: ");
        m.setPrice(sc.nextDouble());
        System.out.println("Mobile Price: " + m.getPrice());
    }
}
```

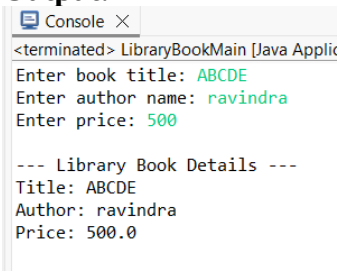
Output:



21) Create a class representing a Library Book with private member variables (title, author, price) and public methods (getters and setters).

```
import java.util.Scanner;
public class LibraryBookMain {
   private String title;
   private String author;
   private double price;
   public void setTitle(String t) {
      title = t;
   }
   public void setAuthor(String a) {
      author = a;
   }
   public void setPrice(double p) {
      price = p;
   }
   public String getTitle() {
      return title;
   }
   public String getAuthor() {
      return author;
   }
   public double getPrice() {
      return price;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      LibraryBookMain book = new LibraryBookMain();
      System.out.print("Enter book title: ");
      book.setTitle(sc.nextLine());
      System.out.print("Enter author name: ");
      book.setAuthor(sc.nextLine());
      System.out.print("Enter price: ");
      book.setPrice(sc.nextDouble());
      System.out.println("\n--- Library Book Details ---");
      System.out.println("Title: " + book.getTitle());
      System.out.println("Author: " + book.getAuthor());
      System.out.println("Price: " + book.getPrice());
   }
}
```

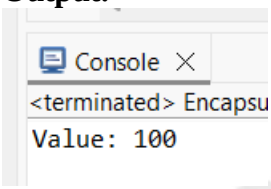
Output:



22) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
class Demo {
    private int value;
    public void setValue(int v) {
        value = v;
    }
    public int getValue() {
        return value;
    }
}
public class EncapsulationDemo22 {
    public static void main(String[] args) {
        Demo d = new Demo();
        d.setValue(100);
        System.out.println("Value: " + d.getValue());
    }}
```

Output:

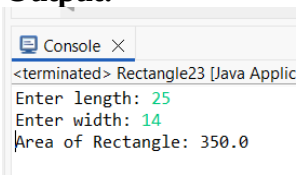


23) Create a class representing a rectangle with private member variables (length, width, area) and public methods (getters and setters).

```
import java.util.Scanner;
public class Rectangle23 {
    private double length, width, area;
    public void setLength(double length) {
        this.length = length;
    }
    public void setWidth(double width) {
        this.width = width;
    }
    public void calculateArea() {
        area = length * width;
    }
    public double getArea() {
        return area;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        Rectangle23 r = new Rectangle23();
        System.out.print("Enter length: ");
        r.setLength(sc.nextDouble());
        System.out.print("Enter width: ");
        r.setWidth(sc.nextDouble());
        r.calculateArea();
        System.out.println("Area of Rectangle: " + r.getArea());
    }
}
```

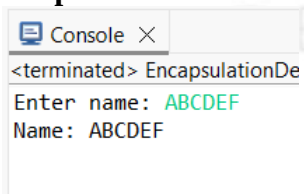
Output:



24) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class Test24 {
    private String name;
    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }}
public class EncapsulationDemo24 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Test24 t = new Test24();
        System.out.print("Enter name: ");
        t.setName(sc.nextLine());
        System.out.println("Name: " + t.getName());
    }}
```

Output:

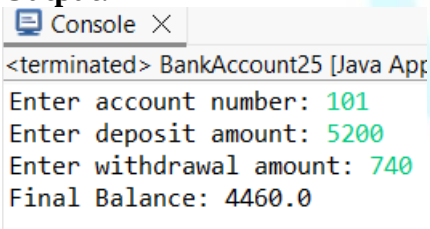


25) Create a class representing a bank account with private member variables (account number, balance) and public methods (deposit, withdraw).

```
import java.util.Scanner;
public class BankAccount25 {
    private int accNo;
    private double balance;
    public void setAccNo(int accNo) {
        this.accNo = accNo;
    }
    public void deposit(double amount) {
```

```
        balance += amount;
    }
    public void withdraw(double amount) {
        if (amount <= balance)
            balance -= amount;
        else
            System.out.println("Insufficient Balance");
    }
    public double getBalance() {
        return balance;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        BankAccount25 b = new BankAccount25();
        System.out.print("Enter account number: ");
        b.setAccNo(sc.nextInt());
        System.out.print("Enter deposit amount: ");
        b.deposit(sc.nextDouble());
        System.out.print("Enter withdrawal amount: ");
        b.withdraw(sc.nextDouble());
        System.out.println("Final Balance: " + b.getBalance());
    }}
```

Output:

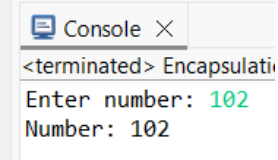


26) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class Sample26 {
    private int number;
    public void setNumber(int number) {
        this.number = number;
    }
    public int getNumber() {
        return number;
    }}
public class EncapsulationDemo26 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Sample26 s = new Sample26();
        System.out.print("Enter number: ");
        s.setNumber(sc.nextInt());
        System.out.println("Number: " + s.getNumber());
    }

}
```

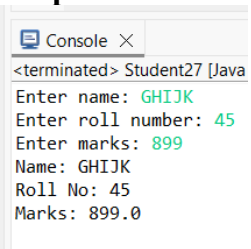
Output:



27) Create a class representing a student with private member variables (name, roll number, marks) and public methods (getters and setters).

```
import java.util.Scanner;
public class Student27 {
   private String name;
    private int roll;
    private double marks;
    public void setName(String name) {
       this.name = name;}
    public void setRoll(int roll) {
       this.roll = roll;
    }
    public void setMarks(double marks) {
       this.marks = marks;
    }
    public String getName() {
       return name;
    }
    public int getRoll() {
       return roll;
    }
    public double getMarks() {
       return marks;
    }
    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       Student27 s = new Student27();
       System.out.print("Enter name: ");
       s.setName(sc.nextLine());
       System.out.print("Enter roll number: ");
       s.setRoll(sc.nextInt());
       System.out.print("Enter marks: ");
       s.setMarks(sc.nextDouble());
       System.out.println("Name: " + s.getName());
       System.out.println("Roll No: " + s.getRoll());
       System.out.println("Marks: " + s.getMarks());
    }}
```

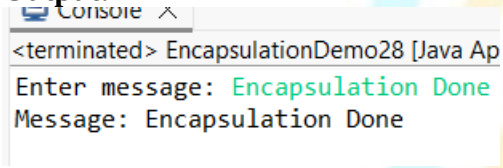
Output:



28) Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.

```
import java.util.Scanner;
class Encap28 {
   private String message;
   public void setMessage(String message) {
      this.message = message;
   }
   public String getMessage() {
      return message;
   }}
public class EncapsulationDemo28 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      Encap28 e = new Encap28();
     System.out.print("Enter message: ");
      e.setMessage(sc.nextLine());
      System.out.println("Message: " + e.getMessage());
   }}
```

Output:

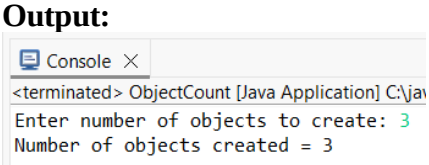


Static

1) Write a Java program to count the number of objects created for a class using a static variable.

```
import java.util.Scanner;
class ObjectCount {
   static int count = 0;
   ObjectCount() {
      count++;
   }
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter number of objects to create: ");
      int n = sc.nextInt();
      for (int i = 0; i < n; i++) {
         new ObjectCount();
      }
      System.out.println("Number of objects created = " + count);
   }}
```

Output:

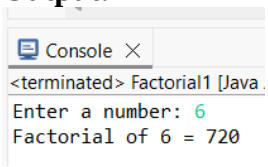


2) Implement a static method to find the factorial of a number.

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

```
class Factorial1 {
    static int factorial(int n) {
        int fact = 1;
        for (int i = 1; i <= n; i++)
            fact *= i;
        return fact;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();
        System.out.println("Factorial of " + n + " = " + factorial(n));
    }}
```

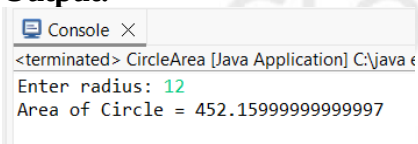
Output:



3) Write a Java program to calculate the area of a circle using a static method.

```
import java.util.Scanner;
class CircleArea {
    static double area(double r) {
        return 3.14 * r * r;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter radius: ");
        double r = sc.nextDouble();
        System.out.println("Area of Circle = " + area(r));
    }}
```

Output:



4) Write a Java program to find the sum of elements in an array using a static method.

```
import java.util.Scanner;
class ArraySum {
    static int sum(int[] a) {
        int s = 0;
        for (int i : a)
            s += i;
        return s;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter array size: ");
        int n = sc.nextInt();
        int[] a = new int[n];
        System.out.println("Enter array elements:");
```

```
        for (int i = 0; i < n; i++)
            a[i] = sc.nextInt();
        System.out.println("Sum of elements = " + sum(a));
    }}
```

Output:

```
Console ×
<terminated> ArraySum [Java
Enter array size: 4
Enter array elements:
12 52 32 14
Sum of elements = 110
```

5) Write a Java program to find the maximum of two numbers using a static method.

```
import java.util.Scanner;
class Max {
    static int max(int a, int b) {
        return (a > b) ? a : b;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter two numbers: ");
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println("Maximum number = " + max(a, b));
    }}
```

Output:

```
Console ×
<terminated> Max [Java Applicati
Enter two numbers: 45 87
Maximum number = 87
```

6) Write a Java program to find the minimum of two numbers using a static method.

```
import java.util.Scanner;
class Min {
    static int min(int a, int b) {
        return (a < b) ? a : b;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter two numbers: ");
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println("Minimum number = " + min(a, b));
    }}
```

Output:

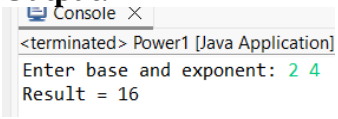
```
Console ×
<terminated> Min [Java Applicati
Enter two numbers: 78 89
Minimum number = 78
```

7) Write a Java program to find the power of a number using a static method.

```
import java.util.Scanner;
class Power1 {
    static int power(int a, int b) {
        return (int) Math.pow(a, b);
```

```
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter base and exponent: ");
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println("Result = " + power(a, b));
    }}
```

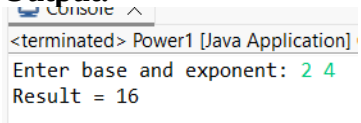
Output:



8) Write a Java program to calculate the square root of a number using a static method.

```
import java.util.Scanner;
class Power1 {
    static int power(int a, int b) {
        return (int) Math.pow(a, b);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter base and exponent: ");
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println("Result = " + power(a, b));
    }}
```

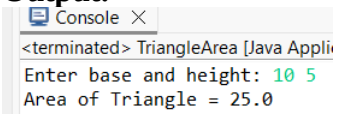
Output:



9) Write a Java program to find the area of a triangle using a static method.

```
import java.util.Scanner;
class TriangleArea {
    static double area(double b, double h) {
        return 0.5 * b * h;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter base and height: ");
        double b = sc.nextDouble();
        double h = sc.nextDouble();
        System.out.println("Area of Triangle = " + area(b, h));
    }}
```

Output:



10) Write a Java program to calculate simple interest using a static method.

```
import java.util.Scanner;
class SimpleInterest1 {
    static double calculateSI(double p, double r, double t) {
        return (p * r * t) / 100;
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter principal, rate and time: ");
        double p = sc.nextDouble();
        double r = sc.nextDouble();
        double t = sc.nextDouble();
        System.out.println("Simple Interest = " + calculateSI(p, r, t));
    }}
```

Output:

```
Console ×
<terminated> SimpleInterest1 [Java Application] C:\java ee
Enter principal, rate and time: 10000 2 3
Simple Interest = 600.0
```

11) Write a Java program to find the volume of a cylinder using a static method.

```
import java.util.Scanner;
class CylinderVolume {
    static double volume(double r, double h) {
        return 3.14 * r * r * h;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter radius and height: ");
        double r = sc.nextDouble();
        double h = sc.nextDouble();
        System.out.println("Volume of Cylinder = " + volume(r, h));
    }}
```

Output:

```
Console ×
<terminated> CylinderVolume [Java Applic
Enter radius and height: 12 25
Volume of Cylinder = 11304.0
```

12) Write a Java program to calculate compound interest using a static method.

```
import java.util.Scanner;
class CompoundInterest {
    static double ci(double p, double r, double t) {
        return p * Math.pow((1 + r / 100), t) - p;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter principal, rate and time: ");
        double p = sc.nextDouble();
        double r = sc.nextDouble();
        double t = sc.nextDouble();
        System.out.println("Compound Interest = " + ci(p, r, t));
    }}
```

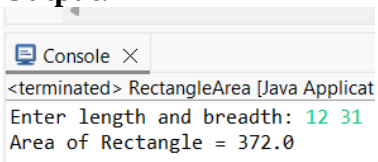
Output:

```
Console ×
<terminated> CompoundInterest [Java Application] C:\java ee
Enter principal, rate and time: 200000 2 6
Compound Interest = 25232.48385280001
```

13) Write a Java program to find the area of a rectangle using a static method.

```
import java.util.Scanner;
class RectangleArea {
   static double area(double l, double b) {
      return l * b;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter length and breadth: ");
      double l = sc.nextDouble();
      double b = sc.nextDouble();
      System.out.println("Area of Rectangle = " + area(l, b));
   }}
```

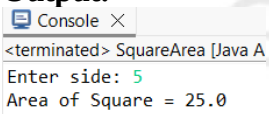
Output:



14) Write a Java program to find the area of a square using a static method.

```
import java.util.Scanner;
class SquareArea {
   static double area(double s) {
      return s * s;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter side: ");
      double s = sc.nextDouble();
      System.out.println("Area of Square = " + area(s));
   }}
```

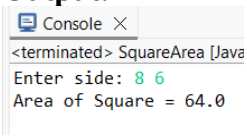
Output:



15) Write a Java program to find the area of a rhombus using a static method.

```
import java.util.Scanner;
class RhombusArea {
   static double area(double d1, double d2) {
      return 0.5 * d1 * d2;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter diagonals: ");
      double d1 = sc.nextDouble();
      double d2 = sc.nextDouble();
      System.out.println("Area of Rhombus = " + area(d1, d2));
   }
}
```

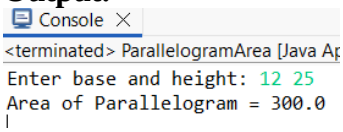
Output:



16) Write a Java program to find the area of a parallelogram using a static method.

```
import java.util.Scanner;
class ParallelogramArea {
   static double area(double b, double h) {
      return b * h;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter base and height: ");
      double b = sc.nextDouble();
      double h = sc.nextDouble();
      System.out.println("Area of Parallelogram = " + area(b, h));
   }}
```

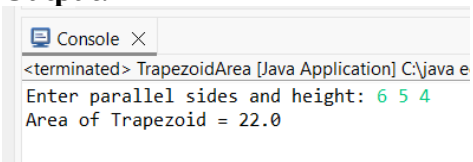
Output:



17) Write a Java program to find the area of a trapezoid using a static method.

```
import java.util.Scanner;
class TrapezoidArea {
   static double area(double a, double b, double h) {
      return 0.5 * (a + b) * h;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter parallel sides and height: ");
      double a = sc.nextDouble();
      double b = sc.nextDouble();
      double h = sc.nextDouble();
      System.out.println("Area of Trapezoid = " + area(a, b, h));
   }}
```

Output:

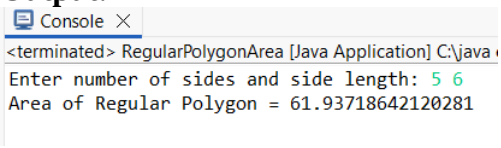


18) Write a Java program to find the area of a regular polygon using a static method.

```
import java.util.Scanner;
class RegularPolygonArea {
   static double area(int n, double s) {
      return (n * s * s) / (4 * Math.tan(Math.PI / n));
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter number of sides and side length: ");
```

```
        int n = sc.nextInt();
        double s = sc.nextDouble();
    System.out.println("Area of Regular Polygon = " + area(n, s));
    }}
```

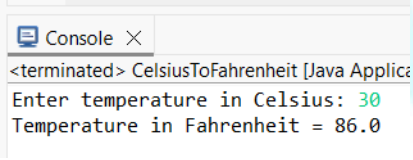
Output:



19) Write a Java program to convert temperature from Celsius to Fahrenheit using a static method.

```
import java.util.Scanner;
class CelsiusToFahrenheit {
    static double convert(double c) {
        return (c * 9 / 5) + 32;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter temperature in Celsius: ");
        double c = sc.nextDouble();
        System.out.println("Temperature in Fahrenheit = " + convert(c));
    }}
```

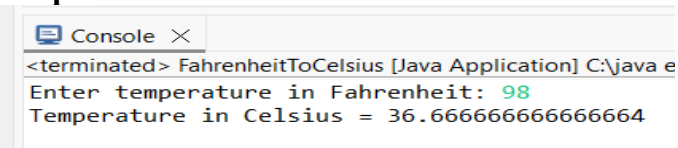
Output:



20) Write a Java program to convert temperature from Fahrenheit to Celsius using a static method.

```
import java.util.Scanner;
class FahrenheitToCelsius {
    static double convert(double f) {
        return (f - 32) * 5 / 9;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter temperature in Fahrenheit: ");
        double f = sc.nextDouble();
        System.out.println("Temperature in Celsius = " + convert(f));
    }}
```

Output:

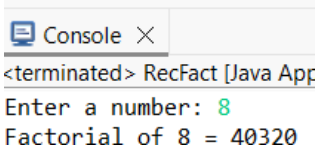


21) Write a Java program to find the factorial of a given number using recursion.

```
import java.util.Scanner;
class RecFact {
    static int factorial(int n) {
        if (n == 1)
```

```
            return 1;
        return n * factorial(n - 1);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();
    System.out.println("Factorial of " + n + " = " + factorial(n));
    }}
```

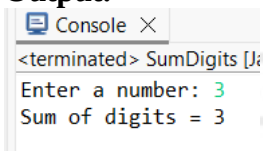
Output:



22) Write a Java program to find the sum of digits of a given number using recursion.

```
import java.util.Scanner;
class SumDigits {
    static int sum(int n) {
        if (n == 0)
            return 0;
        return (n % 10) + sum(n / 10);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = sc.nextInt();
        System.out.println("Sum of digits = " + sum(n));
    }}
```

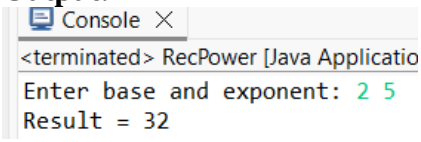
Output:



23) Write a Java program to calculate the power of a number using recursion.

```
import java.util.Scanner;
class RecPower {
    static int power(int a, int b) {
        if (b == 0)
            return 1;
        return a * power(a, b - 1);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter base and exponent: ");
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println("Result = " + power(a, b));
    }}
```

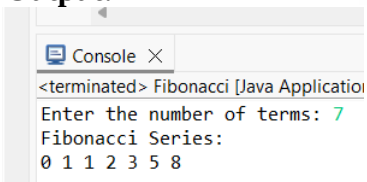
Output:



24) Write a Java program to generate the Fibonacci series using recursion.

```
import java.util.Scanner;
class Fibonacci {
   static int fib(int n) {
      if (n <= 1)
         return n;
      return fib(n - 1) + fib(n - 2);
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of terms: ");
      int n = sc.nextInt();
      System.out.println("Fibonacci Series:");
      for (int i = 0; i < n; i++) {
         System.out.print(fib(i) + " ");
      }
}
}
```

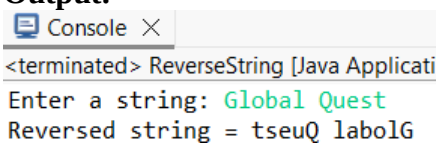
Output:



25) Write a Java program to reverse a given string using recursion.

```
import java.util.Scanner;
class ReverseString {
   static String reverse(String s) {
      if (s.isEmpty())
         return s;
      return reverse(s.substring(1)) + s.charAt(0);
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a string: ");
      String s = sc.nextLine();
      System.out.println("Reversed string = " + reverse(s));
   }}
```

Output:



Abstraction:

1) Write an abstract class "Shape" with abstract methods "calculateArea" and "calculatePerimeter". Implement it in subclasses "Circle" and "Rectangle".

```
package Abstraction;
abstract class Shape {
   abstract void calculateArea();
   abstract void calculatePerimeter();
}
class Circle extends Shape {
   double r = 5;
   void calculateArea() {
      System.out.println("Circle Area: " + (3.14 * r * r));
   }
   void calculatePerimeter() {
      System.out.println("Circle Perimeter: " + (2 * 3.14 * r));
   }
}
class Rectangle extends Shape {
   int l = 4, b = 6;
   void calculateArea() {
      System.out.println("Rectangle Area: " + (l * b));
   }
   void calculatePerimeter() {
      System.out.println("Rectangle Perimeter: " + (2 * (l + b)));
   }
}
public class Program1 {
   public static void main(String[] args) {
      Shape s1 = new Circle();
      s1.calculateArea();
      s1.calculatePerimeter();
      Shape s2 = new Rectangle();
      s2.calculateArea();
      s2.calculatePerimeter();
   }
}
```

OUTPUT:

```
<terminated> Program1 (5) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 3:26:
Circle Area: 78.5
Circle Perimeter: 31.400000000000002
Rectangle Area: 24
Rectangle Perimeter: 20
```

2) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal {
   abstract void eat();
   abstract void sleep();
}
```

```
class Dog extends Animal {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat extends Animal {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program2 {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        a1.eat();
        a1.sleep();
        Animal a2 = new Cat();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:

```
<terminated> Program2 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.e
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

3) Write an abstract class "Animal" with abstract methods "eat" and "sleep". Implement it in subclasses "Dog" and "Cat".

```
package Abstraction;
abstract class Animal {
    abstract void eat();
    abstract void sleep();
}
class Dog extends Animal {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat extends Animal {
    void eat() {
```

```
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program2 {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        a1.eat();
        a1.sleep();
        Animal a2 = new Cat();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:

```
<terminated> Program2 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 3:32:03 pm – 3:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

4) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal {
    abstract void eat();
    abstract void sleep();
}
class Dog extends Animal {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat extends Animal {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program2 {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        a1.eat();
        a1.sleep();
```

```
        Animal a2 = new Cat();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:

```
<terminated> Program2 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.e
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

5) Write an abstract class "Employee" with abstract methods "calculateSalary" and "calculateBonus". Implement it in subclasses "Manager" and "Clerk".

```
package Abstraction;
abstract class Employee {
    abstract void calculateSalary();
    abstract void calculateBonus();
}
class Manager extends Employee {
    void calculateSalary() {
        System.out.println("Manager Salary: 50000");
    }
    void calculateBonus() {
        System.out.println("Manager Bonus: 10000");
    }
}
class Clerk extends Employee {
    void calculateSalary() {
        System.out.println("Clerk Salary: 20000");
    }
    void calculateBonus() {
        System.out.println("Clerk Bonus: 3000");
    }}
public class Program3{
    public static void main(String[] args) {
        Employee e1 = new Manager();
        e1.calculateSalary();
        e1.calculateBonus();
        Employee e2 = new Clerk();
        e2.calculateSalary();
        e2.calculateBonus();
    }
}
```

OUTPUT:

```
<terminated> Program3 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 3:
Manager Salary: 50000
Manager Bonus: 10000
Clerk Salary: 20000
Clerk Bonus: 3000
```

6) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal {
   abstract void eat();
   abstract void sleep();
}
class Dog extends Animal {
   void eat() {
      System.out.println("Dog eats bones");
   }
   void sleep() {
      System.out.println("Dog sleeps at night");
   }
}
class Cat extends Animal {
   void eat() {
      System.out.println("Cat eats fish");
   }
   void sleep() {
      System.out.println("Cat sleeps during the day");
   }}
public class Program2 {
   public static void main(String[] args) {
      Animal a1 = new Dog();
      a1.eat();
      a1.sleep();
      Animal a2 = new Cat();
      a2.eat();
      a2.sleep();
   }
}
```

OUTPUT:

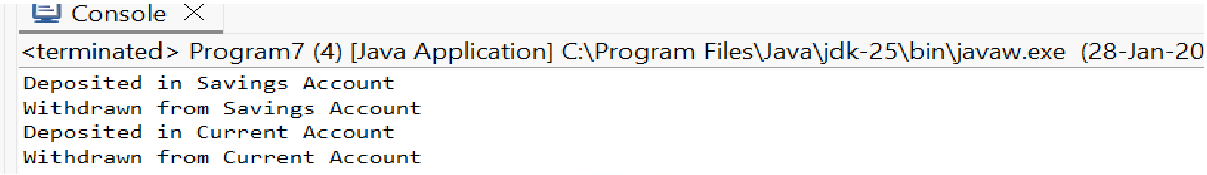
```
<terminated> Program2 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.e
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

7) Write an abstract class "BankAccount" with abstract methods "deposit" and "withdraw". Implement it in subclasses "SavingsAccount" and "CurrentAccount".

```
package Abstraction;
abstract class BankAccount {
   abstract void deposit();
   abstract void withdraw();
}
class SavingsAccount extends BankAccount {
   void deposit() {
      System.out.println("Deposited in Savings Account");
```

```
    }
    void withdraw() {
        System.out.println("Withdrawn from Savings Account");
    }
}
class CurrentAccount extends BankAccount {
    void deposit() {
        System.out.println("Deposited in Current Account");
    }
    void withdraw() {
        System.out.println("Withdrawn from Current Account");
    }
}
public class Program7 {
    public static void main(String[] args) {
        BankAccount b1 = new SavingsAccount();
        b1.deposit();
        b1.withdraw();
        BankAccount b2 = new CurrentAccount();
        b2.deposit();
        b2.withdraw();
    }
}
```

OUTPUT:

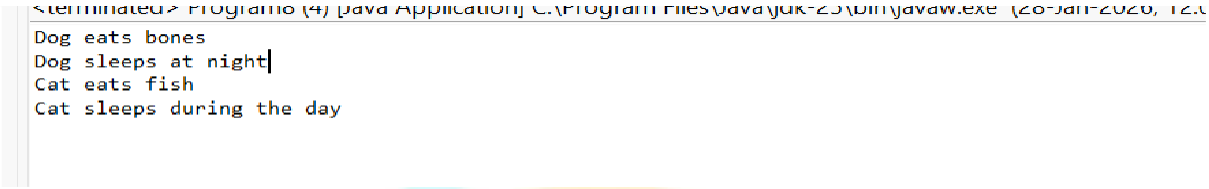


8) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
```

```
    }
}
public class Program8 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:



9) Write an abstract class "Vehicle" with abstract methods "start" and "stop". Implement it in subclasses "Car" and "Motorcycle".

```
abstract class Vehicle {
    abstract void start();
    abstract void stop();
}
class Car extends Vehicle {
    void start() {
        System.out.println("Car started");
    }
    void stop() {
        System.out.println("Car stopped");
    }
}
class Motorcycle extends Vehicle {
    void start() {
        System.out.println("Motorcycle started");
    }
    void stop() {
        System.out.println("Motorcycle stopped");
    }
}
public class VehicleDemo {
    public static void main(String[] args) {
        Vehicle v1 = new Car();
        v1.start();
        v1.stop();
        Vehicle v2 = new Motorcycle();
        v2.start();
        v2.stop();
    }
}
```

OUTPUT:

```
<terminated> Program9 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026, 12:09:54 pm
Car started
Car stopped
Motorcycle started
Motorcycle stopped
```

10) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
   }
   void sleep() {
      System.out.println("Dog sleeps at night");
   }
}
class Cat4 extends Animal4 {
   void eat() {
      System.out.println("Cat eats fish");
   }
   void sleep() {
      System.out.println("Cat sleeps during the day");
   }
}
public class Program8 {
   public static void main(String[] args) {
      Animal4 a1 = new Dog4();
      a1.eat();
      a1.sleep();
      Animal4 a2 = new Cat4();
      a2.eat();
      a2.sleep();
   }
}
```

OUTPUT:

```
<terminated> Program8 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

11) Write an abstract class "Shape" with abstract methods "calculateArea" and "calculatePerimeter". Implement it in subclasses "Triangle" and "Circle".

```
package Abstraction;
abstract class Shape1 {
   abstract void calculateArea();
   abstract void calculatePerimeter();
```

```
}
class Triangle extends Shape1 {
    int b = 5, h = 4;
    void calculateArea() {
        System.out.println("Triangle Area: " + (0.5 * b * h));
    }
    void calculatePerimeter() {
        System.out.println("Triangle Perimeter: 12");
    }
}
class Circle1 extends Shape1 {
    int r = 3;
    void calculateArea() {
        System.out.println("Circle Area: " + (3.14 * r * r));
    }
    void calculatePerimeter() {
        System.out.println("Circle Perimeter: " + (2 * 3.14 * r));
    }
}
public class Program10 {
    public static void main(String[] args) {
        Shape1 s1 = new Triangle();
        s1.calculateArea();
        s1.calculatePerimeter();
        Shape1 s2 = new Circle1();
        s2.calculateArea();
        s2.calculatePerimeter();
    }
}
```

OUTPUT:

```
<terminated> Program10 (5) [Java Application] C:\Program Files\Java\jd
Triangle Area: 10.0
Triangle Perimeter: 12
Circle Area: 28.259999999999998
Circle Perimeter: 18.84
```

12) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
```

```
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:

```
<terminated> Program13 (4) [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

13) Write an abstract class "Bank" with abstract methods "openAccount" and "closeAccount". Implement it in subclasses "SavingsBank" and "CurrentBank".

```
package Abstraction;
abstract class Bank {
    abstract void openAccount();
    abstract void closeAccount();
}
class SavingsBank extends Bank {
    void openAccount() {
        System.out.println("Savings Account opened");
    }
    void closeAccount() {
        System.out.println("Savings Account closed");
    }}
class CurrentBank extends Bank {
    void openAccount() {
        System.out.println("Current Account opened");
    }
    void closeAccount() {
        System.out.println("Current Account closed");
    }}
public class Program13 {
    public static void main(String[] args) {
        Bank b1 = new SavingsBank();
```

```
        b1.openAccount();
        b1.closeAccount();
        Bank b2 = new CurrentBank();
        b2.openAccount();
        b2.closeAccount();
    }}
```

OUTPUT:

```
<terminated> Program13 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\java
Savings Account opened
Savings Account closed
Current Account opened
Current Account closed
```

14) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:

```
<terminated> Program8 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

15) Write an abstract class "Figure" with abstract methods "draw" and "erase". Implement it in subclasses "Rectangle" and "Circle".

```
package Abstraction;
abstract class Figure {
   abstract void draw();
   abstract void erase();
}
class Rectangle2 extends Figure {
   void draw() {
      System.out.println("Rectangle drawn");
   }
   void erase() {
      System.out.println("Rectangle erased");
   }
}
class Circle2 extends Figure {
   void draw() {
      System.out.println("Circle drawn");
   }
   void erase() {
      System.out.println("Circle erased");
   }
}
public class Program14 {
   public static void main(String[] args) {
      Figure f1 = new Rectangle2();
      f1.draw();
      f1.erase();
      Figure f2 = new Circle2();
      f2.draw();
      f2.erase();
   }
}
```

OUTPUT:

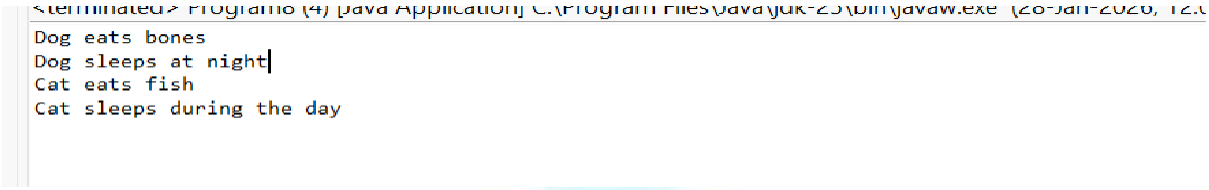
```
Rectangle drawn
Rectangle erased
Circle drawn
Circle erased
```

16) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
   }
```

```
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:



17) Write an abstract class "Vehicle" with abstract methods "drive" and "stop". Implement it in subclasses "Car" and "Truck".

```
package Abstraction;
abstract class Vehicle1 {
    abstract void drive();
    abstract void stop();
}
class Car1 extends Vehicle1 {
    void drive() {
        System.out.println("Car is driving");
    }
    void stop() {
        System.out.println("Car stopped");
    }}
class Truck1 extends Vehicle1 {
void drive() {
        System.out.println("Truck is driving");
    }
    void stop() {
        System.out.println("Truck stopped");
    }}
public class Program17 {
```

```
    public static void main(String[] args) {
        Vehicle1 v1 = new Car1();
        v1.drive();
        v1.stop();
        Vehicle1 v2 = new Truck1();
        v2.drive();
        v2.stop();
    }}
```

OUTPUT:

```
Car is driving
Car stopped
Truck is driving
Truck stopped
```

18) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

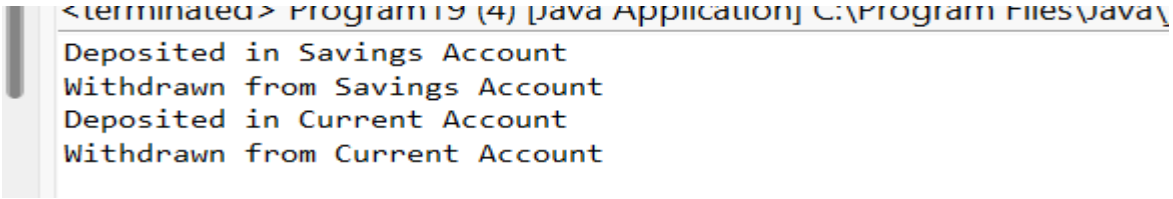
OUTPUT:

```
<terminated> Program8 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

19) Write an abstract class "BankAccount" with abstract methods "deposit" and "withdraw". Implement it in subclasses "SavingsAccount" and "CurrentAccount".

```
package Abstraction;
abstract class BankAccount {
   abstract void deposit();
   abstract void withdraw();
}
class SavingsAccount extends BankAccount {
   void deposit() {
      System.out.println("Amount deposited in Savings Account");
   }
   void withdraw() {
      System.out.println("Amount withdrawn from Savings Account");
   }
}
class CurrentAccount extends BankAccount {
   void deposit() {
      System.out.println("Amount deposited in Current Account");
   }
   void withdraw() {
      System.out.println("Amount withdrawn from Current Account");
   }
}
public class Program19 {
   public static void main(String[] args) {
      BankAccount b1 = new SavingsAccount();
      b1.deposit();
      b1.withdraw();
      BankAccount b2 = new CurrentAccount();
      b2.deposit();
      b2.withdraw();
   }
}
```

OUTPUT:

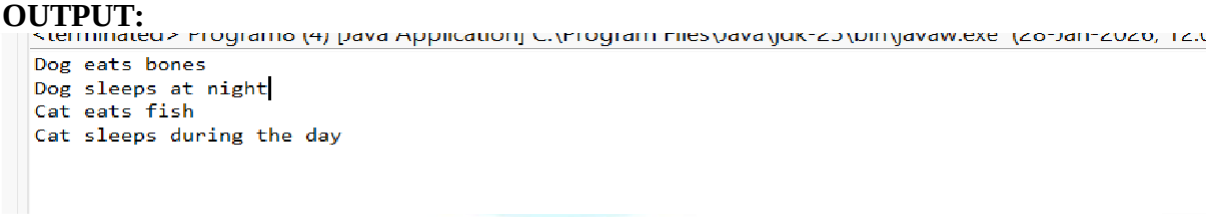


20) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
   }
```

```
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:



21) Write an abstract class "Animal" with abstract methods "eat" and "sleep". Implement it in subclasses "Dog" and "Cat".

```
package Abstraction;
abstract class Animal {
    abstract void eat();
    abstract void sleep();
}
class Dog extends Animal {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat extends Animal {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
```

```
public class Program2 {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        a1.eat();
        a1.sleep();
        Animal a2 = new Cat();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:

```
<terminated> Program2 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 3:32:03 pm – 3:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

22) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:

```
<terminated> Program6 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

23) Write an abstract class "Shape" with abstract methods "calculateArea" and "calculatePerimeter". Implement it in subclasses "Rectangle" and "Square".

```
package Abstraction;
abstract class Shape2 {
   abstract void calculateArea();
   abstract void calculatePerimeter();
}
class Rectangle1 extends Shape2 {
   int length = 5;
   int breadth = 4;
   void calculateArea() {
      System.out.println("Area of Rectangle: " + (length * breadth));
   }
   void calculatePerimeter() {
      System.out.println("Perimeter of Rectangle: " + (2 * (length + breadth)));
   }
}
class Square extends Shape2{
   int side = 4;
   void calculateArea() {
      System.out.println("Area of Square: " + (side * side));
   }
   void calculatePerimeter() {
      System.out.println("Perimeter of Square: " + (4 * side));
   }
public class Program23 {
   public static void main(String[] args) {
      Shape2 s1 = new Rectangle1();
      s1.calculateArea();
      s1.calculatePerimeter();
      Shape2 s2 = new Square();
      s2.calculateArea();
      s2.calculatePerimeter();
   }
}
```

OUTPUT:

```
<terminated> Program23 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026
Area of Rectangle: 20
Perimeter of Rectangle: 18
Area of Square: 16
Perimeter of Square: 16
```

24) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
```

```
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:

```
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

25) Write an abstract class "Bank" with abstract methods "openAccount" and "closeAccount". Implement it in subclasses "SavingsBank" and "CurrentBank".

```
package Abstraction;
abstract class Bank {
    abstract void openAccount();
    abstract void closeAccount();
}
class SavingsBank extends Bank {
    void openAccount() {
        System.out.println("Savings Account opened");
    }
    void closeAccount() {
        System.out.println("Savings Account closed");
    }}
class CurrentBank extends Bank {
    void openAccount() {
        System.out.println("Current Account opened");
    }
    void closeAccount() {
        System.out.println("Current Account closed");
    }}
public class Program13 {
```

```
    public static void main(String[] args) {
        Bank b1 = new SavingsBank();
        b1.openAccount();
        b1.closeAccount();
        Bank b2 = new CurrentBank();
        b2.openAccount();
        b2.closeAccount();
    }}
```

OUTPUT:

```
<terminated> Program13 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\java
Savings Account opened
Savings Account closed
Current Account opened
Current Account closed
```

26) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:

```
<terminated> Program8 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

27) Write an abstract class "Vehicle" with abstract methods "start" and "stop". Implement it in subclasses "Car" and "Motorcycle".

```
abstract class Vehicle {
   abstract void start();
   abstract void stop();
}
class Car extends Vehicle {
   void start() {
      System.out.println("Car started");
   }
   void stop() {
      System.out.println("Car stopped");
   }
}
class Motorcycle extends Vehicle {
   void start() {
      System.out.println("Motorcycle started");
   }
   void stop() {
      System.out.println("Motorcycle stopped");
   }
}
public class VehicleDemo {
   public static void main(String[] args) {
      Vehicle v1 = new Car();
      v1.start();
      v1.stop();
      Vehicle v2 = new Motorcycle();
      v2.start();
      v2.stop();
   }
}
```

OUTPUT:

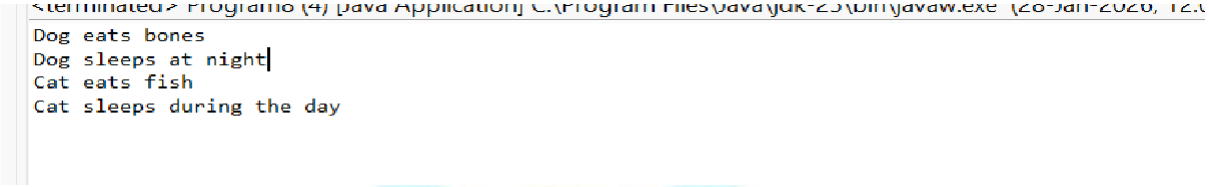
```
<terminated> Program9 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026, 12:09:54 pm
Car started
Car stopped
Motorcycle started
Motorcycle stopped
```

28) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
   }
   void sleep() {
      System.out.println("Dog sleeps at night");
```

```
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }
}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }
}
```

OUTPUT:



29) Write an abstract class "Shape" with abstract methods "calculateArea" and "calculatePerimeter". Implement it in subclasses "Triangle" and "Circle".

```
package Abstraction;
abstract class Shape1 {
    abstract void calculateArea();
    abstract void calculatePerimeter();
}
class Triangle extends Shape1 {
    int b = 5, h = 4;
    void calculateArea() {
        System.out.println("Triangle Area: " + (0.5 * b * h));
    }
    void calculatePerimeter() {
        System.out.println("Triangle Perimeter: 12");
    }}
class Circle1 extends Shape1 {
    int r = 3;
    void calculateArea() {
        System.out.println("Circle Area: " + (3.14 * r * r));
    }
    void calculatePerimeter() {
        System.out.println("Circle Perimeter: " + (2 * 3.14 * r));
    }}
public class Program10 {
```

```
    public static void main(String[] args) {
        Shape1 s1 = new Triangle();
        s1.calculateArea();
        s1.calculatePerimeter();
        Shape1 s2 = new Circle1();
        s2.calculateArea();
        s2.calculatePerimeter();
    }}
```

OUTPUT:

```
<terminated> Program10 (5) [Java Application] C:\Program Files\Java\jd
Triangle Area: 10.0
Triangle Perimeter: 12
Circle Area: 28.259999999999998
Circle Perimeter: 18.84
```

30) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
    abstract void eat();
    abstract void sleep();
}
class Dog4 extends Animal4 {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:

```
<terminated> Program8 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

31) Write an abstract class "BankAccount" with abstract methods "deposit" and "withdraw". Implement it in subclasses "SavingsAccount" and "CurrentAccount".

```
package Abstraction;
abstract class Bank {
   abstract void openAccount();
   abstract void closeAccount();
}
class SavingsBank extends Bank {
   void openAccount() {
      System.out.println("Savings Account opened");
   }
   void closeAccount() {
      System.out.println("Savings Account closed");
   }
}
class CurrentBank extends Bank {
   void openAccount() {
      System.out.println("Current Account opened");
   }
   void closeAccount() {
      System.out.println("Current Account closed");
   }
}
public class Program13 {
   public static void main(String[] args) {
      Bank b1 = new SavingsBank();
      b1.openAccount();
      b1.closeAccount();
      Bank b2 = new CurrentBank();
      b2.openAccount();
      b2.closeAccount();
   }
}
```

OUTPUT:

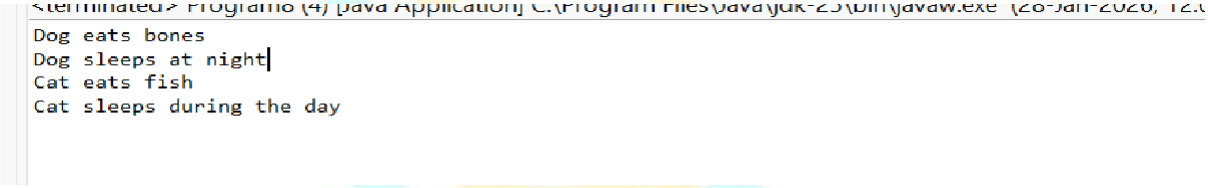
```
<terminated> Program13 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\java
Savings Account opened
Savings Account closed
Current Account opened
Current Account closed
```

32) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
   }
   void sleep() {
```

```
        System.out.println("Dog sleeps at night");
    }
}
class Cat4 extends Animal4 {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program12 {
    public static void main(String[] args) {
        Animal4 a1 = new Dog4();
        a1.eat();
        a1.sleep();
        Animal4 a2 = new Cat4();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:



33) Write an abstract class "Animal" with abstract methods "eat" and "sleep". Implement it in subclasses "Dog" and "Cat".

```
package Abstraction;
abstract class Animal {
    abstract void eat();
    abstract void sleep();
}
class Dog extends Animal {
    void eat() {
        System.out.println("Dog eats bones");
    }
    void sleep() {
        System.out.println("Dog sleeps at night");
    }}
class Cat extends Animal {
    void eat() {
        System.out.println("Cat eats fish");
    }
    void sleep() {
        System.out.println("Cat sleeps during the day");
    }}
public class Program2 {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        a1.eat();
```

```
        a1.sleep();
        Animal a2 = new Cat();
        a2.eat();
        a2.sleep();
    }}
```

OUTPUT:

```
<terminated> Program2 (3) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (27-Jan-2026, 3:32:03 pm – 3:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

34) Write a program to demonstrate abstraction by creating objects of subclasses and invoking abstract methods.

```
package Abstraction;
abstract class Animal4{
   abstract void eat();
   abstract void sleep();
}
class Dog4 extends Animal4 {
   void eat() {
      System.out.println("Dog eats bones");
   }
   void sleep() {
      System.out.println("Dog sleeps at night");
   }
}
class Cat4 extends Animal4 {
   void eat() {
      System.out.println("Cat eats fish");
   }
   void sleep() {
      System.out.println("Cat sleeps during the day");
   }
}
public class Program12 {
   public static void main(String[] args) {
      Animal4 a1 = new Dog4();
      a1.eat();
      a1.sleep();
      Animal4 a2 = new Cat4();
      a2.eat();
      a2.sleep();
   }
}
```

OUTPUT:

```
<terminated> Program8 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (28-Jan-2026, 12:
Dog eats bones
Dog sleeps at night
Cat eats fish
Cat sleeps during the day
```

Inheritance:

1) **Create a base class "Vehicle" with properties (make, model, year) and a subclass "Car" with additional properties (color, mileage).**

```
class Vehicle {
    String make;
    String model;
    int year;
    Vehicle(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }
    void displayVehicle() {
        System.out.println("Make : " + make);
        System.out.println("Model : " + model);
        System.out.println("Year : " + year);
    }}
class Car extends Vehicle {
    String color;
    int mileage;
    Car(String make, String model, int year, String color, int mileage) {
        super(make, model, year);
        this.color = color;
        this.mileage = mileage;
    }
    void displayCar() {
        displayVehicle();
        System.out.println("Color : " + color);
        System.out.println("Mileage : " + mileage + " km");
    }}
public class Main {
    public static void main(String[] args) {
        Vehicle v = new Vehicle("Honda", "Activa", 2020);
        System.out.println("Vehicle Details:");
        v.displayVehicle();

        System.out.println("\nCar Details:");
        Car c = new Car("Toyota", "Innova", 2022, "White", 15000);
        c.displayCar();
    }}
```

OUTPUT:

```
Vehicle Details:
Make : Honda
Model : Activa
Year : 2020

Car Details:
Make : Toyota
Model : Innova
Year : 2022
Color : White
Mileage : 15000 km
```

2) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Person {
String name;
 int age;
}
class Student extends Person {
 int rollNo;
 String course;
}
public class Program3
{
 public static void main(String[] args) {
    Person p = new Person();
    p.name = "Ravi";
    p.age = 40;
    System.out.println("Person Details:");
    System.out.println("Name: " + p.name);
    System.out.println("Age: " + p.age);
    System.out.println();
    Student s = new Student();
    s.name = "Bharathi";
    s.age = 21;
    s.rollNo = 101;
    s.course = "Java";
    System.out.println("Student Details:");
    System.out.println("Name: " + s.name);
    System.out.println("Age: " + s.age);
    System.out.println("Roll No: " + s.rollNo);
    System.out.println("Course: " + s.course);
 }}
```

OUTPUT:

```
Person Details:
Name: Ravi
Age: 40

Student Details:
Name: Bharathi
Age: 21
Roll No: 101
Course: Java
```

3) Create a base class "Shape" with methods to calculate area and perimeter. Derive classes "Circle" and "Rectangle" from it and override the methods.

```
abstract class Shape {
   abstract void calculateArea();
   abstract void calculatePerimeter();
}
class Circle extends Shape {
   double radius;
```

```
    Circle(double radius) {
        this.radius = radius;
    }
    void calculateArea() {
        System.out.println("Circle Area : " + (Math.PI * radius * radius));
    }
    void calculatePerimeter() {
        System.out.println("Circle Perimeter : " + (2 * Math.PI * radius));
    }
}
class Rectangle extends Shape {
    double length, width;
    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    void calculateArea() {
        System.out.println("Rectangle Area : " + (length * width));
    }
    void calculatePerimeter() {
        System.out.println("Rectangle Perimeter : " + (2 * (length + width)));
    }
}
public class Main {
    public static void main(String[] args) {
        Shape s1 = new Circle(5);

s1.calculateArea();

        s1.calculatePerimeter();

        Shape s2 = new Rectangle(10, 5);
        s2.calculateArea();
        s2.calculatePerimeter();
    }}
```

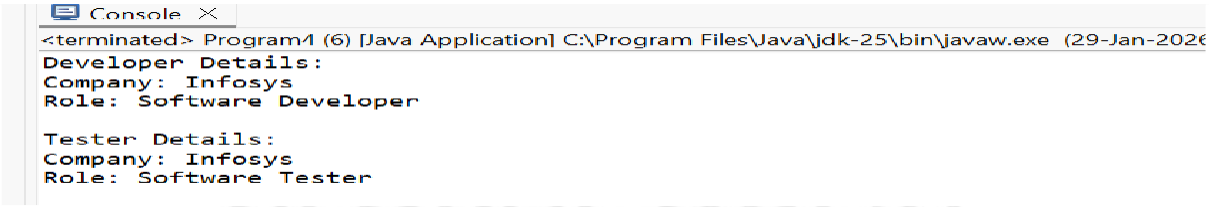
OUTPUT:

4) Write a program to demonstrate inheritance by creating objects of derived classes and invoking base class methods.

```
package Inheritance;
class Employee {
String company;
Employee() {
    company = "Infosys";
 }
 void displayCompany() {
```

```
        System.out.println("Company: " + company);
    }
}
class Developer extends Employee {
    String role = "Software Developer";
    void displayDeveloper() {
        displayCompany();
        System.out.println("Role: " + role);
    }
}
class Tester extends Employee {
    String role = "Software Tester";
    void displayTester() {
        displayCompany();
        System.out.println("Role: " + role);
    }}
public class Program4 {
    public static void main(String[] args) {
        Developer d = new Developer();
        System.out.println("Developer Details:");
        d.displayDeveloper();
        System.out.println();
        Tester t = new Tester();
        System.out.pri ntln("Tester Details:");
        t.displayTester();
    }}
```

OUTPUT:



5) Create a base class "Animal" with properties (name, age) and subclasses "Dog" and "Cat" with additional properties (breed, color).

```
package Inheritance;
class Animal {
String name;
    int age;
    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }
    void displayAnimal() {
        System.out.println("Name : " + name);
        System.out.println("Age : " + age);
    }
}
class Dog extends Animal {
    String breed;
```

```
 Dog(String name, int age, String breed) {
    super(name, age);
    this.breed = breed;
 }
 void displayDog() {
    displayAnimal();
    System.out.println("Breed : " + breed);
 }
}
class Cat extends Animal {
 String color;
 Cat(String name, int age, String color) {
    super(name, age);
    this.color = color;
 }
 void displayCat() {
    displayAnimal();
    System.out.println("Color : " + color);
 }
}
public class Program5 {
 public static void main(String[] args) {
    Dog d = new Dog("Bruno", 3, "German Shepherd");
    System.out.println("Dog Details:");
    d.displayDog();
    System.out.println();
    Cat c = new Cat("Kitty", 2, "White");
    System.out.println("Cat Details:");
    c.displayCat();
 }}
```

OUTPUT:

```
<terminated> Program5 (3) [Java Application] C:\Program Files\Java\jdk-25\bi
Dog Details:
Name : Bruno
Age : 3
Breed : German Shepherd

Cat Details:
Name : Kitty
Age : 2
Color : White
```

6) Write a program to demonstrate inheritance by creating objects of derived classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
```

```
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

7) Create a base class Employee with properties (name, id, salary) and a subclass Manager with additional properties (department, designation).

```
package Inheritance;
class Employee1 {
   String name;
   int id;
   double salary;
}
class Manager extends Employee1{
   String department;
   String designation;
}
class Program6 {
   public static void main(String[] args) {
      Employee1 e = new Employee1();
      e.name = "Ravi";
      e.id = 101;
      e.salary = 30000;
      System.out.println("Employee Details:");
      System.out.println(e.name + " " + e.id + " " + e.salary);
      Manager m = new Manager();
      m.name = "Bharathi";
      m.id = 102;
      m.salary = 60000;
```

```
        m.department = "IT";
        m.designation = "Manager";
        System.out.println("Manager Details:");
        System.out.println(m.name + " " + m.id + " " + m.salary);
        System.out.println(m.department + " " + m.designation);
    }
}
```

OUTPUT:

```
Employee Details:
Ravi 101 30000.0
Manager Details:
Bharathi 102 60000.0
IT Manager
```

8) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

9) Create a base class Person with properties (name, age) and subclasses Student and Teacher with additional properties (roll number, subject).

```
package Inheritance;
class Person3 {
   String name;
   int age;
}
class Student3 extends Person3 {
   int rollNo;
}
class Teacher extends Person3 {
   String subject;
}
class Program9 {
   public static void main(String[] args) {
      Student s = new Student();
      s.name = "Anu";
      s.age = 20;
      s.rollNo = 101;
      Teacher t = new Teacher();
      t.name = "Mr. Rao";
      t.age = 45;
      t.subject = "Java";
      System.out.println(s.name + " " + s.age + " " + s.rollNo);
      System.out.println(t.name + " " + t.age + " " + t.subject);
   }
}
```

OUTPUT:

```
<terminated> Program9 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (29-Jan-2026, 9
Anu 20 101
Mr. Rao 45 Java
```

10) Write a program to demonstrate inheritance by creating objects of derived classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
```

```
        a.color = "Red";
        a.taste = "Sweet";
        Banana b = new Banana();
        b.name = "Banana";
        b.color = "Yellow";
        b.size = "Large";
        System.out.println("Apple Details:");
        System.out.println(a.name + " " + a.color + " " + a.taste);
        System.out.println();
        System.out.println("Banana Details:");
        System.out.println(b.name + " " + b.color + " " + b.size);
    }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

11) Create a base class BankAccount with properties (account number, balance) and subclasses SavingsAccount and CurrentAccount.

```
package Inheritance;
class BankAccount {
    int accountNumber;
    double balance;
}
class SavingsAccount extends BankAccount {
    double interestRate;
}
class CurrentAccount extends BankAccount {
    double overdraftLimit;
}
class Program11 {
    public static void main(String[] args) {
        SavingsAccount s = new SavingsAccount();
        s.accountNumber = 111;
        s.balance = 5000;
        s.interestRate = 4.5;
        CurrentAccount c = new CurrentAccount();
        c.accountNumber = 222;
        c.balance = 10000;
        c.overdraftLimit = 2000;
        System.out.println(s.accountNumber + " " + s.balance + " " + s.interestRate);
        System.out.println(c.accountNumber + " " + c.balance + " " + c.overdraftLimit);
    }
}
```

OUTPUT:

```
<terminated> Program11 (5) [Java Application] C:\Program Files\Java\jdk-25\bin
111 5000.0 4.5
222 10000.0 2000.0
```

12) Write a program to demonstrate inheritance by creating objects of derived classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
     Apple a = new Apple();
     a.name = "Apple";
     a.color = "Red";
     a.taste = "Sweet";
     Banana b = new Banana();
     b.name = "Banana";
     b.color = "Yellow";
     b.size = "Large";
     System.out.println("Apple Details:");
     System.out.println(a.name + " " + a.color + " " + a.taste);
     System.out.println();
     System.out.println("Banana Details:");
     System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

13) Create a base class Shape with properties (type, color) and a subclass Triangle with additional properties (base, height).

```
package Inheritance;
class Shape1 {
   String type;
   String color;
}
```

```
class Triangle extends Shape1 {
   double base;
   double height;
}
class Program13 {
   public static void main(String[] args) {
      Shape1 s = new Shape1();
      s.type = "Generic Shape";
      s.color = "Red";
      Triangle t = new Triangle();
      t.type = "Triangle";
      t.color = "Blue";
      t.base = 10;
      t.height = 5;
      System.out.println(s.type + " " + s.color);
      System.out.println(t.type + " " + t.color + " " + t.base + " " + t.height);
   }
}
```

OUTPUT:

```
<terminated> Program13 (5) [Java Application] C:\Program Files\Java\jdk-25\bin\j
Generic Shape Red
Triangle Blue 10.0 5.0
```

14) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
```

```
        System.out.println("Banana Details:");
        System.out.println(b.name + " " + b.color + " " + b.size);
    }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

15) Create a base class Vehicle with properties (make, model, year) and a subclass Truck with additional properties (capacity, mileage).

```
package Inheritance;
class Vehicle1 {
    String make;
    String model;
    int year;
}
class Truck extends Vehicle1{
    int capacity;
    int mileage;
}
class Program15 {
    public static void main(String[] args) {
        Vehicle1 v = new Vehicle1();
        v.make = "Tata";
        v.model = "Ace";
        v.year = 2020;
        Truck t = new Truck();
        t.make = "Ashok Leyland";
        t.model = "Dost";
        t.year = 2022;
        t.capacity = 2000;
        t.mileage = 15;
        System.out.println(v.make + " " + v.model + " " + v.year);
        System.out.println(t.make + " " + t.model + " " + t.year + " " + t.capacity + " " +
t.mileage);
    }
}
```

OUTPUT:

```
<terminated> Program15 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (29-Jan-2026,
Tata Ace 2020
Ashok Leyland Dost 2022 2000 15
```

16) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
```

```
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

17) Create a base class Fruit with properties (name, color) and subclasses Apple and Banana with additional properties (taste, size).

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
```

```
        a.taste = "Sweet";
        Banana b = new Banana();
        b.name = "Banana";
        b.color = "Yellow";
        b.size = "Large";
        System.out.println("Apple Details:");
        System.out.println(a.name + " " + a.color + " " + a.taste);
        System.out.println();
        System.out.println("Banana Details:");
        System.out.println(b.name + " " + b.color + " " + b.size);
    }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

18) Write a program to demonstrate inheritance by creating objects of derived classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

19) Create a base class Animal with properties (name, type) and subclasses Dog and Cat with additional properties (breed, color).

```
package Inheritance;
class Animal1 {
   String name;
   String type;
}
class Dog1 extends Animal1 {
   String breed;
}

class Cat1 extends Animal1 {
   String color;
}
class Program19 {
   public static void main(String[] args) {
      Dog1 d = new Dog1();
      d.name = "Bruno";
      d.type = "Pet";
      d.breed = "German Shepherd";
      Cat1 c = new Cat1();
      c.name = "Kitty";
      c.type = "Pet";
      c.color = "White";
      System.out.println(d.name + " " + d.type + " " + d.breed);
      System.out.println(c.name + " " + c.type + " " + c.color);
   }
}
```

OUTPUT:

```
Bruno Pet German Shepherd
Kitty Pet White
```

20) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
```

```
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

21) Create a base class Person with properties (name, age) and a subclass Employee with additional properties (id, salary).

```
class Person1 {
   String name;
   int age;
}
class Employee2 extends Person1 {
   int id;
   double salary;
}
class Program9 {
   public static void main(String[] args) {
      Person p = new Person();
      p.name = "Ravi";
      p.age = 35;
      Employee2 e = new Employee2();
      e.name = "Bharathi";
      e.age = 22;
      e.id = 101;
      e.salary = 25000;
      System.out.println(p.name + " " + p.age);
      System.out.println(e.name + " " + e.age + " " + e.id + " " + e.salary);
   }
```

```
}
```

OUTPUT:

```
<terminated> Program9 (4) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe  (29
Ravi 35
Bharathi 22 101 25000.0
```

22) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
     Apple a = new Apple();
     a.name = "Apple";
     a.color = "Red";
     a.taste = "Sweet";
     Banana b = new Banana();
     b.name = "Banana";
     b.color = "Yellow";
     b.size = "Large";
     System.out.println("Apple Details:");
     System.out.println(a.name + " " + a.color + " " + a.taste);
     System.out.println();
     System.out.println("Banana Details:");
     System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

23) Create a base class Shape with properties (type, color) and a subclass Rectangle with additional properties (length, width).

```
package Inheritance;
class Shape2 {
   String type;
   String color;
}
class Rectangle1 extends Shape2 {
   int length;
```

```
    int width;
}
class Program23 {
    public static void main(String[] args) {
        Shape2 s = new Shape2();
        s.type = "Shape";
        s.color = "Green";

        Rectangle1 r = new Rectangle1();
        r.type = "Rectangle";
        r.color = "Blue";
        r.length = 10;
        r.width = 5;
        System.out.println(s.type + " " + s.color);
        System.out.println(r.type + " " + r.color + " " + r.length + " " + r.width);
    }
}
```

OUTPUT:

```
Shape Green
Rectangle Blue 10 5
```

24) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
     Apple a = new Apple();
     a.name = "Apple";
     a.color = "Red";
     a.taste = "Sweet";
     Banana b = new Banana();
     b.name = "Banana";
     b.color = "Yellow";
     b.size = "Large";
     System.out.println("Apple Details:");
     System.out.println(a.name + " " + a.color + " " + a.taste);
     System.out.println();
     System.out.println("Banana Details:");
```

```
        System.out.println(b.name + " " + b.color + " " + b.size);
    }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

25) Create a base class Vehicle with properties (make, model, year) and a subclass Car with additional properties (color, mileage).

```
package Inheritance;
class Vehicle1 {
    String make;
    String model;
    int year;
}
class Car extends Vehicle {
    String color;
    int mileage;
}
class Program25 {
    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.make = "Honda";
        v.model = "City";
        v.year = 2021;
        Car c = new Car();
        c.make = "Toyota";
        c.model = "Innova";
        c.year = 2023;
        c.color = "White";
        c.mileage = 18;
        System.out.println(v.make + " " + v.model + " " + v.year);
        System.out.println(c.make + " " + c.model + " " + c.year + " " + c.color + " " +
c.mileage);
    }
}
```

OUTPUT:

```
<terminated> Program25 (3) [Java Application] C:\Program Files
Honda City 2021
Toyota Innova 2023 White 18
```

26) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
```

```
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
     Apple a = new Apple();
     a.name = "Apple";
     a.color = "Red";
     a.taste = "Sweet";
     Banana b = new Banana();
     b.name = "Banana";
     b.color = "Yellow";
     b.size = "Large";
     System.out.println("Apple Details:");
     System.out.println(a.name + " " + a.color + " " + a.taste);
     System.out.println();
     System.out.println("Banana Details:");
     System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

27) Create a base class Animal with properties (name, type) and a subclass Bird with additional properties (color, wingspan).

```
package Inheritance;
class Animal2{
   String name;
   String type;
}
class Bird extends Animal2 {
   String color;
   double wingspan;
}
class Program27 {
   public static void main(String[] args) {
      Animal2 a = new Animal2();
      a.name = "Generic";
      a.type = "Animal";
      Bird b = new Bird();
      b.name = "Parrot";
```

```
        b.type = "Bird";
        b.color = "Green";
        b.wingspan = 0.5;
        System.out.println(a.name + " " + a.type);
        System.out.println(b.name + " " + b.type + " " + b.color + " " + b.wingspan);
    }
}
```

OUTPUT:

```
<terminated> Program27 [Java Application] C:\Program Files\Java\jdk-25\bin\javaw.exe (
Generic Animal
Parrot Bird Green 0.5
```

28) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

29) Create a base class Shape with properties (type, color) and a subclass Circle with additional properties (radius, area).

```
package Inheritance;
```

```
class Shape3 {
 String type;
 String color;
}
class Circle1 extends Shape3 {
 double radius;
 double area;}
public class Program29 {
 public static void main(String[] args) {
    Shape3 s = new Shape3();
    s.type = "Shape";
    s.color = "Blue";
    Circle1 c = new Circle1();
    c.type = "Circle";
    c.color = "Red";
    c.radius = 5;
    c.area = 3.14 * c.radius * c.radius;
    System.out.println("Shape Details:");
    System.out.println(s.type + " " + s.color);
    System.out.println("\nCircle Details:");
    System.out.println(c.type + " " + c.color + " " + c.radius + " " + c.area);
 }}
```

OUTPUT:

```
Shape Details:
Shape Blue

Circle Details:
Circle Red 5.0 78.5
```

30) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties.

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
```

```
        b.name = "Banana";
        b.color = "Yellow";
        b.size = "Large";
        System.out.println("Apple Details:");
        System.out.println(a.name + " " + a.color + " " + a.taste);
        System.out.println();
        System.out.println("Banana Details:");
        System.out.println(b.name + " " + b.color + " " + b.size);
    }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

31) Create a base class Employee with properties (name, id, salary) and a subclass Manager with additional properties (department, designation).

```
package Inheritance;
class Employee3 {
String name;
 int id;
 double salary;
}
class Manager1 extends Employee3{
 String department;
 String designation;
}
public class Program31 {
 public static void main(String[] args) {
    Employee3 e = new Employee3();
    e.name = "Ravi";
    e.id = 101;
    e.salary = 30000;
    Manager m = new Manager();
    m.name = "Bharathi";
    m.id = 102;
m.salary = 60000;
    m.department = "IT";
    m.designation = "Manager";
    System.out.println("Employee Details:");
    System.out.println(e.name + " " + e.id + " " + e.salary);
    System.out.println("\nManager Details:");
    System.out.println(m.name + " " + m.id + " " + m.salary);
    System.out.println(m.department + " " + m.designation);
 }
}
```

OUTPUT:

```
Employee Details:
Ravi 101 30000.0

Manager Details:
Bharathi 102 60000.0
IT Manager
```

32) Write a program to demonstrate inheritance by creating objects of both classes and accessing properties

```
package Inheritance;
class Fruit {
 String name;
 String color;
}
class Apple extends Fruit {
 String taste;
}
class Banana extends Fruit {
 String size;
}
public class Program6 {
 public static void main(String[] args) {
    Apple a = new Apple();
    a.name = "Apple";
    a.color = "Red";
    a.taste = "Sweet";
    Banana b = new Banana();
    b.name = "Banana";
    b.color = "Yellow";
    b.size = "Large";
    System.out.println("Apple Details:");
    System.out.println(a.name + " " + a.color + " " + a.taste);
    System.out.println();
    System.out.println("Banana Details:");
    System.out.println(b.name + " " + b.color + " " + b.size);
 }
}
```

OUTPUT:

```
Apple Details:
Apple Red Sweet

Banana Details:
Banana Yellow Large
```

Exception Handling

1) Write a program to handle ArrayIndexOutOfBoundsException.

```
package Exceptions;
class Program1 {
```

```
    public static void main(String[] args) {
        try {
            int[] a = {1, 2, 3};
            System.out.println(a[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds");
        }
    }
}
```

OUTPUT:

```
<terminated> Program1 (6) [Java Application] C:\Program Files\Java\jdk-23\bin\jav
Array index out of bounds
```

2) Implement a program to handle ArithmeticException such as division by zero.

```
package Exceptions;

class Program2 {

    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero");
        }
    }
}
```

OUTPUT:

```
<terminated> Program2 (6) [Java Application] C:\Program Files\Java
Cannot divide by zero
```

3) Write a program to handle NullPointerException.

```
package Exceptions;
class Program3 {
    public static void main(String[] args) {
        try {
            String s = null;
            System.out.println(s.length());
        } catch (NullPointerException e) {
            System.out.println("Null reference accessed");
        }
    }
}
```

OUTPUT:

```
<terminated> Program3 (7) [Java Application] C:\Progra
Null reference accessed
```

4) Implement a program to handle FileNotFoundException.

```
package Exceptions;
import java.io.*;
class Program4 {
   public static void main(String[] args) {
      try {
         FileReader fr = new FileReader("abc.txt");
      } catch (FileNotFoundException e) {
         System.out.println("File not found");
      }
   }
}
```

OUTPUT:

```
<terminated> Program4 (7) [Java Application] C:\Program Files
File not found
```

5) Write a program to handle NumberFormatException.

```
package Exceptions;
class Program5 {
   public static void main(String[] args) {
      try {
         int n = Integer.parseInt("abc");
      } catch (NumberFormatException e) {
         System.out.println("Invalid number format");
      }   }
}
```

OUTPUT:

```
<terminated> Program5 (4) [Java Application] C:\Program Files
Invalid number format
```

6) Implement a program to handle IOException.

```
package Exceptions;
import java.io.*;
class Program6 {
   public static void main(String[] args) {
      try {
         throw new IOException();
      } catch (IOException e) {
         System.out.println("IO Exception handled");
      }
   }
}
```

OUTPUT:

```
IO Exception handled
```

7) Write a program to handle ClassNotFoundException.

```
package Exceptions;
class Program7 {
   public static void main(String[] args) {
      try {
         Class.forName("Test");
      } catch (ClassNotFoundException e) {
         System.out.println("Class not found");
      }   }
}
```

OUTPUT:

```
Class not found
```

8) Implement a program to handle StackOverflowError.

```
package Exceptions;
class Program8 {
   static void display() {
      display();
   }
   public static void main(String[] args) {
      try {
         display();
      } catch (StackOverflowError e) {
         System.out.println("Stack overflow error");
      }   }}
```

OUTPUT:

```
Stack overflow error
```

9) Write a program to handle NegativeArraySizeException.

```
package Exceptions;
class Program9 {
   public static void main(String[] args) {
      try {
         int[] a = new int[-5];
      } catch (NegativeArraySizeException e) {
         System.out.println("Negative array size");
      }
   }
}
```

OUTPUT:

```
Negative array size
```

10) Implement a program to handle InterruptedException.

```
package Exceptions;
class Program10 {
   public static void main(String[] args) {
      try {
         Thread.sleep(100);
         System.out.println("Thread woke up after 1 second");

      } catch (InterruptedException e) {
         System.out.println("Thread interrupted");
      }
   }
}
```

OUTPUT:

```
<terminated> Program10 (6) [Java Application] C:\Program Files\Java\jdk-25\bin\
Thread woke up after 1 second
```

11) Write a program to handle ArrayStoreException.

```
package Exceptions;
class Program11 {
   public static void main(String[] args) {
      try {
         Object[] arr = new Integer[5];
         arr[0] = "Hello";
      } catch (ArrayStoreException e) {
         System.out.println("Invalid array store");
      }   }}
```

OUTPUT:

```
<terminated> Program11 (6) [Java Application] C:\Program Files\Java\jd
Invalid array store
```

12) Implement a program to handle IllegalStateException.

```
package Exceptions;
import java.util.*;
class Program12 {
   public static void main(String[] args) {
      try {
         Iterator<Integer> it = new ArrayList<Integer>().iterator();
         it.remove();
      } catch (IllegalStateException e) {
         System.out.println("Illegal state");
      }   }}
```

OUTPUT:

```
Illegal state
```

13) Write a program to handle NoSuchElementException.

```
package Exceptions;
import java.util.*;
class Program13 {
   public static void main(String[] args) {
      try {
         Iterator<Integer> it = new ArrayList<Integer>().iterator();
         it.next();
      } catch (NoSuchElementException e) {
         System.out.println("No element found");
      }
   }
}
```

OUTPUT:

```
<terminated> Program13 (6) [Java Application] C:\Program Files\Jav
No element found
```

14) Implement a program to handle UnsupportedOperationException.

```
package Exceptions;
import java.util.*;
class Program14 {
   public static void main(String[] args) {
      try {
         List<Integer> list = List.of(1,2,3);
         list.add(4);
      } catch (UnsupportedOperationException e) {
         System.out.println("Operation not supported");
      }
   }
}
```

OUTPUT:

```
<terminated> Program14 (5) [Java Application] C:\Program Files\Ja
Operation not supported
```

15) Write a program to handle UnsupportedOperationException.

```
package Exceptions;
import java.util.*;
class Program14 {
   public static void main(String[] args) {
      try {
         List<Integer> list = List.of(1,2,3);
         list.add(4);
      } catch (UnsupportedOperationException e) {
         System.out.println("Operation not supported");
      }
```

```
    }
}
```

OUTPUT:

```
<terminated> Program14 (5) [Java Application] C:\Program Files\Ja
Operation not supported
```

16) Implement a program to handle ConcurrentModificationException.

```
package Exceptions;
import java.util.*;
class Program15 {
    public static void main(String[] args) {
        try {
            ArrayList<Integer> list = new ArrayList<>();
            list.add(1);
            for(Integer i : list) {
                list.add(2);
            }
        } catch (ConcurrentModificationException e) {
            System.out.println("Concurrent modification");
        }   }}
```

OUTPUT:

```
<terminated> Program15 (5) [Java Application] C:\Program Files\Java\
Concurrent modification
```

17) Write a program to handle IllegalArgumentException.

```
package Exceptions;
class Program16 {
    static void check(int age) {
        if(age < 18)
            throw new IllegalArgumentException();
    }
    public static void main(String[] args) {
        try {
            check(15);
        } catch (IllegalArgumentException e) {
            System.out.println("Invalid argument");
        }   }
}
```

OUTPUT:

```
<terminated> Program16 (4) [Java Application] C:\Program Files\
Invalid argument
```

18) Implement a program to handle SecurityException.

```
package Exceptions;
class Program17 {
    public static void main(String[] args) {
```

```
        try {
            throw new SecurityException();
        } catch (SecurityException e) {
            System.out.println("Security exception");
        }
    }
}
```

OUTPUT:

```
<terminated> Program17 (6) [Java Application] C:\Program Files\Java\jc
Security exception
```

19) Write a program to handle DateTimeParseException

```
package Exceptions;
import java.time.*;
import java.time.format.*;
class Program18 {
    public static void main(String[] args) {
        try {
            LocalDate.parse("abc");
        } catch (DateTimeParseException e) {
            System.out.println("Invalid date format");
        }
    }
}
```

OUTPUT:

```
<terminated> Program18 (4) [Java Application] C:\Program
Invalid date format
```

20) Implement a program to handle PatternSyntaxException.

```
package Exceptions;
import java.util.regex.*;
class Program19 {
    public static void main(String[] args) {
        try {
            Pattern.compile("[");
        } catch (PatternSyntaxException e) {
            System.out.println("Pattern syntax error");
        }
    }
}
```

OUTPUT:

```
<terminated> Program19 (6) [Java Application] C:\Program Files
Pattern syntax error
```

21) Write a program to handle MissingResourceException.

```
package Exceptions;
import java.util.*;
class Program20 {
```

```
    public static void main(String[] args) {
        try {
            ResourceBundle.getBundle("test");
        } catch (MissingResourceException e) {
            System.out.println("Resource missing");
        }
    }}
```

OUTPUT:

```
<terminated> Program20 (4) [Java Application] C:\Program Files\Java\jdk-25\
Resource missing
```

22) Implement a program to handle FormatterClosedException.

```
package Exceptions;
import java.util.*;
class Program21 {
    public static void main(String[] args) {
        try {
            Formatter f = new Formatter();
            f.close();
            f.format("Hello");
        } catch (FormatterClosedException e) {
            System.out.println("Formatter closed");
        }   }}
```

OUTPUT:

```
<terminated> Program21 (5) [Java Application] C:\Program F
Formatter closed
```

23) Write a program to handle BufferOverflowException.

```
package Exceptions;
import java.nio.*;

class Program22 {
    public static void main(String[] args) {
        try {
            ByteBuffer buffer = ByteBuffer.allocate(2);
            buffer.put((byte)1);
            buffer.put((byte)2);
            buffer.put((byte)3);
        } catch (BufferOverflowException e) {
            System.out.println("Buffer overflow");
        }
    }
}
```

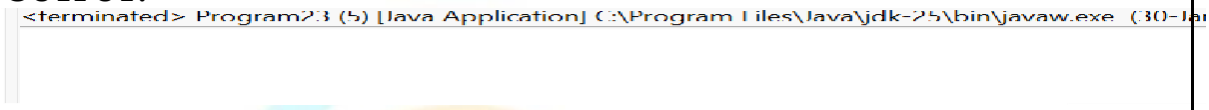
OUTPUT:

```
Buffer overflow
```

24) Implement a program to handle BufferUnderflowException.

```
package Exceptions;
import java.nio.*;
class Program23 {
   public static void main(String[] args) {
      try {
         ByteBuffer buffer = ByteBuffer.allocate(6);
         buffer.get();
      } catch (BufferUnderflowException e) {
         System.out.println("Buffer underflow");
      }
   }
}
```

OUTPUT:

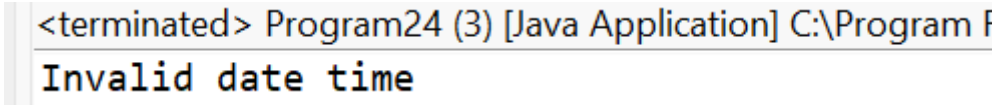


25) Write a program to handle DateTimeException.

```
package Exceptions;
import java.time.*;

class Program24 {
   public static void main(String[] args) {
      try {
         LocalDate.of(2025, 13, 10);
      } catch (DateTimeException e) {
         System.out.println("Invalid date time");
      }
   }
}
```

OUTPUT:



Interface:

1) Create interfaces Drawable and Resizable with methods draw and resize. Implement them in a class representing a shape.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
```

```
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

2) Write a program to demonstrate interface implementation by creating objects of the shape class and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

3) Create interfaces Flyable and Swimmable with methods fly and swim. Implement them in classes representing a bird and a fish.

```
package Interfaces;
interface Flyable {
   void fly();
}
interface Swimmable {
   void swim();
}
class Bird implements Flyable {
   public void fly() {
      System.out.println("Bird is flying");
   }
}
class Program2 implements Swimmable {
   public void swim() {
      System.out.println("Fish is swimming");
   }
   public static void main(String[] args) {
      Bird b = new Bird();
      Program2 f = new Program2();
      b.fly();
      f.swim();
   }
}
```

OUTPUT:

```
<terminated> Program2 (7) [Java Application] C:\Program Files\Java\jdk
Bird is flying
Fish is swimming
```

4) Write a program to demonstrate interface implementation by creating objects of the bird and fish classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
```

```
            Program1 s = new Program1();
            s.draw();
            s.resize();
        }
    }
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

5) Create interfaces Comparable and Cloneable with methods compareTo and clone. Implement them in classes representing a number and a person

```
package Interfaces;
interface MyComparable {
   int compareTo(int x);
}
interface MyCloneable {
   Object clone();
}
class MyNumber implements MyComparable, MyCloneable {
   int n;
   MyNumber(int n) {
      this.n = n;
   }
   public int compareTo(int x) {
      return n - x;
   }
   public Object clone() {
      return new MyNumber(n);
   }
}
class Person implements MyComparable, MyCloneable {
   int age;
   Person(int age) {
      this.age = age;
   }
   public int compareTo(int x) {
      return age - x;
   }
   public Object clone() {
      return new Person(age);
   }
}
public class Program5 {
   public static void main(String[] args) {
      MyNumber num = new MyNumber(10);
      Person p = new Person(25);
      System.out.println(num.compareTo(5));
```

```
        System.out.println(p.compareTo(30));
        MyNumber n2 = (MyNumber) num.clone();
        Person p2 = (Person) p.clone();
        System.out.println("Cloned objects created");
    }
}
```

OUTPUT:

```
<terminated> Program5 (5) [Java Application] C:\Program Files\Java\jdk-25\bin\javaw
5
-5
Cloned objects created
```

6) Write a program to demonstrate interface implementation by creating objects of the number and person classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
    void resize();
}
class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

7) Create interfaces List and Set with methods add, remove, and contains. Implement them in classes representing an array list and a hash set.

```
package Interfaces;
interface MyList {
    void add(int x);
    void remove(int x);
    boolean contains(int x);
}
```

```
interface MySet {
    void add(int x);
    void remove(int x);
    boolean contains(int x);
}
class MyArrayList implements MyList {
    int[] arr = new int[5];
    int size = 0;
    public void add(int x) {
        arr[size++] = x;
    }
    public void remove(int x) {
        size--;
    }
    public boolean contains(int x) {
        for(int i=0;i<size;i++)
            if(arr[i]==x) return true;
        return false;
    }
}
class MyHashSet implements MySet {
    int[] arr = new int[5];
    int size = 0;
    public void add(int x) {
        arr[size++] = x;
    }
    public void remove(int x) {
        size--;
    }
    public boolean contains(int x) {
        for(int i=0;i<size;i++)
            if(arr[i]==x) return true;
        return false;
    }
}
public class Program7 {
    public static void main(String[] args) {
        MyArrayList list = new MyArrayList();
        MyHashSet set = new MyHashSet();
        list.add(10);
        set.add(20);
        System.out.println(list.contains(10));
        System.out.println(set.contains(20));
    }
}
```

OUTPUT:

```
<terminated> Program7 (7) [Java Application] C:\Program Files\Java\jdk-25\bi
true
true
```

8) **Write a program to demonstrate interface implementation by creating objects of the array list and hash set classes and invoking interface methods.**

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

9) **Create interfaces Printable and Scannable with methods print and scan. Implement them in classes representing a printer and a scanner.**

```
package Interfaces;
interface Printable {
   void print();
}
interface Scannable {
   void scan();
}
class Printer implements Printable {
   public void print() {
      System.out.println("Printing document");
   }
}
class ScannerMachine implements Scannable {
   public void scan() {
      System.out.println("Scanning document");
   }
}
public class Program9 {
   public static void main(String[] args) {
```

```
        Printer p = new Printer();
        ScannerMachine s = new ScannerMachine();
        p.print();
        s.scan();
    }
}
```

OUTPUT

```
Printing document
Scanning document
```

10) Write a program to demonstrate interface implementation by creating objects of the printer and scanner classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
    void resize();
}
class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

11) Create interfaces Sortable and Searchable with methods sort and search. Implement them in classes representing a list and a dictionary.

```
package Interfaces;
interface Sortable {
    void sort();
}
interface Searchable {
    void search();
}
```

```
class MyListData implements Sortable {
    public void sort() {
        System.out.println("List sorted");
    }
}
class Dictionary implements Searchable {
    public void search() {
        System.out.println("Word searched in dictionary");
    }
}
public class Program11 {
    public static void main(String[] args) {
        MyListData l = new MyListData();
        Dictionary d = new Dictionary();
        l.sort();
        d.search();
    }
}
```

OUTPUT:

```
<terminated> Program11 (7) [Java Application] C:\Program Files\Java\jdk-25\bin\ja
List sorted
Word searched in dictionary
```

12) Write a program to demonstrate interface implementation by creating objects of the list and dictionary classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
    void resize();
}
class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

13) Create interfaces Serializable and Deserializable with methods serialize and deserialize. Implement them in classes representing a file and a database.

```
package Interfaces;
interface Serializable {
   void serialize();
}
interface Deserializable {
   void deserialize();
}
class FileData implements Serializable {
   public void serialize() {
      System.out.println("File serialized");
   }
}
class DatabaseData implements Deserializable {
   public void deserialize() {
      System.out.println("Database deserialized");
   }
}
public class Program13 {
   public static void main(String[] args) {
      FileData f = new FileData();
      DatabaseData d = new DatabaseData();
      f.serialize();
      d.deserialize();
   }
}
```

OUTPUT:

```
File serialized
Database deserialized
```

14) Write a program to demonstrate interface implementation by creating objects of the file and database classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
```

```
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }
}
```

OUTPUT:

15) Create interfaces Encryptable and Decryptable with methods encrypt and decrypt. Implement them in classes representing an encoder and a decoder.

```
package Interfaces;
interface Encryptable {
    void encrypt();
}
interface Decryptable {
    void decrypt();
}
class Encoder implements Encryptable {
    public void encrypt() {
        System.out.println("Data encrypted");
    }
}
class Decoder implements Decryptable {
    public void decrypt() {
        System.out.println("Data decrypted");
    }
}
public class Program15 {
    public static void main(String[] args) {
        Encoder e = new Encoder();
        Decoder d = new Decoder();

        e.encrypt();
        d.decrypt();
    }
}
```

OUTPUT:

16) Write a program to demonstrate interface implementation by creating objects of the encoder and decoder classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

17) Create interfaces Runnable and Walkable with methods run and walk. Implement them in classes representing a cheetah and a tortoise.

```
package Interfaces;
interface Runnable {
   void run();
}
interface Walkable {
   void walk();
}
class Cheetah implements Runnable {
   public void run() {
      System.out.println("Cheetah is running fast");
   }
}
class Tortoise implements Walkable {
   public void walk() {
      System.out.println("Tortoise is walking slowly");
   }
}
public class Program17 {
   public static void main(String[] args) {
      Cheetah c = new Cheetah();
      Tortoise t = new Tortoise();
```

```
        c.run();
        t.walk();
    }
}
```

OUTPUT:

```
Cheetah is running fast
Tortoise is walking slowly
```

18) Write a program to demonstrate interface implementation by creating objects of the cheetah and tortoise classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
    void resize();
}

class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

19) Create interfaces Playable and Recordable with methods play and record. Implement them in classes representing a music player and a recorder.

```
package Interfaces;
interface Playable {
    void play();
}
interface Recordable {
    void record();
}
```

```
class MusicPlayer implements Playable {
    public void play() {
        System.out.println("Playing music");
    }
}
class Recorder implements Recordable {
    public void record() {
        System.out.println("Recording audio");
    }
}
public class Program19 {
    public static void main(String[] args) {
        MusicPlayer m = new MusicPlayer();
        Recorder r = new Recorder();
        m.play();
        r.record();
    }
}
```

OUTPUT:

```
Playing music
Recording audio
```

20) Write a program to demonstrate interface implementation by creating objects of the music player and recorder classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
    void resize();
}
class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

21) Create interfaces Drawable and Erasable with methods draw and erase. Implement them in classes representing a whiteboard and a chalkboard.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Erasable {
   void erase();
}
class WhiteBoard implements Drawable {
   public void draw() {
      System.out.println("Drawing on whiteboard");
   }
}
class ChalkBoard implements Erasable {
   public void erase() {
      System.out.println("Erasing chalkboard");
   }
}
public class Program21 {
   public static void main(String[] args) {
      WhiteBoard w = new WhiteBoard();
      ChalkBoard c = new ChalkBoard();
      w.draw();
      c.erase();
   }
}
```

OUTPUT:

```
Drawing on whiteboard
Erasing chalkboard
```

the whiteboard and chalkboard classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
```

```
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

23) Create interfaces Sendable and Receivable with methods send and receive. Implement them in classes representing a transmitter and a receiver.

```
package Interfaces;
interface Sendable {
    void send();
}
interface Receivable {
    void receive();
}
class Transmitter implements Sendable {
    public void send() {
        System.out.println("Sending signal");
    }
}class Receiver implements Receivable {
    public void receive() {
        System.out.println("Receiving signal");
    }
}
public class Program23 {
    public static void main(String[] args) {
        Transmitter t = new Transmitter();
        Receiver r = new Receiver();
        t.send();
        r.receive();
    }
}
```

OUTPUT:

```
<terminated> Program23 (6) [Java Application] C:\Program Files\Java
Sending signal
Receiving signal
```

24) Write a program to demonstrate interface implementation by creating objects of the transmitter and receiver classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
```

```
    void resize();
}
class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

25) Create interfaces Encryptable and Decryptable with methods encrypt and decrypt. Implement them in classes representing an encryption algorithm and a decryption algorithm.

```
package Interfaces;
class EncryptionAlgorithm implements Encryptable {
    public void encrypt() {
        System.out.println("Algorithm encrypting data");
    }
}
class DecryptionAlgorithm implements Decryptable {
    public void decrypt() {
        System.out.println("Algorithm decrypting data");
    }
}
public class Program25 {
    public static void main(String[] args) {
        EncryptionAlgorithm e = new EncryptionAlgorithm();
        DecryptionAlgorithm d = new DecryptionAlgorithm();
        e.encrypt();
        d.decrypt();
    }
}
```

OUTPUT:

```
Algorithm encrypting data
Algorithm decrypting data
```

26) Write a program to demonstrate interface implementation by creating objects of the encryption and decryption classes and invoking interface methods.

```
package Interfaces;
```

```
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

27) Create interfaces Writable and Readable with methods write and read. Implement them in classes representing a text file and a database table.

```
package Interfaces;
interface Writable {
   void write();
}
interface Readable {
   void read();
}
class TextFile implements Writable {
   public void write() {
      System.out.println("Writing to text file");
   }
}
class DatabaseTable implements Readable {
   public void read() {
      System.out.println("Reading from database table");
   }
}
public class Program27 {
   public static void main(String[] args) {
      TextFile t = new TextFile();
      DatabaseTable d = new DatabaseTable();
      t.write();
      d.read();
```

```
}}
```

OUTPUT:

```
<terminated> Program27 (1) [Java Application] C:\Program File
Writing to text file
Reading from database table
```

28) Write a program to demonstrate interface implementation by creating objects of the text file and database table classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

29) Create interfaces Drawable and Printable with methods draw and print. Implement them in classes representing a canvas and a printer.

```
package Interfaces;
class Canvas implements Drawable {
   public void draw() {
      System.out.println("Drawing on canvas");
   }
}
class OfficePrinter implements Printable {
   public void print() {
      System.out.println("Printing from printer");
   }
}
public class Program29 {
```

```
    public static void main(String[] args) {
        Canvas c = new Canvas();
        OfficePrinter p = new OfficePrinter();
        c.draw();
        p.print();
    }
}
```

OUTPUT:

```
<terminated> Program29 (1) [Java Application] C:\Program Files\Java\
Drawing on canvas
Printing from printer
```

30) Write a program to demonstrate interface implementation by creating objects of the canvas and printer classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
    void draw();
}
interface Resizable {
    void resize();
}

class Program1 implements Drawable, Resizable {
    public void draw() {
        System.out.println("Drawing shape");
    }
    public void resize() {
        System.out.println("Resizing shape");
    }
    public static void main(String[] args) {
        Program1 s = new Program1();
        s.draw();
        s.resize();
    }
}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java\
Drawing shape
Resizing shape
```

31) Create interfaces Runnable and Callable with methods run and call. Implement them in classes representing a thread and a task.

```
package Interfaces;
interface MyRunnable {
    void run();
}
```

```
interface MyCallable {
   void call();
}
class MyThread implements MyRunnable {
   public void run() {
      System.out.println("Thread running");
   }
class MyTask implements MyCallable {
   public void call() {
      System.out.println("Task executing");
   }}
public class Program31 {
   public static void main(String[] args) {
      MyThread t = new MyThread();
      MyTask task = new MyTask();
      t.run();
      task.call();
   }
}
```

OUTPUT:

```
<terminated> Program31 (1) [Java Application] C:\Program Files\Java
Thread running
Task executing
```

32) Write a program to demonstrate interface implementation by creating objects of the thread and task classes and invoking interface methods.

```
package Interfaces;
interface Drawable {
   void draw();
}
interface Resizable {
   void resize();
}
class Program1 implements Drawable, Resizable {
   public void draw() {
      System.out.println("Drawing shape");
   }
   public void resize() {
      System.out.println("Resizing shape");
   }
   public static void main(String[] args) {
      Program1 s = new Program1();
      s.draw();
      s.resize();
   }}
```

OUTPUT:

```
<terminated> Program1 (9) [Java Application] C:\Program Files\Java
Drawing shape
Resizing shape
```

Collection

1) **Write a program to demonstrate ArrayList by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class ArrayListDemo {
   public static void main(String[] args) {
      ArrayList<String> list = new ArrayList<>();
      list.add("A");
      list.add("B");
      list.add("C");
      list.remove("B");
      for(String s : list) {
         System.out.println(s);
      }
   }
}
```

OUTPUT:

```
<terminated> Program1 (10) [Java Application] C:\Program Files\Java\jd
A
C
```

2) **Implement a program to demonstrate LinkedList by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class Program1 {
   public static void main(String[] args) {
      LinkedList<Integer> list = new LinkedList<>();
      list.add(10);
      list.add(20);
      list.add(30);
      list.remove(Integer.valueOf(20));
      for(int i : list) {
         System.out.println(i);
      }
   }
}
```

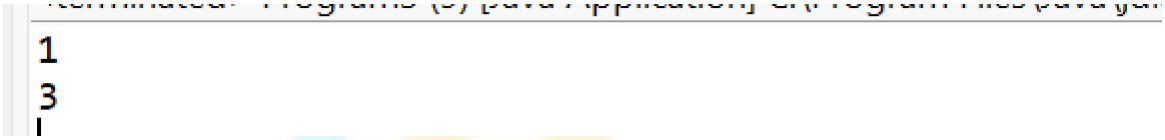
OUTPUT:

```
10
30
```

3) **Write a program to demonstrate HashSet by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class Program3 {
   public static void main(String[] args) {
      HashSet<Integer> set = new HashSet<>();
      set.add(1);
      set.add(2);
      set.add(3);
      set.remove(2);
      for(int i : set) {
         System.out.println(i);
      }
   }
}
```

OUTPUT:



4) Implement a program to demonstrate TreeSet by adding, removing, and iterating over elements.

```
package Collections;
import java.util.*;
class Program4 {
   public static void main(String[] args) {
      TreeSet<Integer> set = new TreeSet<>();
      set.add(30);
      set.add(10);
      set.add(20);
      set.remove(10);
      for(int i : set) {
         System.out.println(i);
      }
   }
}
```

OUTPUT:

5) Write a program to demonstrate HashMap by adding and retrieving key value pairs.

```
package Collections;
import java.util.*;
class Program5 {
   public static void main(String[] args) {
```

```
        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "Java");
        map.put(2, "Python");
        System.out.println(map.get(1));
        System.out.println(map.get(2));
    }
}
```

OUTPUT:

```
Java
Python
```

6) **Implement a program to demonstrate TreeMap by adding and retrieving key value pairs.**

```
package Collections;
import java.util.*;
class Program6 {
    public static void main(String[] args) {
        TreeMap<Integer, String> map = new TreeMap<>();
        map.put(3, "C");
        map.put(1, "Java");
        map.put(2, "Python");
        for(Map.Entry<Integer,String> e : map.entrySet()) {
            System.out.println(e.getKey() + " " + e.getValue());
        }
    }
}
```

OUTPUT:

```
<terminated> Program6 (6) [Java Application] C:\Program Files\Java\jd
1 Java
2 Python
3 C
```

7) **Write a program to demonstrate LinkedHashMap by adding and retrieving key value pairs.**

```
package Collections;
import java.util.*;
class Program7 {
    public static void main(String[] args) {
        LinkedHashMap<Integer, String> map = new LinkedHashMap<>();
        map.put(1, "One");
        map.put(2, "Two");
        for(Map.Entry<Integer,String> e : map.entrySet()) {
            System.out.println(e.getKey() + " " + e.getValue());
        }
    }
}
```

OUTPUT:

```
<terminated> Program7 (8) [Java Application] C:\Program Files\Java\jdk-25\b
1 One
2 Two
```

8) **Implement a program to demonstrate Queue by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class Program8 {
   public static void main(String[] args) {
      Queue<Integer> q = new LinkedList<>();
      q.add(10);
      q.add(20);
      q.remove();
      for(int i : q) {
         System.out.println(i);
      }
   }}
```

OUTPUT:

```
<terminated> Program8 (7) [Java Application] C:\Program Files\Java\jd
20
```

9) **Write a program to demonstrate PriorityQueue by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class Program9 {
   public static void main(String[] args) {
      PriorityQueue<Integer> pq = new PriorityQueue<>();
      pq.add(30);
      pq.add(10);
      pq.add(20);
      pq.poll();
      for(int i : pq) {
         System.out.println(i);
      }   }}
```

OUTPUT:

```
20
30
```

10) **Implement a program to demonstrate Stack by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class Program10 {
   public static void main(String[] args) {
```

```
        Stack<Integer> stack = new Stack<>();
        stack.push(1);
        stack.push(2);
        stack.pop();
        for(int i : stack) {
            System.out.println(i);
        }
    }
}
```

OUTPUT:

```
<terminated> Program10 (7) [Java Application] C:\Program Files\Java\jdk-2
1
```

11) **Write a program to demonstrate ArrayDeque by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
class Program11 {
    public static void main(String[] args) {
        ArrayDeque<Integer> dq = new ArrayDeque<>();
        dq.add(10);
        dq.add(20);
        dq.remove();
        for(int i : dq) {
            System.out.println(i);
        }
    }}
```

OUTPUT:

```
<terminated> Program11 (8) [Java Applicati
20
```

12) **Implement a program to demonstrate EnumSet by adding, removing, and iterating over elements.**

```
package Collections;
import java.util.*;
enum Day { MON, TUE, WED }
class Program12 {
    public static void main(String[] args) {
        EnumSet<Day> set = EnumSet.allOf(Day.class);
        set.remove(Day.TUE);
        for(Day d : set) {
            System.out.println(d);
        }
    }}
```

OUTPUT:

```
<terminated> Program12 (7) [Java Application] C:\Program Fil
MON
WED
```

13) Write a program to demonstrate BitSet by adding, removing, and iterating over elements.

```
package Collections;
import java.util.*;
class Program13 {
    public static void main(String[] args) {
        BitSet bs = new BitSet();
        bs.set(1);
        bs.set(3);
        bs.clear(1);
        System.out.println(bs);
    }}
```

OUTPUT:

```
{3}
```

14) Implement a program to demonstrate Hashtable by adding and retrieving key value pairs.

```
package Collections;
import java.util.*;
class Program14 {
    public static void main(String[] args) {
        Hashtable<Integer, String> ht = new Hashtable<>();
        ht.put(1, "One");
        ht.put(2, "Two");
        System.out.println(ht.get(1));
    }}
```

OUTPUT:

```
One
```

15) Write a program to demonstrate Properties by adding and retrieving key value pairs.

```
package Collections;
import java.util.*;
class Program15 {
    public static void main(String[] args) {
        Properties p = new Properties();
        p.setProperty("username", "admin");
        p.setProperty("password", "123");
        System.out.println(p.getProperty("username"));
    }}
```

OUTPUT:

```
<terminated> Program15 (7) [Java Application] C:\Progra
admin
```

16) Implement a program to demonstrate Vector by adding, removing, and iterating over elements.

```
package Collections;
import java.util.*;
class Program16 {
   public static void main(String[] args) {
      Vector<String> v = new Vector<>();
      v.add("A");
      v.add("B");
      v.remove("A");
      for(String s : v) {
         System.out.println(s);
      }
   }
}
```

OUTPUT:

```
<terminated> Program16 (5) [Java Application] C:\P
B
```

17) Write a program to demonstrate Enumeration by iterating over elements of a collection.

```
package Collections;
import java.util.*;
class Program17 {
   public static void main(String[] args) {
      Vector<Integer> v = new Vector<>();
      v.add(1);
      v.add(2);
      Enumeration<Integer> e = v.elements();
      while(e.hasMoreElements()) {
         System.out.println(e.nextElement());
      }
   }
}
```

OUTPUT:

```
<terminated> Program17 (8) [Java Application] C:\Pr
1
2
```

18) Implement a program to demonstrate ListIterator by iterating over elements of a list.

```
package Collections;
import java.util.*;
class Program18 {
   public static void main(String[] args) {
      ArrayList<String> list = new ArrayList<>();
```

```
        list.add("X");
        list.add("Y");
        ListIterator<String> it = list.listIterator();
        while(it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

OUTPUT:

```
<terminated> Program18 (5) [Java Application] C:\Pro
X
Y
```

19) Write a program to demonstrate Iterator by iterating over elements of a collection.

```
package Collections;
import java.util.*;
class Program19 {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        Iterator<Integer> it = list.iterator();
        while(it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

OUTPUT:

```
<terminated> Program19 (8) [Java Application] C:\Program Files\
10
20
```

20) Implement a program to demonstrate ArrayBlockingQueue by adding, removing, and iterating over elements.

```
package Collections;
import java.util.concurrent.*;
class Program20 {
    public static void main(String[] args) {
        ArrayBlockingQueue<Integer> q = new ArrayBlockingQueue<>(3);
        q.add(1);
        q.add(2);
        q.remove();
        for(int i : q) {
            System.out.println(i);
        }
    }
}
```

OUTPUT:

```
2
```

21) Write a program to demonstrate LinkedBlockingQueue by adding, removing, and iterating over elements.

```
package Collections;
import java.util.concurrent.*;
class Program21 {
   public static void main(String[] args) {
      LinkedBlockingQueue<Integer> q = new LinkedBlockingQueue<>();
      q.add(10);
      q.add(20);
      q.remove();
      for(int i : q) {
         System.out.println(i);
      }
   }
}
```

OUTPUT:

```
<terminated> Program21 (7) [Java Application] C:\Program
20
```

22) Implement a program to demonstrate PriorityBlockingQueue by adding, removing, and iterating over elements.

```
package Collections;
import java.util.concurrent.*;
class Program22 {
   public static void main(String[] args) {
      PriorityBlockingQueue<Integer> q = new PriorityBlockingQueue<>();
      q.add(30);
      q.add(10);
      q.add(20);
      q.poll();
      for(int i : q) {
         System.out.println(i);
      }
   }
}
```

OUTPUT:

```
<terminated> Program22 (5) [Java Application] C:\Program Files\Java\jdk-25
20
30
```

23) Write a program to demonstrate SynchronousQueue by adding and removing elements.

```
package Collections;
import java.util.concurrent.*;
class Program23 {
   public static void main(String[] args) throws Exception {
      SynchronousQueue<Integer> q = new SynchronousQueue<>();
      new Thread(() -> {
```

```
            try {
                q.put(10);
            } catch (Exception e) {}
        }).start();
        System.out.println(q.take());
    }
}
```

OUTPUT:

```
<terminated> Program23 (7) [Java Application] C:\P
10
```

24) Implement a program to demonstrate DelayQueue by adding and retrieving elements.

```
package Collections;
import java.util.concurrent.*;
import java.util.*;
class MyDelay implements Delayed {
    long time;
    MyDelay(long delay) {
        time = System.currentTimeMillis() + delay;
    }
    public long getDelay(TimeUnit unit) {
        return unit.convert(time - System.currentTimeMillis(),
TimeUnit.MILLISECONDS);
    }
    public int compareTo(Delayed d) {
        return Long.compare(this.time, ((MyDelay)d).time);
    }
}
class Program24 {
    public static void main(String[] args) throws Exception {
        DelayQueue<MyDelay> dq = new DelayQueue<>();
        dq.add(new MyDelay(1000));
        System.out.println("Element retrieved: " + dq.take());
    }
}
```

OUTPUT:

```
<terminated> Program24 (4) [Java Application] C:\Program Files\Java\jdk
Element retrieved: Collections.MyDelay@c387f44
```

25) Write a program to demonstrate ConcurrentLinkedQueue by adding, removing, and iterating over elements.

```
package Collections;
import java.util.concurrent.*;
class Program25 {
    public static void main(String[] args) {
        ConcurrentLinkedQueue<Integer> q = new ConcurrentLinkedQueue<>();
        q.add(1);
        q.add(2);
        q.remove();
```

```
        for(int i : q) {
            System.out.println(i);
        }
    }}
```

OUTPUT:

```
<terminated> Program25 (5) [Java Application] C:\Program File
2
```

Multi Threading

1) **Write a Java program to create multiple threads and display their names.**

```
package Multithreding;
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread name: " + Thread.currentThread().getName());
    }
}
public class Program1 {
    public static void main(String[] args) {
        new MyThread().start();
        new MyThread().start();
        new MyThread().start();
    }
}
```

OUTPUT:

```
<terminated> Program1 (11) [Java Application] C:\Progra
Thread name: Thread-2
Thread name: Thread-1
Thread name: Thread-0
```

2) **Implement a program to demonstrate thread synchronization using synchronized blocks.**

```
package Multithreding;
class Counter {
    int count = 0;
    void increment() {
        synchronized (this) {
            count++;
        }
    }
}
public class Program2 {
    public static void main(String[] args) throws Exception {
        Counter c = new Counter();
        Thread t1 = new Thread(() -> { for(int i=0;i<1000;i++) c.increment(); });
        Thread t2 = new Thread(() -> { for(int i=0;i<1000;i++) c.increment(); });
        t1.start(); t2.start();
        t1.join();                          t2.join();
        System.out.println("Count: " + c.count);
    }
}
```

OUTPUT:

```
<terminated> Program2 (9) [Java Applicatic
Count: 2000
```

3) **Write a Java program to create multiple threads and display their priorities.**

```
package Multithreding;
class Program3 extends Thread {
   public void run() {
      System.out.println(getName() + " Priority: " + getPriority());
   }
   public static void main(String[] args) {
      Program3 t1 = new Program3();
      Program3 t2 = new Program3();
      t1.setPriority(1);
      t2.setPriority(10);
      t1.start();
      t2.start();
   }
}
```

OUTPUT:

```
Thread-0 Priority: 1
Thread-1 Priority: 10
```

4) **Implement a program to create a thread pool and execute multiple tasks using Executor Service.**

```
package Multithreding;
import java.util.concurrent.*;
public class Program4 {
   public static void main(String[] args) {
      ExecutorService es = Executors.newFixedThreadPool(3);
      for(int i=1;i<=5;i++) {
         es.execute(() -> System.out.println("Task by " +
Thread.currentThread().getName()));
      }
      es.shutdown();
   }
}
```

OUTPUT:

```
<terminated> Program4 (10) [Java Application] C:\Program File
Task by pool-1-thread-2
Task by pool-1-thread-3
Task by pool-1-thread-2
Task by pool-1-thread-3
Task by pool-1-thread-1
```

5) **Write a Java program to create multiple threads and join them.**

```
package Multithreding;
class Program5 extends Thread {
   public void run() {
      System.out.println(getName() + " running");
   }
   public static void main(String[] args) throws Exception {
      Program5 t1 = new Program5();
```

```
        Program5 t2 = new Program5();
        t1.start();
        t1.join();
        t2.start();
    }}
```

OUTPUT:

```
<terminated> Program5 (7) [Java Application] C:\Program Files\
Thread-0 running
Thread-1 running
```

6) **Implement a program to demonstrate deadlock condition in multithreading.**

```
package Multithreding;
class Program6 {
    static final Object A = new Object();
    static final Object B = new Object();
    public static void main(String[] args) {
        Thread t1 = new Thread(() -> {
            synchronized(A) {
                synchronized(B) {}
            }
        });
        Thread t2 = new Thread(() -> {
            synchronized(B) {
                synchronized(A) {}
            }
        });
        t1.start();
        t2.start();
    }
}
```

OUTPUT:

```
<terminated> Program6 (7) [Java Application] C:\Program Files\Java\jdk-25\bin\
```

7) **Write a Java program to create multiple threads and interrupt them.**

```
package Multithreding;
class Program7 extends Thread {
    public void run() {
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }
    public static void main(String[] args) {
        Program7 t = new Program7();
        t.start();
        t.interrupt();
    }
}
```

OUTPUT:

```
Thread interrupted
```

8) Implement a program to demonstrate thread local variables.

```
package Multithreding;
class Program8 {
   static ThreadLocal<Integer> tl = ThreadLocal.withInitial(() -> 0);
   public static void main(String[] args) {
      Runnable r = () -> {
         tl.set((int)(Math.random()*100));
         System.out.println(Thread.currentThread().getName() + " value: " + tl.get());
      };
      new Thread(r).start();
      new Thread(r).start();
   }
}
```

OUTPUT:

```
<terminated> Program8 (8) [Java Application] C:\Program F
Thread-0 value: 58
Thread-1 value: 86
```

9) Write a Java program to create multiple threads and wait for them to complete using Count Down Latch.

```
package Multithreding;
import java.util.concurrent.*;
public class Program9 {
   public static void main(String[] args) throws Exception {
      CountDownLatch latch = new CountDownLatch(3);
      for(int i=1;i<=3;i++) {
         new Thread(() -> {
            System.out.println("Task done");
            latch.countDown();
         }).start();
      }
      latch.await();
      System.out.println("All threads completed");
   }
}
```

OUTPUT:

```
<terminated> Program9 (8) [Java Application] C:\Program Files\Java\jdk-2
Task done
Task done
Task done
All threads completed
```

10) Implement a program to demonstrate thread priorities in Java.

```
package Multithreding;
public class Program10 {
   public static void main(String[] args) {
      Thread t = new Thread();
      t.setPriority(Thread.MAX_PRIORITY);
      System.out.println(t.getPriority());
   }
}
```

OUTPUT:

```
<terminated> Program10 (8) [Java Application] C
10
```

11) Write a Java program to create multiple threads and use thread group.

```
package Multithreding;
public class Program11 {
    public static void main(String[] args) {
        ThreadGroup g = new ThreadGroup("MyGroup");
        new Thread(g, () -> System.out.println("Thread in group")).start();
        System.out.println("Group name: " + g.getName());
    }
}
```

OUTPUT:

```
<terminated> Program11 (9) [Java Application] C:\Program F
Thread in group
Group name: MyGroup
```

12) Implement a program to demonstrate thread communication using wait and notify methods.

```
package Multithreding;
class Shared {
    synchronized void waitMethod() throws Exception {
        wait();
        System.out.println("Resumed");
    }
    synchronized void notifyMethod() {
        notify();
    }
}
public class Program12 {
    public static void main(String[] args) {
        Shared s = new Shared();

        new Thread(() -> {
            try { s.waitMethod(); } catch(Exception e){}
        }).start();

        new Thread(() -> {
            try { Thread.sleep(1000); s.notifyMethod(); } catch(Exception e){}
        }).start();
    }}
```

OUTPUT:

```
<terminated> Program12 (8) [Java Application] C:\Program
Resumed
```

13) Write a Java program to create multiple threads and use thread local variables.

```
package Multithreding;
class Program13 {
    static ThreadLocal<String> tl = new ThreadLocal<>();
    public static void main(String[] args) {
        new Thread(() -> {
```

```
            tl.set("A");
            System.out.println(tl.get());
        }).start();
    }}
```

OUTPUT:

```
<terminated> Program13 (9) [Java Application] C:\Program Files\Java\jdk-2
A
```

14) Implement a program to demonstrate thread communication using volatile keyword.

```
package Multithreding;
class Program14 {
    volatile boolean flag = true;
    public static void main(String[] args) throws Exception {
        Program14 v = new Program14();
        new Thread(() -> {
            while(v.flag){}
            System.out.println("Stopped");
        }).start();
        Thread.sleep(1000);
        v.flag = false;
    }
}
```

OUTPUT:

```
<terminated> Program14 (7) [Java Application] C:\Prog
Stopped
```

15) Write a Java program to create multiple threads and use Executors framework.

```
package Multithreding;
import java.util.concurrent.*;
public class Program15 {
    public static void main(String[] args) {
        ExecutorService es = Executors.newSingleThreadExecutor();
        es.submit(() -> System.out.println("Executor task"));
        es.shutdown();
    }
}
```

OUTPUT:

```
<terminated> Program15 (8) [Java Application] C:\Pro
Executor task
```

16) Implement a program to demonstrate thread interruption in Java.

```
package Multithreding;
public class Program16 {
    public static void main(String[] args) {
        Thread t = new Thread(() -> {
            while(!Thread.currentThread().isInterrupted()) {}
            System.out.println("Interrupted");
        });
        t.start();
        t.interrupt();
```

```
    }
}
```

OUTPUT:

```
<terminated> Program16 (6) [Java Application] C:\Program F
Interrupted
```

17) Write a Java program to create multiple threads and use Callable and Future.

```
package Multithreding;
import java.util.concurrent.*;
public class Program17 {
    public static void main(String[] args) throws Exception {
        ExecutorService es = Executors.newSingleThreadExecutor();
        Future<Integer> f = es.submit(() -> 10 + 20);
        System.out.println(f.get());
        es.shutdown();
    }
}
```

OUTPUT:

```
30
```

18) Implement a program to demonstrate thread communication using BlockingQueue.

```
package Multithreding;
import java.util.concurrent.*;
public class Program18 {
    public static void main(String[] args) throws Exception {
        BlockingQueue<Integer> q = new ArrayBlockingQueue<>(1);
        new Thread(() -> {
            try { q.put(10); } catch(Exception e){}
        }).start();
        System.out.println(q.take());
    }
}
```

OUTPUT:

```
<terminated> Program18 (6) [Java Application] C:\Program Files\Java\jdk-
10
```

19) Write a Java program to create multiple threads and use Phaser.

```
package Multithreding;
import java.util.concurrent.*;
public class Program19 {
    public static void main(String[] args) {
        Phaser p = new Phaser(2);
        new Thread(() -> {
            System.out.println("Thread 1");
            p.arrive();
        }).start();
        new Thread(() -> {
```

```
            System.out.println("Thread 2");
            p.arrive();
        }).start();
    }
}
```

OUTPUT:

```
Thread 1
Thread 2
```

20) Implement a program to demonstrate thread communication using CyclicBarrier.

```
package Multithreding;
import java.util.concurrent.*;
public class Program20 {
    public static void main(String[] args) {
        CyclicBarrier cb = new CyclicBarrier(2, () -> System.out.println("Barrier
reached"));
        new Thread(() -> { try { cb.await(); } catch(Exception e){} }).start();
        new Thread(() -> { try { cb.await(); } catch(Exception e){} }).start();
    }
}
```

OUTPUT:

```
<terminated> Program20 (6) [Java Application] C:\
Barrier reached
```

21) Write a Java program to create multiple threads and use Semaphore.

```
package Multithreding;
import java.util.concurrent.*;
public class Program21 {
    public static void main(String[] args) {
        Semaphore s = new Semaphore(1);
        new Thread(() -> {
            try {
                s.acquire();
                System.out.println("Accessed");
                s.release();
            } catch(Exception e){}
        }).start();
    }
}
```

OUTPUT:

```
<terminated> Program21 (8) [Java Application] C:\Program Files\Ja
Accessed
```

22) Implement a program to demonstrate thread communication using Exchanger.

```
package Multithreding;
import java.util.concurrent.*;
public class Program22 {
    public static void main(String[] args) {
        Exchanger<String> ex = new Exchanger<>();
        new Thread(() -> {
```

```
            try {
                System.out.println(ex.exchange("Hello"));
            } catch(Exception e){}
        }).start();
        new Thread(() -> {
            try {
                System.out.println(ex.exchange("World"));
            } catch(Exception e){}
        }).start();
    }
}
```

OUTPUT:

```
Hello
World
```

23) Write a Java program to create multiple threads and use CompletionService.

```
package Multithreding;
import java.util.concurrent.*;
public class Program23 {
    public static void main(String[] args) throws Exception {
        ExecutorService es = Executors.newFixedThreadPool(2);
        CompletionService<Integer> cs = new ExecutorCompletionService<>(es);
        cs.submit(() -> 5);
        cs.submit(() -> 10);
        System.out.println(cs.take().get());
        System.out.println(cs.take().get());
        es.shutdown();
    }
}
```

OUTPUT:

```
5
10
```

24) Implement a program to demonstrate thread communication using TransferQueue.

```
package Multithreding;
import java.util.concurrent.*;
public class Program24 {
    public static void main(String[] args) throws Exception {
        TransferQueue<Integer> tq = new LinkedTransferQueue<>();
        new Thread(() -> {
            try { tq.transfer(100); } catch(Exception e){}
        }).start();
        System.out.println(tq.take());
    }
}
```

OUTPUT:

```
100
```

25) Write a Java program to create multiple threads and use ScheduledExecutorService.

```
package Multithreding;
import java.util.concurrent.*;
public class Program25 {
   public static void main(String[] args) {
      ScheduledExecutorService ses = Executors.newScheduledThreadPool(1);

      ses.schedule(() -> System.out.println("Scheduled task"), 2,
TimeUnit.SECONDS);
      ses.shutdown();
   }
}
```

OUTPUT:

```
<terminated> Program25 (6) [Java Application] C:\Program Fil
Scheduled task
```

26) Implement a program to demonstrate thread communication using Lock and Condition.

```
package Multithreding;

import java.util.concurrent.locks.*;
public class Program26 {
   static Lock lock = new ReentrantLock();
   static Condition cond = lock.newCondition();
   public static void main(String[] args) {
      new Thread(() -> {
         lock.lock();
         try {
            cond.await();
            System.out.println("Resumed");
         } catch(Exception e){}
         finally { lock.unlock(); }
      }).start();

      new Thread(() -> {
         lock.lock();
         try {
            cond.signal();
         } finally { lock.unlock(); }
      }).start();
   }
}
```

OUTPUT:

```
Resumed
```