

## **CORE JAVA PROGRAMS:**

## Introduction :

## **1)Write a Java program to print your name**

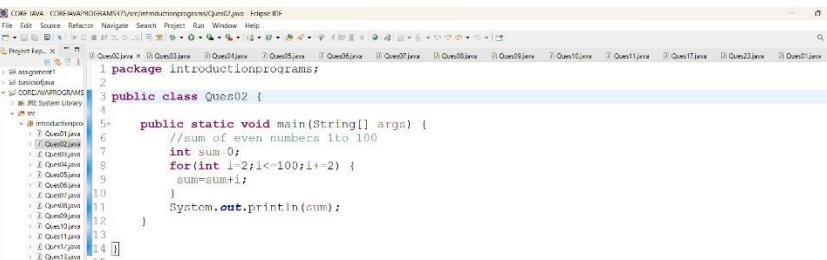
The screenshot shows the Oracle Java Development Kit 11 interface. The code editor displays a Java program named `introductionprograms.java` with the following content:

```
package introductionprograms;
import java.util.Scanner;
public class GreetMe {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("ENTER YOUR NAME");
        String name = scan.nextLine();
        System.out.println("MY NAME IS:" + name);
        scan.close();
    }
}
```

The terminal window below the code editor shows the output of running the program:

```
ENTER YOUR NAME
MY NAME IS:JAYASURYA
```

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

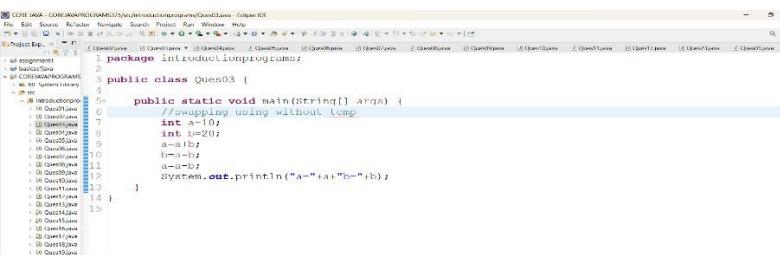


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** COR-JAVA-CORNSA-PROGRAMMING/introductionprograms/Ques02 - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar with icons for New, Open, Save, Cut, Copy, Paste, Find, etc.
- Project Explorer:** Shows a Java project named "introductionprograms". Inside, there is a package named "introductionprograms" containing a class "Ques02.java". Other files listed include "Ques01.java", "Ques03.java", "Ques04.java", "Ques05.java", "Ques06.java", "Ques07.java", "Ques08.java", "Ques09.java", "Ques10.java", "Ques11.java", "Ques12.java", "Ques13.java", "Ques14.java", "Ques15.java", "Ques16.java", "Ques17.java", "Ques18.java", "Ques19.java", "Ques20.java", "Ques21.java", "Ques22.java", and "module-info.java".
- Code Editor:** The "Ques02.java" file is open, displaying the following Java code:

```
1 package introductionprograms;
2
3 public class Ques02 {
4
5     public static void main(String[] args) {
6         //sum of even numbers lto 100
7         int sum = 0;
8         for(int i=2;i<100;i+=2) {
9             sum=sum+i;
10        }
11        System.out.println(sum);
12    }
13}
14
15
```
- Console View:** Shows the message "terminated: Ques02 [Java Application]".
- Bottom Status Bar:** Displays the date and time as "Feb 22, 2025, 12:58:42 AM" and the process ID as "22994".

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



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a package named "introductionprograms" containing a class "Quesn03".
- Code Editor:** Displays the Java code for "Quesn03".

```
public class Quesn03 {
    public static void main(String[] args) {
        System.out.println("Program doing without Comp");
        int a=10;
        int b=20;
        a=a+b;
        b=b-a;
        a=a-b;
        System.out.println("a=" + a + "b=" + b);
    }
}
```
- Outline View:** Shows the class structure with methods "main" and "System.out.println".
- Search View:** Shows search results for "Quesn03".
- Task List:** Shows tasks related to "Quesn03".
- Properties View:** Shows file properties for "Quesn03.java".
- Console View:** Shows the output of the run command: "a=30b=-10".
- Help View:** Shows help information for "Quesn03".

#### 4) Write a program to check whether a given number is prime or not.

The screenshot shows the Eclipse IDE interface with a Java project named "COREJAVA". The code editor displays a file named "Ques04.java" containing the following code:

```
package introductionprograms;
public class Ques04 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("ENTER ANY NUMBER");
        int num = scan.nextInt();
        int count=0;
        for(int i=1;i<=num;i++) {
            if(num%i==0) {
                count++;
            }
        }
        if(count==2) {
            System.out.println(num+" IT IS A PRIME NUMBER");
        } else {
            System.out.println(num+" IT IS NOT A PRIME number");
        }
    }
}
```

The Java console output shows the program's execution:

```
ENTER ANY NUMBER
6
IT IS NOT A PRIME number
```

#### 5) Write a program to calculate the factorial of a given number using recursion

The screenshot shows the Eclipse IDE interface with a Java project named "COREJAVA". The code editor displays a file named "Ques05.java" containing the following code:

```
package introductionprograms;
import java.util.Scanner;
public class Ques05 {
    //factorial of the given number
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("ENTER ANY NUMBER");
        int num=scan.nextInt();
        int fact=1;
        for(int i=1;i<=num;i++) {
            fact=fact*i;
        }
        System.out.println(fact);
        scan.close();
    }
}
```

The Java console output shows the program's execution:

```
ENTER ANY NUMBER
120
120
```

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

The screenshot shows the Eclipse IDE interface with a Java project named "COREJAVA". The code editor displays a file named "Ques06.java" containing the following code:

```
package introductionprograms;
import java.util.Scanner;
public class Ques06 {
    // roots of quadratic equation
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter A value");
        int a = scan.nextInt();
        System.out.print("Enter B value");
        int b = scan.nextInt();
        System.out.print("Enter C value");
        int c = scan.nextInt();
        int rcc;
        res=b*b-(4*a*c)/(2*a);
        System.out.println(res);
        if(res>0) {
            System.out.println("Tl as roots");
        } else {
            System.out.println("It as no roots");
        }
    }
}
```

The Java console output shows the program's execution:

```
Enter A value
5
Enter B value
6
Enter C value
7
Tl as roots
```

```

1 package introductionprograms;
2
3 import java.util.Scanner;
4
5 public class Ques07 {
6
7     public static void main(String[] args) {
8         //area of triangle
9         Scanner scan = new Scanner(System.in);
10        System.out.println("Enter base");
11        int base = scan.nextInt();
12        System.out.println("Enter height");
13        int height = scan.nextInt();
14        float res;
15        res = 0.5f*base*height;
16        System.out.println(res);
17        scan.close();
18    }
19}

```

The screenshot shows the Eclipse IDE interface with the project structure and code editor. The code calculates the area of a triangle using user input for base and height.

## 8) Write a program to print the Fibonacci series up to a given number of terms.

```

1 package introductionprograms;
2
3 public class Ques08 {
4
5     public static void main(String[] args) {
6         // fibonacci series
7         int n=10;
8         int a=0;
9         int b=1;
10        for(int i=1;i<n;i++) {
11            System.out.print(a+" ");
12            int nextTerm=a+b;
13            a=b;
14            b=nextTerm;
15        }
16    }
17}

```

The screenshot shows the Eclipse IDE interface with the project structure and code editor. The code prints the first 10 terms of the Fibonacci series.

## 9. Write a program to find the second largest number in an array.

```

1 package introductionprograms;
2
3 public class Ques09 {
4
5     public static void main(String[] args) {
6         // second largest num in array
7         int[] num = {1,3,0,4,9};
8         System.out.println("the second largest number is");
9         int size=num.length;
10        Arrays.sort(num);
11        System.out.println(num[size-2]);
12    }
13}

```

The screenshot shows the Eclipse IDE interface with the project structure and code editor. The code finds the second largest number in an array of integers.

## 10) Write a program to reverse a string

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows several Java files: assignment1, basiccalculator, corejavaprograms75, introductionprograms, Ques01.java, Ques02.java, Ques03.java, Ques04.java, Ques05.java, Ques06.java, Ques07.java, Ques08.java, Ques09.java, Ques10.java, Ques11.java, Ques12.java, Ques13.java, Ques14.java, Ques15.java, Ques16.java, Ques17.java, Ques18.java, Ques19.java, Ques20.java, Ques21.java, Ques22.java, Ques23.java, Ques24.java, and module-info.java.
- Code Editor:** Displays the Java code for `Ques10.java`. The code defines a class `Ques10` with a static main method that takes a string argument and prints its reverse. A cursor is positioned at the end of the line `System.out.println(s2);`.
- Console:** Shows the output of the Java application, indicating it has started and is ready for input.
- Bottom Bar:** Includes standard operating system icons for file operations, search, and system status.

```
package introductionprograms;
public class Ques10 {
    public static void main(String[] args) {
        //reverse of string
        String s1="surya";
        StringBuilder str = new StringBuilder(s1);
        str.reverse();
        String s2=str.toString();
        System.out.println(s2);
    }
}
```

**11) Write a program to check if a given string is a palindrome.**

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - CORE INQUIRIES&ANSWERS/scr/inquiriesandprograms/Ques1.java - Eclipse IDE
- Project Explorer:** Shows a project named "Ques1" containing files like Ques01.java through Ques25.java.
- Code Editor:** Displays the Java code for "Ques12.java". The code reads a string from the user and counts the occurrences of a specified character ('a' in this case). It includes imports for Scanner and java.util.Scanner, and uses System.out.println statements for output.
- Terminal:** Shows the command "java -jar Ques1.jar" being run and the output "ENTER ANY STRING" followed by "THE OCCURANCE OF A 2".
- System Tray:** Shows icons for battery, signal strength, and system date/time.

**12) Write a program to count the occurrences of a character in a string**

The screenshot shows the Eclipse IDE interface with the following details:

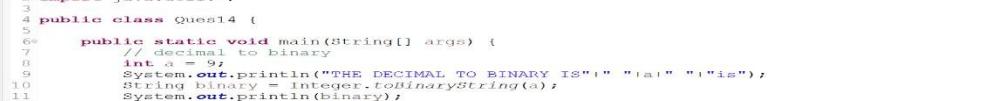
- Title Bar:** CORE JAVA - COREJAVAPROGRAM275 - Search | Recent programs | Documentation | Eclipse IDE
- Project Explorer:** Shows a project named "COREJAVAPROGRAM275" containing several Java files (Ques01.java through Ques20.java) and a build.properties file.
- Code Editor:** Displays the content of Ques11.java. The code defines a class "Ques11" with a static void main method. It takes a string argument, initializes it to "surya", creates a StringBuilder object, reverses the string, and then compares it with the original string. If they are equal, it prints "IT IS A PALINDROME"; otherwise, it prints "IT IS NOT A PALINDROME".
- Console:** Shows the output of the program: "IT IS NOT A PALINDROME".
- Bottom Status Bar:** Shows the date and time as "Date: 22/07/2025, 10:15 AM - 101159 AM" and the IP address "192.168.61.17.0.5.vOD221102-0413301@asuswrt".

13) Write a program to find the GCD (Greatest Common Divisor) of two numbers

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "COREJAVA". Inside, there are several source files: Ques01.java through Ques24.java, and a package named "introductionprograms" containing the file "Ques13.java".
- Code Editor:** The file "Ques13.java" is open. It contains Java code to find the greatest common divisor (GCD) of two numbers,  $a$  and  $b$ . The code uses a for loop to iterate from 1 to the smaller of  $a$  and  $b$ , checking if both numbers are divisible by the current value. If so, it prints the value and sets it as the GCD.
- Console:** The output window shows the execution of the program. It prints "THE GREATEST COMMON FACTOR IS" followed by the value "10".
- Bottom Status Bar:** Displays the date and time as "Date: 22/02/2024, Time: 10:05 AM (IST) [14:05]".

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



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - CORE JAVA/PROGRAMS/17/Introductionprograms/Ques14.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA" containing several Java files (Ques14.java, Ques01.java, Ques02.java, etc.) and a module-info.java file.
- Code Editor:** Displays the Java code for Ques14.java:

```
1 package introductionprograms;
2 import java.util.*;
3
4 public class Ques14 {
5
6     public static void main(String[] args) {
7         int decimal = 9;
8         System.out.println("THE DECIMAL TO BINARY IS:" + (int) 10);
9         String binary = Integer.toBinaryString(decimal);
10        System.out.println(binary);
11    }
12
13
14
15
16 }
17
18 }
```
- Console:** Shows the output of the program: THE DECIMAL TO BINARY IS 9 is 1001
- Bottom Bar:** Includes the Windows taskbar with various pinned icons like File Explorer, Edge, and Mail.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA\PROGRAMS375\src\introductionprograms\Ques15.java - Eclipse IDE
- Project Explorer:** Shows a project named "assignment1" containing several Java files (Ques01.java through Ques25.java) and a module-info.java file.
- Code Editor:** Displays the content of Ques15.java:

```
1 package introductionprograms;
2
3 import java.util.Scanner;
4
5 public class Ques15 {
6
7     public static void main(String[] args) {
8         //sum of digits of number
9         Scanner scan = new Scanner (System.in);
10        System.out.println("ENTER ANY NUMBER");
11        int a = scan.nextInt();
12        System.out.println("THE SUM OF DIGITS OF NUMBER IS");
13        if(a==0) {
14            System.out.println("0");
15        }else {
16            System.out.println(a%10+a/10);
17        }
18    }
19
20}
```
- Console:** Shows the output of the program:

```
ENTER ANY NUMBER
THE SUM OF DIGITS OF NUMBER IS
5
```
- Bottom Status Bar:** Includes system icons for battery, signal, and time (10:05 AM, 2/22/2025).

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS5375/src/introductionprograms' open. The code editor displays Java code for 'Ques16.java' which checks if a year is a leap year. The code uses a Scanner to read input from the user and prints the result. The terminal window shows the output for the year 2004.

```

1 package introductionprograms;
2
3 import java.util.Scanner;
4
5 public class Ques16 {
6     //leap year or not
7     public static void main(String[] args) {
8         Scanner scan = new Scanner(System.in);
9         System.out.println("ENTER ANY YEAR");
10        int year=scan.nextInt();
11        if(year%4==0||year%100==0||year%400==0) {
12            System.out.println(year+" "+"IT IS A LEAP YEAR");
13        }
14        else {
15            System.out.println(year+" "+"IT IS NOT A LEAP YEAR");
16        }
17    }
18 }

```

```

2004 IT IS A LEAP YEAR

```

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS5375/src/introductionprograms' open. The code editor displays Java code for 'Ques17.java' which calculates the factorial of a number using a for loop. The code reads a number from the user and prints the factorial. The terminal window shows the output for the number 5.

```

1 package introductionprograms;
2
3 public class Ques17 {
4     public static void main(String[] args) {
5         int a=1;
6         int fact=1;
7         for(int i=1;i<=5;i++) {
8             fact=fact*i;
9         }
10        System.out.println(fact);
11    }
12 }

```

```

120

```

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS5375/src/introductionprograms' open. The code editor displays Java code for 'Ques18.java' which prints a Pascal's triangle. The code uses nested loops to calculate and print the numbers in each row. The terminal window shows the output for 5 rows of the Pascal's triangle.

```

1 package introductionprograms;
2
3 public class Ques18 {
4     public static void main(String[] args) {
5         int rows = 5; // Number of rows in the Pascal's Triangle
6         // Outer loop for each row
7         for (int i = 0; i < rows; i++) {
8             int num = 1;
9             // Print leading spaces to center the triangle
10            for (int j = 0; j < rows - i - 1; j++) {
11                System.out.print(" ");
12            }
13            // Inner loop for printing numbers in each row
14            for (int k = 0; k < i + 1; k++) {
15                System.out.print(num+" ");
16                // Update number for next element in the row
17                num = num * (i - k) / (k + 1);
18            }
19            // Move to the next line after each row
20            System.out.println();
21        }
22    }
23 }

```

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4

```

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

```

1 package introductionprograms;
2 import java.util.Scanner;
3
4 public class Ques19 {
5     public static void main(String[] args) {
6         //number Armstrong number or not
7         Scanner scanner = new Scanner(System.in);
8         System.out.print("Enter a number: ");
9         int number = scanner.nextInt();
10        String s1= Integer.toString(number);
11        int length=s1.length();
12        int sum = 0;
13        temp = number;
14        while(temp>0) {
15            sum+=temp%10;
16            temp/=10;
17        }
18        if(sum==number) {
19            System.out.println(number + " is an Armstrong number.");
20        } else {
21        }
22    }
23}

```

Console output: Enter a number: 153  
153 is an Armstrong number.

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

```

1 package introductionprograms;
2
3 public class Ques20 {
4     public static void main(String[] args) {
5         // area and perimeter of triangle
6         int base=10;
7         int height=20;
8         int side_b;
9         float res1;
10        int res2;
11        res1=(0.5f)*base*height;
12        res2=side*side*side;
13        System.out.println("AREA OF TRIANGLE"+ " " +res1);
14        System.out.println("PERIMETER OF TRIANGLE"+ " " +res2);
15    }
16}

```

Console output: AREA OF TRIANGLE 100.0  
PERIMETER OF TRIANGLE 125

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

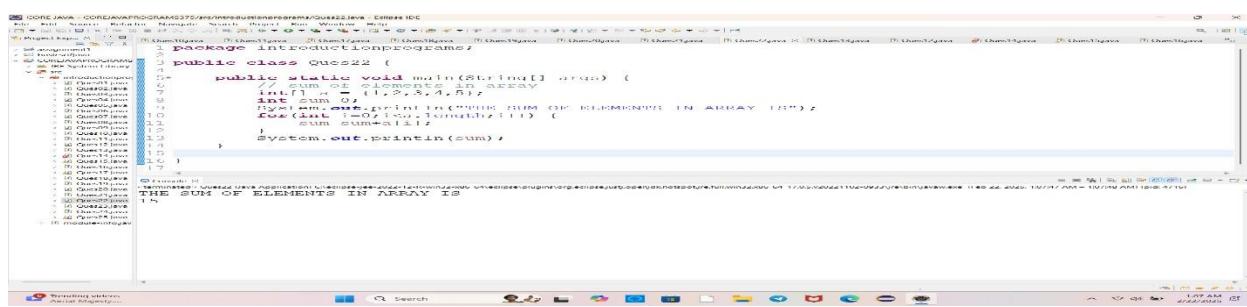
```

1 package introductionprograms;
2
3 import java.util.Scanner;
4
5 public class Ques21 {
6     public static void main(String[] args) {
7         // multiplication of table
8         Scanner scan = new Scanner(System.in);
9         System.out.println("Enter a number");
10        int num = scan.nextInt();
11        int i=1;
12        while(i<=10) {
13            System.out.println(num+"*"+i+"="+(num*i));
14            i++;
15        }
16        scan.close();
17    }
18}

```

Console output: Enter a number  
6+1=6  
6+2=12  
6+3=18  
6+4=24  
6+5=30  
6+6=36  
6+7=42  
6+8=48  
6+9=54  
6+10=60

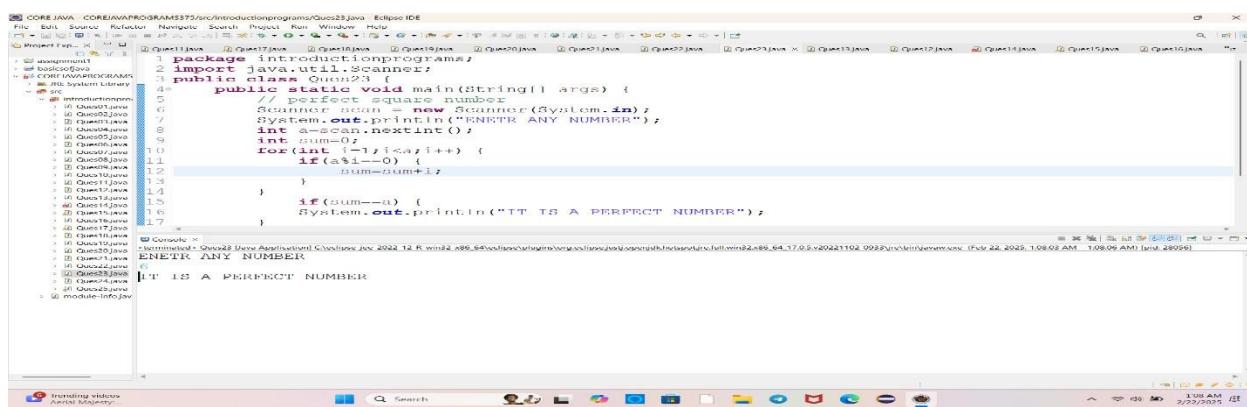
## 22 ) Write a program to find the sum of all elements in an array.



```
public class Quo22 {
    public static void main(String[] args) {
        int sum=0;
        for(int i=0;i<array.length;i++) {
            sum+=array[i];
        }
        System.out.println(sum);
    }
}
```

THE SUM OF ELEMENTS IN ARRAY IS  
15

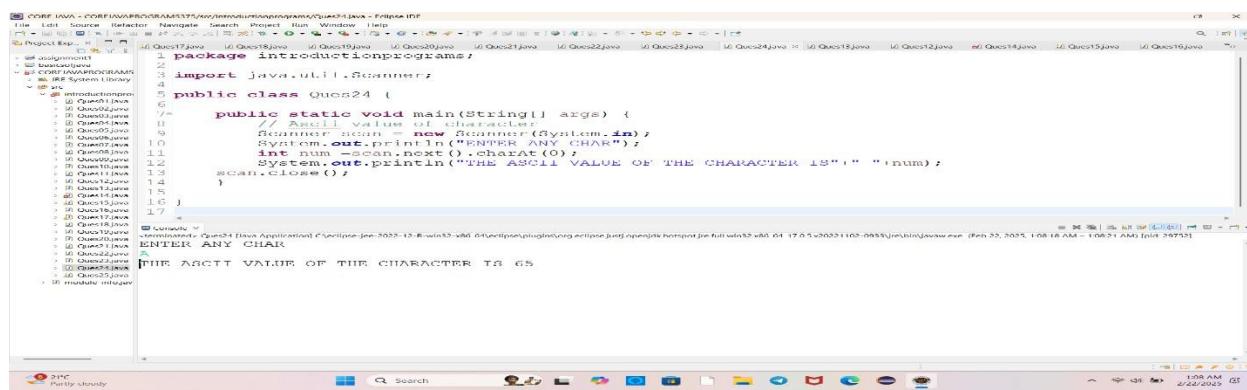
23) . Write a program to check if a given number is a perfect number.



```
public class Quo23 {
    public static void main(String[] args) {
        //perfect square numbers
        Scanner num = new Scanner(System.in);
        System.out.println("ENTER ANY NUMBER");
        int num=0;
        for(int i=1;i<=num;i++) {
            if(i*i==num) {
                num=num+1;
            }
        }
        if(num==i) {
            System.out.println("IT IS A PERFECT NUMBER");
        }
    }
}
```

ENTER ANY NUMBER  
IT IS A PERFECT NUMBER

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



```
public class Quo24 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("ENTER ANY CHAR");
        int num=scan.next().charAt(0);
        System.out.println("THE ASCII VALUE OF THE CHARACTER IS "+num);
        scan.close();
    }
}
```

ENTER ANY CHAR  
THE ASCII VALUE OF THE CHARACTER IS 65

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

```

1 package operatorsprograms;
2
3 import java.util.Scanner;
4
5 public class Ques25 {
6     public static void main(String[] args) {
7         Scanner scan = new Scanner(System.in);
8         System.out.println("ENTER 1ST NUMBER");
9         int num1 = scan.nextInt();
10        System.out.println("ENTER 2ND NUMBER");
11        int num2 = scan.nextInt();
12        System.out.println("ADDITION:" + (num1+num2));
13        System.out.println("SUBTRACTION:" + (num1-num2));
14        System.out.println("MULTIPLICATION:" + (num1*num2));
15        System.out.println("DIVISION:" + (num1/num2));
16        scan.close();
17    }
18}
19
```

## OPERATORS:

1) Write a program to perform arithmetic operations (+ /) on two numbers

```

1 package operatorsprograms;
2
3 import java.util.Scanner;
4
5 public class Ques2 {
6     public static void main(String[] args) {
7         // arithmetic operators
8         Scanner scan = new Scanner(System.in);
9         System.out.println("ENTER 1ST NUMBER");
10        int num1 = scan.nextInt();
11        System.out.println("ENTER 2ND NUMBER");
12        int num2 = scan.nextInt();
13        System.out.println("ADDITION:" + (num1+num2));
14        System.out.println("SUBTRACTION:" + (num1-num2));
15        System.out.println("MULTIPLICATION:" + (num1*num2));
16        System.out.println("DIVISION:" + (num1/num2));
17        scan.close();
18    }
19}
20
```

Console output:  
ENTER 1ST NUMBER  
20  
ENTER 2ND NUMBER  
40  
ADDITION:60  
SUBTRACTION:-20  
MULTIPLICATION:800  
DIVISION:0

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

```

1 package operatorsprograms;
2
3 import java.util.Scanner;
4
5 public class Ques2 {
6     public static void main(String[] args) {
7         // bitwise operators AND, OR, XOR
8         Scanner scan = new Scanner(System.in);
9         System.out.println("ENTER 1ST NUMBER");
10        int num1 = scan.nextInt();
11        System.out.println("ENTER 2ND NUMBER");
12        int num2 = scan.nextInt();
13        if((num1&num2)==0) {
14            System.out.println("true");
15        }else if((num1|num2)==0) {
16            System.out.println("true");
17        }else if((num1^num2)==0) {
18            System.out.println("true");
19        }else {
20            System.out.println("false");
21        }
22    }
23}
24
```

Console output:  
ENTER 1ST NUMBER  
20  
ENTER 2ND NUMBER  
30  
FALSE

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

```

CORE JAVA - COREJAVAPROGRAMS375/src/operatorsprograms/Ques3.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Project Explorer   Ques3.java  Ques4.java  Ques5.java  Ques6.java  Ques7.java  Ques8.java  Ques9.java  Ques10.java  Ques11.java  Ques12.java  Ques13.java  Ques14.java  Ques15.java  Ques16.java  Ques17.java  Ques18.java  Ques19.java  Ques20.java  Ques21.java  Ques22.java  Ques23.java  Ques24.java  Ques25.java  module-info.java
1 package operatorsprograms;
2 import java.util.Scanner;
3
4 public class Ques3 {
5
6     public static void main(String[] args) {
7         Scanner num = new Scanner(System.in);
8         System.out.print("ENTER ANY NUMBER");
9         int num=num.nextInt();
10        if(num>0) {
11            System.out.println(num+" "+"IT IS POSITIVE NUMBER");
12        }
13        else if(num==0) {
14            System.out.println(num+" "+"IT IS ZERO NUMBER");
15        }
16        else {
17            System.out.println(num+" "+"IT IS NEGATIVE NUMBER");
18        }
19    }
20 }

```

The code is a Java program named Ques3. It uses the Scanner class to read an integer from the user. If the number is greater than zero, it prints "IT IS POSITIVE NUMBER". If the number is zero, it prints "IT IS ZERO NUMBER". If the number is less than zero, it prints "IT IS NEGATIVE NUMBER". The output window shows the message "18 IT IS POSITIVE NUMBER".

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

```

CORE JAVA - COREJAVAPROGRAMS375/src/operatorsprograms/Ques4.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Project Explorer   Ques4.java  Ques5.java  Ques6.java  Ques7.java  Ques8.java  Ques9.java  Ques10.java  Ques11.java  Ques12.java  Ques13.java  Ques14.java  Ques15.java  Ques16.java  Ques17.java  Ques18.java  Ques19.java  Ques20.java  Ques21.java  Ques22.java  Ques23.java  Ques24.java  Ques25.java  module-info.java
1 package operatorsprograms;
2
3 public class Ques4 {
4
5     public static void main(String[] args) {
6         // swap two numbers using temp
7         int a=10;
8         int b=20;
9         int temp=a ;
10        a=b-temp;
11        b=temp;
12        System.out.println(a);
13        System.out.println(b);
14        //swapping using XOR operation
15        int c=30;
16        int d=40;
17        c=c^d;
18        d=c^d;
19        c=c^d;
20        System.out.println(c);
21        System.out.println(d);
22    }
23 }

```

The code is a Java program named Ques4. It demonstrates two ways to swap two integers. The first method uses a temporary variable. The second method uses the XOR operator to swap the values without a temporary variable. The output window shows the swapped values: 20 and 10.

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

```

CORE JAVA - COREJAVAPROGRAMS375/src/operatorsprograms/Ques5.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Project Explorer   Ques5.java  Ques6.java  Ques7.java  Ques8.java  Ques9.java  Ques10.java  Ques11.java  Ques12.java  Ques13.java  Ques14.java  Ques15.java  Ques16.java  Ques17.java  Ques18.java  Ques19.java  Ques20.java  Ques21.java  Ques22.java  Ques23.java  Ques24.java  Ques25.java  module-info.java
1 package operatorsprograms;
2
3 import java.util.Scanner;
4
5 public class Ques5 {
6
7     public static void main(String[] args) {
8         // area of circle
9         Scanner scan = new Scanner(System.in);
10        System.out.print("ENTER RADIUS");
11        int r=scan.nextInt();
12        System.out.println("THE AREA OF CIRCLE IS:"+(3.14)*(r*r));
13        scan.close();
14    }
15 }

```

The code is a Java program named Ques5. It calculates the area of a circle using the formula  $\pi r^2$ . It reads the radius from the user using a Scanner. The output window shows the area for a radius of 7, which is approximately 153.94.

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

CORE JAVA - COREJAVA/PROGRAMS375/src/operatorsprograms/Ques6.java - Eclipse IDE

```

1 package operatorsprograms;
2 import java.util.Scanner;
3
4 public class Ques6 {
5
6     public static void main(String[] args) {
7         // TEMP TO FAHRENHEAT
8         Scanner scan = new Scanner(System.in);
9         System.out.println("ENTER TEMPERATURE");
10        double temp=scan.nextDouble();
11        double res=(temp*9/5)+32;
12        System.out.println("FAHERNHEAT IS"+ " "+res);
13        scan.close();
14    }
15 }
16
17 }
18

```

Console

```

terminated: Ques6 [Java Application] C:\eclipse\jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt.core\bin\jre\full\win32\x86_64_17.0.5.v20221102-0933\bin\javaw.exe (Feb 22, 2025, 12:17:15 PM - 12:17:18 PM) [pid: 2740]
ENTER TEMPERATURE
25
FAHERNHEAT IS 77.0

```

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

CORE JAVA - COREJAVA/PROGRAMS375/src/operatorsprograms/Ques7.java - Eclipse IDE

```

1 package operatorsprograms;
2 import java.util.Scanner;
3
4 public class Ques7 {
5
6     public static void main(String[] args) {
7         // given number is divisible by 5 and 7
8         Scanner scan = new Scanner(System.in);
9         System.out.println("ENTER ANY NUMBER");
10        int num=scan.nextInt();
11        if(num%5==0&&num%7==0) {
12            System.out.println(num+" "+"IT IS DIVISIBLE BY 5 AND 7");
13        } else {
14            System.out.println(num+" "+"IT IS NOT DIVISIBLE BY 5 AND 7");
15        }
16        scan.close();
17    }
18 }
19
20

```

Console

```

terminated: Ques7 [Java Application] C:\eclipse\jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt.core\bin\jre\full\win32\x86_64_17.0.5.v20221102-0933\bin\javaw.exe (Feb 22, 2025, 12:22:54 PM - 12:22:56 PM) [pid: 1560]
ENTER ANY NUMBER
55
IT IS NOT DIVISIBLE BY 5 AND 7

```

## 8) Write a program to calculate the compound interest.

CORE JAVA - COREJAVA/PROGRAMS375/src/operatorsprograms/Ques8.java - Eclipse IDE

```

1 package operatorsprograms;
2
3 public class Ques8 {
4
5     public static void main(String[] args) {
6         // calculate the compound interest
7         double principal = 1000;
8         double rate = 5;
9         int years = 3;
10
11        double amount = principal * Math.pow(1 + rate /100, years);
12        System.out.println("Amount after " + years + " years: " + amount);
13
14
15
16
17
18
19
20
21

```

Console

```

terminated: Ques8 [Java Application] C:\eclipse\jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt.core\bin\jre\full\win32\x86_64_17.0.5.v20221102-0933\bin\javaw.exe (Feb 22, 2025, 12:54:17 PM - 12:54:31 PM) [pid: 2905]
Amount after 3 years: 1157.6250000000002

```

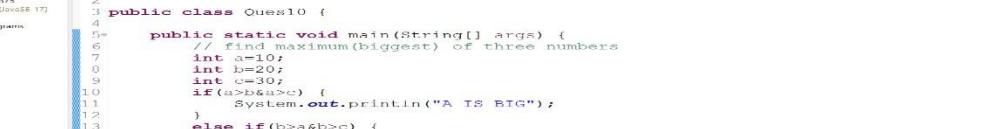
## 9) Write a program to check whether a given character is a vowel or consonant.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - CORE JAVA/ECLIPSE/Java/ECLIPSE/operatorsprograms/Quest9.java - Eclipse IDE
- Project Explorer:** Shows the project structure under "CORE JAVA/operatorsprograms".
- Code Editor:** Displays the Java code for "Quest9.java". The code checks if a character is a vowel or consonant.
- Console:** Shows the output of the program: "ENTER ANY CHARACTER", followed by "C", and "C IS A CONSONENT".
- Bottom Bar:** Shows the system tray and taskbar with various icons.

```
package operatorsprograms;
import java.util.Scanner;
public class Quest9 {
    public static void main(String[] args) {
        // Given character is vowel or consonant
        Scanner scan = new Scanner(System.in);
        System.out.println("ENTER ANY CHARACTER");
        char c=scan.next().charAt(0);
        c=cCharacter.toLowerCase(c);
        if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u') {
            System.out.println(c+" "+"IT IS A VOWEL");
        }
        else {
            System.out.println(c+" "+"IS A CONSONANT");
        }
        scan.close();
    }
}
```

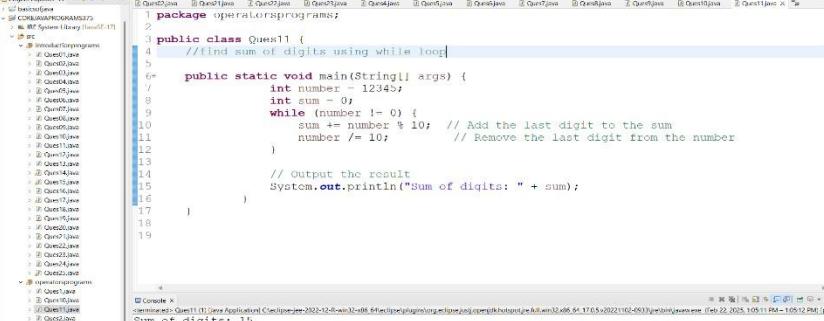
10) Write a program to find the maximum of three numbers using conditional operator.



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORALJAVA". The package structure includes "operatorprograms" and "operatorsprograms". Inside "operatorprograms", there are 24 files: Ques01.java through Ques24.java, and module-info.java.
- Code Editor:** Displays the content of Ques10.java. The code defines a class Ques10 with a main method that prints "C IS BIG!" if  $a > b & a > c$ , "B IS BIG!" if  $b > a & b > c$ , or "A IS BIG!" if  $c > a & c > b$ . It also contains comments explaining the logic.
- Console:** Shows the output of running the program: "C IS BIG!"
- Task List:** Shows several tasks related to the project, such as "Ques10.java", "Ques11.java", "Ques12.java", etc.

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



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with a package named "operatorsprograms" containing a class "Ques11".
- Code Editor:** Displays the Java code for "Ques11". The code defines a package "operatorsprograms" and a class "Ques11". The main method takes a string argument and prints the sum of digits of the number 12345 using a while loop.
- Status Bar:** Shows the file path as "C:\Users\DELL\Desktop\Java Application\Calculator.java" and the current time as "Feb 22, 2015, 10:51:11 PM".
- Bottom Icons:** Includes icons for Home, Search, Minimize, Maximize, and Close.

```
package operatorsprograms;
public class Ques11 {
    public static void main(String[] args) {
        int number = 12345;
        int sum = 0;
        while (number != 0) {
            sum += number % 10; // Add the last digit to the sum
            number /= 10; // Remove the last digit from the number
        }
        // Output the result
        System.out.println("Sum of digits: " + sum);
    }
}
```

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

```

 1 package operatorsprograms;
 2
 3 public class Ques12 {
 4
 5     public static void main(String[] args) {
 6         // given number is palindrome or not
 7         int num=12321;
 8         int num2=num;
 9         int res=0;
10         while(num!=0) {
11             res=res*10+num%10;
12             num/=10;
13         }
14         if(num2==res) {
15             System.out.println("IT IS PALINDROME");
16         }
17         else {
18             System.out.println("IT IS NOT A PALINDROME");
19         }
20     }
21 }
22

```

Console output: IT IS PALINDROME

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

```

 1 package operatorsprograms;
 2
 3 public class Ques13 {
 4
 5     public static void main(String[] args) {
 6         Scanner scan = new Scanner(System.in);
 7         System.out.print("ENTER ANY NUMBER");
 8         int num=scan.nextInt();
 9         int count=0;
10         int i=1;
11         while(i<num) {
12             if(num%i==0) {
13                 count++;
14             }
15             i++;
16         }
17         if(count==2) {
18             System.out.println("IT IS A PRIME NUMBER");
19         }
20         else {
21             System.out.println("IT IS NOT A PRIME NUMBER");
22         }
23     }
24 }
25

```

Console output: ENTER ANY NUMBER  
3  
IT IS A PRIME NUMBER

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

```

 1 package operatorsprograms;
 2
 3 import java.util.Scanner;
 4
 5 //factorial of number using recursion
 6 public class Ques14 {
 7     public static int isFact(int a) {
 8         if(a<0||a==1) {
 9             return 1;
10         }
11         else {
12             return a*isFact(a-1);
13         }
14     }
15
16     public static void main(String[] args) {
17         Scanner scan = new Scanner(System.in);
18         System.out.print("ENTER A NUMBER");
19         int num=scan.nextInt();
20         System.out.println(Ques14.isFact(num));
21         scan.close();
22     }
23 }
24

```

Console output: ENTER A NUMBER  
120

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "CORE JAVA - COREJAVA PROGRAMS375/src/operatorsprograms".
- Code Editor:** Displays the Java code for "Ques15.java".
- Console:** Shows the output of running the program. It prompts for "ENTER BASE" (2), "ENTER POWER" (3.0), and prints the result "8.0".
- System Tray:** Shows weather information ("Mostly cloudy") and system status.

```

1 package operatorsprograms;
2 import java.util.Scanner;
3 public class Ques15 {
4     //calculate power of number using recursion
5     public static double powerNum(int x,double y) {
6         if(x==0) {
7             return 0;
8         }
9         else {
10            return Math.pow(x, y);
11        }
12    }
13
14    public static void main(String[] args) {
15        Scanner scan = new Scanner(System.in);
16        System.out.print("ENTER BASE");
17        int base = scan.nextInt();
18        System.out.print("ENTER POWER ");
19        double pow = scan.nextDouble();
20        System.out.println(Ques15.powerNum(base, pow));
21        scan.close();
22    }
}

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "CORE JAVA - COREJAVA PROGRAMS375/src/operatorsprograms".
- Code Editor:** Displays the Java code for "Ques16.java".
- Console:** Shows the output of running the program. It prompts for "ENTER NUMBER" (5) and prints the Fibonacci sequence "0 1 2 3 5".
- System Tray:** Shows weather information ("AUS - ENG Connected") and system status.

```

1 package operatorsprograms;
2 import java.util.Scanner;
3 public class Ques16 {
4     public static int fiboc(int x) {
5         if(x==0) {
6             return 0;
7         }
8         else if(x==1) {
9             return 1;
10        }
11        return fiboc(x-1)+fiboc(x-2);
12    }
13
14    public static void main(String[] args) {
15        Scanner scan = new Scanner(System.in);
16        System.out.print("ENTER NUMBER");
17        int num=scan.nextInt();
18        for(int i=0;i<=num;i++) {
19            System.out.print(fiboc(i)+" ");
20        }
21        scan.close();
22    }
}

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "CORE JAVA - COREJAVA PROGRAMS375/src/operatorsprograms".
- Code Editor:** Displays the Java code for "Ques17.java".
- Console:** Shows the output of running the program. It prompts for "Enter any string" ("HAPPY") and prints the reversed string "YPPAH".
- System Tray:** Shows weather information ("Mostly cloudy") and system status.

```

1 package operatorsprograms;
2 import java.util.Scanner;
3 public class Ques17 {
4     public static String rev(String strl) {
5         if(strl.isEmpty()) {
6             return strl;
7         }
8         return rev(strl.substring(1))+strl.charAt(0);
9     }
10
11
12    public static void main(String[] args) {
13        // reverse a string using recursion
14        Scanner sc=new Scanner(System.in);
15        System.out.println("Enter any string");
16        String str =sc.nextLine();
17        System.out.println(rev(str));
18        sc.close();
19    }
}

```

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

**Project Explorer**

- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- operatorsprograms
- Ques01.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- operatorsprograms

```

1 package operatorsprograms;
2
3 public class Ques18 {
4
5     public static void main(String[] args) {
6         // sum of numbers
7         int n=5;
8         int sum=0;
9         for(int i=1; i<=n; i++) {
10             sum+=i;
11         }
12         System.out.println("THE SUM OF ALL NUMBERS IS"+sum);
13     }
14 }
15
16 }
17

```

**Console**

```
<terminated> Ques18 (Java Application) C:\eclipse-jee-2022-12-R\win32-x86_64\eclipse\plugins\org.eclipse.jdt.core\openjdk\hotspot\jre\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (Feb 22, 2025, 11:54:11 PM - 11:54:11 PM)
THE SUM OF ALL NUMBERS IS 15
```

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

**Project Explorer**

- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- operatorsprograms
- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- operatorsprograms

```

1 package operatorsprograms;
2
3 import java.util.Scanner;
4
5 class Ques19 {
6
7     public static void main(String[] args) {
8         // leap or not using with conditional operator
9         Scanner sc = new Scanner(System.in);
10        System.out.print("ENTER A YEAR");
11        int y = sc.nextInt();
12        String s=(y%4==0)&(y%100==0)&(y%400==0)?("IT is a LEAP YEAR"):(("IT is NOT a LEAP YEAR"));
13        System.out.println(s);
14
15    }
16
17
18

```

**Console**

```
<terminated> Ques19 (Java Application) C:\eclipse-jee-2022-12-R\win32-x86_64\eclipse\plugins\org.eclipse.jdt.core\openjdk\hotspot\jre\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (Feb 23, 2025, 12:03:29 AM - 12:03:12 AM)
ENTER A YEAR
2004
IT is a LEAP YEAR
```

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

**Project Explorer**

- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- operatorsprograms
- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- operatorsprograms

```

1 package operatorsprograms;
2
3 public class Ques20 {
4
5     public static void main(String[] args) {
6         // lcm of two numbers
7         int a = 12;
8         int b = 18;
9         // Calculate LCM using GCD
10        int lcm = (a * b) / gcd(a, b);
11        System.out.println("LCM OF " + a + " and " + b + " is: " + lcm);
12
13        // Function to calculate GCD using Euclidean algorithm
14        public static int gcd(int a, int b) {
15            while (b != 0) {
16                int temp = b;
17                b = a % b;
18                a = temp;
19            }
20            return a;
21        }
22
23
24
25
26
27
28
29
2

```

**Console**

```
<terminated> Ques20 (Java Application) C:\eclipse-jee-2022-12-R\win32-x86_64\eclipse\plugins\org.eclipse.jdt.core\openjdk\hotspot\jre\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (Feb 23, 2025, 12:09:16 AM - 12:09:17 AM)
LCM of 12 and 18 is: 36
```

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

The screenshot shows the Eclipse IDE interface with the following code in the editor:

```
1 package operatorsprograms;
2
3 public class Ques21 {
4
5     public static void main(String[] args) {
6         // area of triangle using formula
7         int a=5;
8         int b=6;
9         int c=7;
10        double s = (a+b+c)/2;
11        double A = Math.sqrt(s*(s-a)*(s-b)*(s-c));
12        System.out.println(A);
13    }
14
15
16 }
```

The code calculates the area of a triangle with side lengths 5, 6, and 7 using Heron's formula. The output is displayed in the Console tab.

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

The screenshot shows the Eclipse IDE interface with the following code in the editor:

```
1 package operatorsprograms;
2
3 public class Ques22 {
4
5     public static void main(String[] args) {
6         // sum of all even numbers from 1to100 using for loop
7         int sum=0;
8         int i=1;
9         for(int i=1;i<=100;i+=2) {
10             sum+=i;
11         }
12         System.out.println(sum);
13     }
14
15
16 }
```

The code uses a for loop to iterate through even numbers from 1 to 100 and calculates their sum. The output is displayed in the Console tab.

23) Write a program to calculate the simple interest.

The screenshot shows the Eclipse IDE interface with the following code in the editor:

```
1 package operatorsprograms;
2
3 public class Ques23 {
4
5     public static void main(String[] args) {
6         // SIMPLE INTEREST
7         int P=1000;
8         int R=5;
9         int T=6;
10        int SI=(P*T*R)/100;
11        System.out.println("SIMPLE INTEREST IS "+SI);
12    }
13
14
15 }
```

The code calculates simple interest using the formula  $SI = \frac{P \times R \times T}{100}$ . The output is displayed in the Console tab.

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

```

public class Ques24 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("ENTER ANY NUMBER");
        int num = sc.nextInt();
        for(int i=1;i<=10;i++) {
            System.out.println(num+"*"+i+" = "+(num*i));
        }
        sc.close();
    }
}

```

Output:

```

ENTER ANY NUMBER
2*1 = 2
2*2 = 4
2*3 = 6
2*4 = 8
2*5 = 10
2*6 = 12
2*7 = 14
2*8 = 16
2*9 = 18
2*10 = 20

```

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

```

public class Ques25 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("ENTER ANY NUMBER");
        int num=sc.nextInt();
        String s1=Integer.toString(num);
        int s2=s1.length();
        System.out.println(s2);
        int sum=0;
        int Temp=num;
        while(Temp>0) {
            int digits=Temp%10;
            sum+=Math.pow(digits,s2);
            Temp/=10;
        }
        if(sum==num) {
            System.out.println("IT IS ARMSTRONG");
        }
        else {
            System.out.println("IT IS NOT A ARMSTRONG");
        }
    }
}

```

Output:

```

ENTER ANY NUMBER
153
3
IT IS ARMSTRONG

```

## ARRAYS:

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

```

public class Ques01 {
    public static void main(String[] args) {
        // Find sum of elements in the array
        int arr[]={1,2,3,5,6,7};
        int sum=0;
        for(int i=0;i<arr.length;i++) {
            sum=arr[i]+sum;
        }
        System.out.println("THE SUM OF ELEMENTS OF ARRAY IS : "+sum);
    }
}

```

Output:

```

THE SUM OF ELEMENTS OF ARRAY IS : 22

```

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

**3. Write a program to find largest and smallest elements in an array.**

```

1 package arraysprograms;
2 import java.util.*;
3 public class Ques02 {
4
5     public static void main(String[] args) {
6         // find largest and smallest elements in array
7         int[] a= {1,7,5,3,8};
8         int len=a.length;
9         System.out.println(len);
10        Arrays.sort(a);
11        System.out.println("THE LARGEST ELEMENTS IN ARRAY IS"+a[len-1]);
12        System.out.println();
13        System.out.println("THE LARGEST ELEMENTS IN ARRAY IS"+a[0]);
14
15
16
17    }
18
19
20 }

```

The console output shows:

```

THE LARGEST ELEMENTS IN ARRAY IS 8
THE LARGEST ELEMENTS IN ARRAY IS 1

```

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

```

3 public class Ques03 {
4     public static void main(String[] args) {
5         // copy one array to another array
6         int[] a=new int[5];
7         int b[]=new int[5];
8         Scanner scan = new Scanner(System.in);
9         System.out.println("enter first array elements");
10        for(int i=0;i<a.length;i++) {
11            a[i]=scan.nextInt();
12        }
13        System.out.println("YOUR ARRAY ELEMENTS ARE:");
14        for(int i=0;i<a.length;i++) {
15            System.out.print(a[i]+" ");
16        }
17        System.out.println("\nenter second array elements");
18        for(int i=0;i<b.length;i++) {
19            b[i]=a[i];
20            System.out.print(b[i]+" ");
21        }
22    }

```

The console output shows:

```

enter first array elements
1 2 3 4 5
YOUR ARRAY ELEMENTS ARE:
1 2 3 4 5
enter second array elements
1 2 3 4 5

```

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

```

1 package arraysprograms;
2
3 public class Ques04 {
4
5     public static void main(String[] args) {
6         // remove duplicates elements in the array
7         int[] a= {1,2,2,3,4,4,5};
8         int j=1;
9         for(int i=0;i<a.length-1;i++) {
10            if(a[i]==a[i+1]) {
11                a[j]=a[i+1];
12                j++;
13            }
14        }
15        for(int i=0;i<j;i++) {
16            System.out.print(a[i]+ " ");
17        }
18    }
19
20 }

```

The console output shows:

```

1 2 3 4 5

```

### 5. Write a program to reverse an array.

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - COREJAVAPROGRAMS375/src/arraysprograms/Ques05.java - Eclipse IDE". The Project Explorer view on the left shows a project named "arraysprograms" containing several Java files like Ques01.java through Ques05.java. The code editor displays the following Java code:

```
1 package arraysprograms;
2
3 public class Ques05 {
4
5     public static void main(String[] args) {
6         // reverse an array
7         int [] a = {1,2,3,4,5};
8         System.out.println("REVERSE OF ANY ARRAY");
9         for(int i=a.length-1; i>=0; i--) {
10             System.out.print(a[i]+" ");
11         }
12     }
13
14 }
15
16
17 }
18
19 }
```

The console output window at the bottom shows the reversed array: "REVERSE OF ANY ARRAY" followed by the numbers 5 4 3 2 1.

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

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - COREJAVAPROGRAMS375/src/arraysprograms/Ques06.java - Eclipse IDE". The Project Explorer view on the left shows a project named "arraysprograms" containing files Ques05.java and Ques06.java. The code editor displays the following Java code:

```
1 package arraysprograms;
2 import java.util.Arrays;
3 public class Ques06 {
4     public static void main(String[] args) {
5         // sort an array in ascending and descending order
6         int[] a = {1,3,2,6,5};
7         Arrays.sort(a);
8         System.out.println("THE ASENDING ORDER OF AN ARRAY");
9         for(int i=0;i<a.length; i++) {
10             System.out.print(a[i]+" ");
11         }
12         System.out.println();
13         System.out.println("THE DESCENDING ORDER OF AN ARRAY IS");
14         for(int i=a.length-1; i>=0; i--) {
15             System.out.print(a[i]+" ");
16         }
17     }
18 }
19 }
```

The console output window at the bottom shows two sorted arrays: "THE ASENDING ORDER OF AN ARRAY" (1 2 3 5 6) and "THE DESCENDING ORDER OF AN ARRAY IS" (6 5 3 2 1).

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

```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer [Ques07.java] [Ques08.java] [Ques09.java] [Ques10.java] [Ques11.java] [Ques12.java]
1 package arraysprograms;
2
3 import java.util.Scanner;
4
5 public class Ques07 {
6
7     public static void main(String[] args) {
8         // Find the frequency of an element in array
9         int[] a = {1, 2, 3, 4, 4, 5};
10        Scanner scan = new Scanner(System.in);
11        System.out.println("ENTER ANY ELEMENT IN ARRAY");
12        int e = scan.nextInt();
13        int f = 0;
14        System.out.println("THE FREQUENCY OF AN ELEMENT IS");
15        for (int i = 0; i < a.length; i++) {
16            if (a[i] == e) {
17                f++;
18            }
19        }
20        System.out.println(f);
21    }
22}
```

Console > terminated: Ques07 (1) [Java Application] C:\eclipse\jee-2022-12-R-win32-x86\_64\plugins\org.eclipse.jdt.core\openjdk\hotspot\jre\full\win32\x86\_64\_17.0.5.v20221102-0933\jre\bin\javaw.exe (Feb 24, 2025, 10:05:17 PM) [pid: 34952]

ENTER ANY ELEMENT IN ARRAY

THE FREQUENCY OF AN ELEMENT IS

## 8) Write a program to merge two sorted arrays.

```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer [Ques06.java] [Ques07.java] [Ques08.java] [Ques09.java] [Ques10.java] [Ques11.java]
1 package arraysprograms;
2
3 import java.util.Arrays;
4
5 public class Ques08 {
6
7     public static void main(String[] args) {
8         // merge two sorted arrays
9         int[] a = {1, 2, 3, 4, 5};
10        int[] b = {6, 7, 8, 9, 10};
11        int[] c = new int[a.length+b.length];
12        for(int i=0;i<a.length;i++) {
13            c[i]=a[i];
14        }
15        for(int i=0;i<b.length;i++) {
16            c[a.length+i]=b[i];
17        }
18        System.out.println(" " + Arrays.toString(c));
19    }
20}
```

Console > terminated: Ques08 (1) [Java Application] C:\eclipse\jee-2022-12-R-win32-x86\_64\plugins\org.eclipse.jdt.core\openjdk\hotspot\jre\full\win32\x86\_64\_17.0.5.v20221102-0933\jre\bin\javaw.exe (Feb 24, 2025, 10:22:21 PM) [pid: 16080]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

System > 24°C Partly cloudy

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

```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer [Ques06.java] [Ques07.java] [Ques08.java] [Ques09.java] [Ques10.java] [Ques11.java]
1 package arraysprograms;
2
3 public class Ques09 {
4
5     public static void main(String[] args) {
6         // find intersection of two arrays
7         int[] a = {1, 2, 3};
8         int[] b = {2, 3, 4};
9         System.out.println("THE INTERSECTION OF TWO ARRAYS");
10        for(int i=0;i<a.length;i++) {
11            for(int j=0;j<b.length;j++) {
12                if(a[i]==b[j]) {
13                    System.out.println(a[i]);
14                }
15            }
16        }
17    }
18}
19
```

Console > terminated: Ques09 (1) [Java Application] C:\eclipse\jee-2022-12-R-win32-x86\_64\plugins\org.eclipse.jdt.core\openjdk\hotspot\jre\full\win32\x86\_64\_17.0.5.v20221102-0933\jre\bin\javaw.exe (Feb 24, 2025, 11:10:18 PM) [pid: 34216]

THE INTERSECTION OF TWO ARRAYS

System > 23°C Partly cloudy

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

```
1 package arraysprograms;
2 public class Ques10 {
3     public static void main(String[] args) {
4         // check whether an array is a palindrome or not.
5         int[] arr = {1, 2, 3, 2, 1};
6         int l = 0; int r = arr.length - 1;
7         for(int i = 0; i < arr.length / 2; i++) {
8             if(arr[i] != arr[r]) {
9                 System.out.println("NOT A PALINDROME");
10            }
11        }
12        if(l == r) {
13            System.out.println("IT IS A PALINDROME");
14        } else {
15            System.out.println("IT IS NOT A PALINDROME");
16        }
17    }
18 }
```

The code checks if the array [1, 2, 3, 2, 1] is a palindrome. It iterates through the first half of the array, comparing elements at index *i* with elements at index *r* (from the end). Since all elements are equal, it prints "IT IS A PALINDROME".

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

## 12. Write a program to fin the sum of all negative numbers in an array.

```
1 package arraysprograms;
2 public class Ques12 {
3     public static void main(String[] args) {
4         // sum of all the negative numbers
5         int[] a = {-1, -2, -3, -4, -5};
6         int sum=0;
7         for(int i=0;i<a.length;i++) {
8             sum+=a[i];
9         }
10        System.out.println(sum);
11    }
12 }
```

The code initializes an array with negative integers [-1, -2, -3, -4, -5] and sums them up. The output in the console is -15.

d

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

```
1 package arraysprograms;
2 public class Ques13 {
3     public static void main(String[] args) {
4         // product of elements in an array
5         int[] a = {1, 2, 3, 4, 5};
6         int prod=1;
7         for(int i=0;i<a.length;i++) {
8             prod*=a[i];
9         }
10        System.out.println(prod);
11    }
12 }
```

The code initializes an array [1, 2, 3, 4, 5] and calculates the product of all elements. The output in the console is 120.

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

```

1 CORE JAVA - COREJAVAPROGRAMS575/src/arrayprograms/Quat14.java - Eclipse IDE
2 File Edit Source Refactor Navigate Project Run Window Help
3 Project Explorer 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
4 package arraysprogram;
5
6 import java.util.Arrays;
7
8 public class Quat14 {
9
10     public static void main(String[] args) {
11         // To find second largest and second smallest element in an array
12         int[] arr = {1, 3, 5, 2, 7, 0, 6};
13         int len = arr.length;
14         Arrays.sort(arr);
15         System.out.println("THE SECOND LARGEST NUMBER IS "+arr[len-2]);
16         System.out.println();
17         System.out.println("THE SECOND SMALLEST NUMBER IS "+arr[1]);
18     }
19 }

```

Console

```

100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
THE SECOND LARGEST NUMBER IS 7
THE SECOND SMALLEST NUMBER IS 2

```

## **15 Write a program to find the index of a given element in an array**

```

1 CORE JAVA - COREJAVAPROGRAMS575/src/arrayprograms/Quat15.java - Eclipse IDE
2 File Edit Source Refactor Navigate Project Run Window Help
3 Project Explorer 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
4 package arraysprogram;
5
6 import java.util.Scanner;
7
8 public class Quat15 {
9
10     public static void main(String[] args) {
11         // Find the index of an element in an array
12         int[] arr = {1, 3, 4, 5};
13         int index = 0;
14         Scanner scan = new Scanner(System.in);
15         System.out.println("ENTER ANY ELEMENT FROM ARRAY");
16         int elem = scan.nextInt();
17         for(int i=0;i<arr.length;i++) {
18             if(arr[i]==elem) {
19                 index=i;
20             }
21         }
22         System.out.println(index+" IS THE INDEX OF ARRAY");
23     }
24 }

```

Console

```

100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
ENTER ANY ELEMENT FROM ARRAY
5
5 IS THE INDEX OF ARRAY

```

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

```

1 CORE JAVA - COREJAVAPROGRAMS575/src/arrayprograms/Quat16.java - Eclipse IDE
2 File Edit Source Refactor Navigate Project Run Window Help
3 Project Explorer 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
4 package arraysprogram;
5
6 import java.util.Scanner;
7
8 public class Quat16 {
9
10     public static void main(String[] args) {
11         // Program to rotate an array to the left to right
12         int[] arr = {1, 2, 3, 4, 5, 6, 7};
13         Scanner scan = new Scanner(System.in);
14         System.out.println("ENTER HOW TIME SHIFT TO LEFT TO RIGHT");
15         int shift = scan.nextInt();
16         int temp[] = new int[arr.length];
17         int d=arr.length;
18         for(int i=0; i<arr.length; i++) {
19             temp[i]=arr[i];
20         }
21         int i;
22         for(i=d-shift; i<arr.length; i++) {
23             arr[i-d+shift]=arr[i];
24         }
25         for(j=0; j<shift; j++) {
26             arr[j]=temp[d-j];
27         }
28     }
29 }

```

Console

```

100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
ENTER HOW TIME SHIFT TO LEFT TO RIGHT
2
3 4 5 6 7 1 2

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVA PROGRAMS375" containing several Java files under "src/arraysprograms".
- Code Editor:** Displays the content of "Ques17.java" which prints the elements of a 2D array in spiral order.
- Console:** Shows the output of running the program, prompting for input and displaying the spiral printout.
- Taskbar:** Shows the system tray with icons for battery, signal, and date/time.

```

1 package arraysprograms;
2 import java.util.Scanner;
3 public class Ques17 {
4     public static void main(String[] args) {
5         // print the elements of the 2d array in spiral order
6         Scanner scan = new Scanner(System.in);
7         System.out.println("enter equal number of rows and columns");
8         int r=scan.nextInt();
9         int c=scan.nextInt();
10        int arr[][]= new int[r][c];
11        System.out.println("enter array elements");
12        for(int i=0;i<a.length;i++) {
13            for(int j=0;j<a.length;j++) {
14                arr[i][j]=scan.nextInt();
15            }
16        }
17        int rs=0;
18        int re=a.length-1;
19        int cs=0;
20        int ce=a.length-1;
21        int r=rs;
22        int c=cs;
23        while(r<re && c<ce) {
24            if(c==ce) {
25                r++;
26            }
27            if(r==re) {
28                c--;
29            }
30            if(r>rs && c<ce) {
31                System.out.print(arr[r][c]);
32            }
33            if(r==re && c>cs) {
34                r--;
35            }
36            if(r<rs && c>cs) {
37                c--;
38            }
39        }
40    }
41 }

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVA PROGRAMS375" containing several Java files under "src/arraysprograms".
- Code Editor:** Displays the content of "Ques18.java" which checks if two arrays are equal.
- Console:** Shows the output of running the program, comparing two arrays and printing the result.
- Taskbar:** Shows the system tray with icons for battery, signal, and date/time.

```

1 package arraysprograms;
2 import java.util.Arrays;
3 public class Ques18 {
4     public static void main(String[] args) {
5         // program to check whether the two arrays are equals or not
6         int [] a = {1,2,5,6,7};
7         int [] b = {1,2,5,6,7};
8         if(Arrays.equals(a, b)) {
9             System.out.println("arrays are equal");
10        } else {
11            System.out.println("arrays are not equal");
12        }
13    }
14 }

```

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

**triangle of a matrix.**

```

1 package arraysprograms;
2
3 public class Ques19 {
4
5     public static void main(String[] args) {
6         // find the sum of elements in the upper triangle
7         int a [][]= {{1,2,3},
8                     {4,5,6},
9                     {7,8,9}};
10
11         int sum=0;
12         for(int i=0;i<a.length;i++) {
13             for(int j=i;j<a.length;j++) {
14                 sum+=a[i][j];
15             }
16         }
17
18         System.out.println("the sum of elements in upper triangle is" + sum);
19     }
20 }

```

The screenshot shows the Eclipse IDE interface with the code for Question 19. The code calculates the sum of elements in the upper triangle of a 3x3 matrix. The output window shows the result: "the sum of elements in upper triangle is 26".

## **20. Write a program to find the sum of elements in the lower row and column of a matrix.**

```

1 package arraysprograms2;
2
3 public class Ques20 {
4
5     public static void main(String[] args) {
6         // find the sum of elements in the lower triangle
7         int a [][]= {{1,2,3},
8                     {4,5,6},
9                     {7,8,9}};
10
11         int sum=0;
12         for(int i=0;i<a.length;i++) {
13             for(int j=0;j<=i;j++) {
14                 sum+=a[i][j];
15             }
16         }
17
18         System.out.println(" find the sum of elements in the lower triangle"+ sum);
19     }
20 }

```

The screenshot shows the Eclipse IDE interface with the code for Question 20. The code calculates the sum of elements in the lower triangle of a 3x3 matrix. The output window shows the result: "find the sum of elements in the lower triangle 26".

## **21. Write a program to find the sum of elements in each**

```

1 package arraysprograms;
2
3 public class Ques21 {
4
5     public static void main(String[] args) {
6         // find the sum of elements in each row and column of matrix
7         int a [][]= {{1,2,3},
8                     {4,5,6},
9                     {7,8,9}};
10
11         int sum=0;
12         for(int i=0;i<a.length;i++) {
13             for(int j=0;j<a[0].length;j++) {
14                 sum+=a[i][j];
15             }
16         }
17
18         System.out.println("the sum of elements in row and col are "+ sum);
19     }
20 }

```

The screenshot shows the Eclipse IDE interface with the code for Question 21. The code calculates the sum of elements in each row and column of a 3x3 matrix. The output window shows the result: "the sum of elements in row and col are 27".

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

```

1 package arraysprograms;
2
3 public class Ques22 {
4
5     public static void main(String[] args) {
6         // check given matrix is symmetric or not
7         int a [][]= {{1,2,3},
8                     {4,5,6},
9                     {7,8,9}};
10
11         System.out.println("the original matrix is");
12         for(int i=0;i<a.length;i++) {
13             for(int j=0;j<a[0].length;j++) {
14                 System.out.print(a[i][j] + " ");
15             }
16             System.out.println();
17         }
18
19         System.out.println("the transpose matrix is");
20         for(int i=0;i<a.length;i++) {
21             for(int j=0;j<a[0].length;j++) {
22                 System.out.print(a[j][i] + " ");
23             }
24             System.out.println();
25         }
26
27         if(a.equals(transpose))
28             System.out.println("it is symmetric");
29         else
30             System.out.println("it is not symmetric");
31     }
32 }

```

The screenshot shows the Eclipse IDE interface with the code for Question 22. The code checks if a given 3x3 matrix is symmetric by comparing it with its transpose. The output window shows the original matrix and its transpose, followed by a message indicating it is symmetric.

## **24 Write a program to find the k th smallest and kth largest elements in an array**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "arraysprogramz" containing several source files: Ques01.java, Ques02.java, Ques03.java, Ques04.java, Ques05.java, Ques06.java, Ques07.java, Ques08.java, Ques09.java, Ques10.java, Ques11.java, Ques12.java, Ques13.java, Ques14.java, Ques15.java, Ques16.java, Ques17.java, Ques18.java, Ques19.java, Ques20.java, Ques21.java, Ques22.java, Ques23.java, Ques24.java, Ques25.java, Ques26.java, Ques27.java, Ques28.java, Ques29.java, Ques30.java, and Ques31.java.
- Code Editor:** Displays the content of the file Ques25.java. The code defines a class Ques25 with a main method that initializes an array 'a' of size 3x3 with values 1, 2, 3, 4, 5, 6, 7, 8, 9. It then prints the transpose matrix and checks if it is symmetric by comparing it with the original matrix.
- Terminal:** Shows the output of the program: "the transpose matrix is" followed by the transpose matrix:  
1 2 3  
4 5 6  
7 8 9
- Status Bar:** Shows the current time as 8:00 PM and the date as 3/17/2025.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "arraysprograms" with several source files: Ques24.java, Ques25.java, Ques26.java, Ques27.java, Ques28.java, Ques29.java, Ques30.java, Ques31.java, Ques32.java, Ques33.java, Ques34.java, Ques35.java, Ques36.java, Ques37.java, Ques38.java, Ques39.java, Ques40.java, Ques41.java, Ques42.java, Ques43.java, Ques44.java, Ques45.java, Ques46.java, Ques47.java, Ques48.java, Ques49.java, Ques50.java, Ques51.java, Ques52.java, Ques53.java, Ques54.java, Ques55.java, Ques56.java, Ques57.java, Ques58.java, Ques59.java, Ques60.java, Ques61.java, Ques62.java, Ques63.java, Ques64.java, Ques65.java, Ques66.java, Ques67.java, Ques68.java, Ques69.java, Ques70.java, Ques71.java, Ques72.java, Ques73.java, Ques74.java, Ques75.java, Ques76.java, Ques77.java, Ques78.java, Ques79.java, Ques80.java, Ques81.java, Ques82.java, Ques83.java, Ques84.java, Ques85.java, Ques86.java, Ques87.java, Ques88.java, Ques89.java, Ques90.java, Ques91.java, Ques92.java, Ques93.java, Ques94.java, Ques95.java, Ques96.java, Ques97.java, Ques98.java, Ques99.java, Ques100.java, Ques101.java, Ques102.java, Ques103.java, Ques104.java, Ques105.java, Ques106.java, Ques107.java, Ques108.java, Ques109.java, Ques110.java, Ques111.java, Ques112.java, Ques113.java, Ques114.java, Ques115.java, Ques116.java, Ques117.java, Ques118.java, Ques119.java, Ques120.java, Ques121.java, Ques122.java, Ques123.java, Ques124.java, Ques125.java, Ques126.java, Ques127.java, Ques128.java, Ques129.java, Ques130.java, Ques131.java, Ques132.java, Ques133.java, Ques134.java, Ques135.java, Ques136.java, Ques137.java, Ques138.java, Ques139.java, Ques140.java, Ques141.java, Ques142.java, Ques143.java, Ques144.java, Ques145.java, Ques146.java, Ques147.java, Ques148.java, Ques149.java, Ques150.java, Ques151.java, Ques152.java, Ques153.java, Ques154.java, Ques155.java, Ques156.java, Ques157.java, Ques158.java, Ques159.java, Ques160.java, Ques161.java, Ques162.java, Ques163.java, Ques164.java, Ques165.java, Ques166.java, Ques167.java, Ques168.java, Ques169.java, Ques170.java, Ques171.java, Ques172.java, Ques173.java, Ques174.java, Ques175.java, Ques176.java, Ques177.java, Ques178.java, Ques179.java, Ques180.java, Ques181.java, Ques182.java, Ques183.java, Ques184.java, Ques185.java, Ques186.java, Ques187.java, Ques188.java, Ques189.java, Ques190.java, Ques191.java, Ques192.java, Ques193.java, Ques194.java, Ques195.java, Ques196.java, Ques197.java, Ques198.java, Ques199.java, Ques200.java, Ques201.java, Ques202.java, Ques203.java, Ques204.java, Ques205.java, Ques206.java, Ques207.java, Ques208.java, Ques209.java, Ques210.java, Ques211.java, Ques212.java, Ques213.java, Ques214.java, Ques215.java, Ques216.java, Ques217.java, Ques218.java, Ques219.java, Ques220.java, Ques221.java, Ques222.java, Ques223.java, Ques224.java, Ques225.java, Ques226.java, Ques227.java, Ques228.java, Ques229.java, Ques230.java, Ques231.java, Ques232.java, Ques233.java, Ques234.java, Ques235.java, Ques236.java, Ques237.java, Ques238.java, Ques239.java, Ques240.java, Ques241.java, Ques242.java, Ques243.java, Ques244.java, Ques245.java, Ques246.java, Ques247.java, Ques248.java, Ques249.java, Ques250.java, Ques251.java, Ques252.java, Ques253.java, Ques254.java, Ques255.java, Ques256.java, Ques257.java, Ques258.java, Ques259.java, Ques260.java, Ques261.java, Ques262.java, Ques263.java, Ques264.java, Ques265.java, Ques266.java, Ques267.java, Ques268.java, Ques269.java, Ques270.java, Ques271.java, Ques272.java, Ques273.java, Ques274.java, Ques275.java, Ques276.java, Ques277.java, Ques278.java, Ques279.java, Ques280.java, Ques281.java, Ques282.java, Ques283.java, Ques284.java, Ques285.java, Ques286.java, Ques287.java, Ques288.java, Ques289.java, Ques290.java, Ques291.java, Ques292.java, Ques293.java, Ques294.java, Ques295.java, Ques296.java, Ques297.java, Ques298.java, Ques299.java, Ques299.java, Ques300.java.
- Code Editor:** Displays the content of `Ques24.java`. The code defines a class `Ques24` with a main method that prints the  $k^{\text{th}}$  smallest and largest elements from an array of integers.
- Console:** Shows the output of the program, which includes the transpose matrix and a check for symmetry.

## DATATYPES:

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



The screenshot shows the Eclipse IDE interface with a Java project named "Ques01" open. The code in the editor prints various primitive data types:

```
public class Ques01 {
    public static void main(String[] args) {
        // program to use of primitive datatypes
        byte b=1;
        short s=2;
        int i=3;
        long l=4;
        float f=3.3f;
        double d=3.3455;
        System.out.println("byte-"+b);
        System.out.println("short-"+s);
        System.out.println("int-"+i);
        System.out.println("long-"+l);
        System.out.println("float-"+f);
        System.out.println("double-"+d);
    }
}
```

The output window shows the results of the println statements:

```
byte-1
short-2
int-3
long-4
float-3.3
double-3.3455
```

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

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - CORFIJAVAPROGRAMS5375/src/datatypesprograms/Ques02.java". The code implements basic arithmetic operations using float and double types. The output window displays the results of these operations.

```
package datatypesprograms;
public class Ques02 {
    public static void main(String[] args) {
        // perform arithmetic operations mainly using float and double
        float f=3.31;
        double d=5.555;
        System.out.println("sum-"+(f+d));
        System.out.println("sub-"+(f-d));
        System.out.println("mul-"+(f*d));
        System.out.println("div-"+(f/d));
        System.out.println("mod-"+(f%d));
    }
}
```

```
Console > Ques02 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jst\openjdk\hotspot\re\full\win32\x86_64\17.0.5.v20221102-0933\re\bin\javaw.exe (Feb 27, 2025, 11:55:50 PM - 11:55:51 PM) [pid: 21581]
sum=8.85499952316284
sub=-2.255000476837155
mul=18.331499735116957
div=0.5940593973566668
mod=3.299999952316284
```

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

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - CORFIJAVAPROGRAMS5375/src/datatypesprograms/Ques03.java". The code demonstrates how to convert an integer to a binary string and vice versa using Java's built-in methods.

```
package datatypesprograms;
import java.util.function.BinaryOperator;
public class Ques03 {
    public static void main(String[] args) {
        // convert to integer to binary to vice versa
        int a= 123;
        String s= Integer.toBinaryString(a);
        System.out.println("the intger to binary value is"+s);
        String b="1001";
        int i=Integer.parseInt(b,2);
        System.out.println("the binary to string is:"+i);
    }
}
```

```
Console > Ques03 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jst\openjdk\hotspot\re\full\win32\x86_64\17.0.5.v20221102-0933\re\bin\javaw.exe (Feb 28, 2025, 11:21:19 AM - 11:21:19 AM) [pid: 10464]
the intger to binary value is 1111011
the binary to string is 9
```

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

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - CORFIJAVAPROGRAMS5375/src/datatypesprograms/Ques04.java". The code illustrates how to convert characters from lowercase to uppercase and vice versa using Java's Character class methods.

```
package datatypesprograms;
public class Ques04 {
    public static void main(String[] args) {
        // operations on the characters to convert lower to upper case
        char c='A';
        char ch=Character.toLowerCase(c);
        System.out.println("the uppercase to lowercase is"+ch);
        char cl='d';
        char ch1=Character.toUpperCase(cl);
        System.out.println("the lowercase to uppercase is"+ch1);
    }
}
```

```
Console > Ques04 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jst\openjdk\hotspot\re\full\win32\x86_64\17.0.5.v20221102-0933\re\bin\javaw.exe (Feb 28, 2025, 12:10:32 AM - 12:10:32 AM) [pid: 24124]
the uppercase to lowercase is a
the lowercase to uppercase is D
```

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

Project Explorer: Ques05.java, Ques06.java, Ques07.java, Ques08.java, Ques09.java, Ques10.java

```
1 package datatypeprograms;
2
3 public class Ques05 {
4
5     public static void main(String[] args) {
6         // demonstration of boolean datatype
7         boolean a=true;
8         if(a==true) {
9             System.out.println("TRUE");
10        }
11        else {
12            System.out.println("FALSE");
13        }
14    }
15 }
16
17 }
```

Console: <terminated> Ques05 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86\_64\eclipse\plugins\org.eclipse.jst\openjdk\hotspot\re\full\win32\x86\_64\_17.0.v20221102-0933\re\bin\view.exe (Feb 20, 2025, 12:21:07 AM - 12:21:08 AM) [pid: 17952]

TRUE

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

Project Explorer: Ques06.java, Ques07.java, Ques08.java, Ques09.java, Ques10.java

```
1 package datatypeprograms;
2
3 public class Ques06 {
4
5     public static void main(String[] args) {
6         // demonstrate the use of arrays in java
7         int[] a={1,2,3,4,5};
8         System.out.println(a.length);
9         for(int i=0;i<a.length-1;i++) {
10             System.out.print(a[i]+",");
11         }
12     }
13 }
14
15 }
```

Console: <terminated> Ques06 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86\_64\eclipse\plugins\org.eclipse.jst\openjdk\hotspot\re\full\win32\x86\_64\_17.0.v20221102-0933\re\bin\view.exe (Feb 20, 2025, 12:20:10 AM - 12:20:20 AM) [pid: 17952]

5,4,3,2,1

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

Project Explorer: Ques07.java

```
1 package datatypeprograms;
2
3 import java.util.Arrays;
4
5 public class Ques07 {
6
7     public static void main(String[] args) {
8         // Find maximum and minimum element
9         int arr={3,7,2,4,9,8};
10        int n=arr.length;
11        Arrays.sort(arr);
12        System.out.println("THE MAXIMUM ELEMENT IS "+arr[n-1]);
13        System.out.println("THE MINIMUM ELEMENT IS "+arr[0]);
14    }
15 }
```

Console: <terminated> Ques07 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86\_64\eclipse\plugins\org.eclipse.jst\openjdk\hotspot\re\full\win32\x86\_64\_17.0.v20221102-0933\re\bin\view.exe (Feb 20, 2025, 12:30:10 AM - 12:30:11 AM) [pid: 21745]

THE MAXIMUM ELEMENT IS 9  
THE MINIMUM ELEMENT IS 2

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

The screenshot shows the Eclipse IDE interface with a Java project named "datatypesprograms". The code in the main class "Ques08" calculates the average of an array of integers. The console output shows the result: "the average of the array is= 3".

```
1 package datatypesprograms;
2
3 public class Ques08 {
4
5     public static void main(String[] args) {
6         // calculate the average of an array
7         int[] a={1,2,3,4,5};
8         int len=a.length;
9         int sum=0;
10        for(int i=0;i<a.length;i++) {
11            sum+=a[i];
12        }
13        System.out.println("sum-"+sum);
14        int avg=sum/len;
15        System.out.println("the average of the array is-"+avg);
16    }
17 }
18
19 }
20 }
```

```
Console
terminated: Ques08 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt\openjdt\hotspot\re\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\java.exe (Feb 28, 2025, 12:42:50 AM - 12:42:51 AM) [pid: 11604]
sum= 15
the average of the array is= 3
```

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

The screenshot shows the Eclipse IDE interface with a Java project named "datatypesprograms". The code in the main class "Ques09" finds the length of a string "surya" and prints it to the console. The output shows the length is 5.

```
1 package datatypesprograms;
2
3 public class Ques09 {
4
5     public static void main(String[] args) {
6         // program to find the length of the string
7         String s="surya";
8         int len=s.length();
9         System.out.println("the length of the string is"+len);
10    }
11
12 }
13
14 }
```

```
Console
terminated: Ques09 (2) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt\openjdt\hotspot\re\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\java.exe (Feb 28, 2025, 12:45:43 AM - 12:45:43 AM) [pid: 11604]
the length of the string is 5
```

## 10. Write a program to concatenate two strings.

The screenshot shows the Eclipse IDE interface with a Java project named "datatypesprograms". The code in the main class "Ques10" uses a StringBuilder to concatenate the strings "jaya" and "surya" and prints the result to the console. The output shows "jayasurya".

```
1 package datatypesprograms;
2
3 public class Ques10 {
4
5     public static void main(String[] args) {
6         // concatenate two string
7         StringBuilder s = new StringBuilder("jaya");
8         s.append("surya");
9         System.out.println(s);
10    }
11
12 }
13
14 }
```

```
Console
terminated: Ques10 (3) [Java Application] C:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt\openjdt\hotspot\re\full\win32\x86_64_17.0.5.v20221102-0933\jre\bin\java.exe (Feb 28, 2025, 1:01:08 AM - 1:01:08 AM) [pid: 16612]
jayasurya
```

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

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - CORIANDERPROGRAMS375/src/datatypesprograms/Ques11.java - Eclipse IDE". The Project Explorer shows multiple Java files. The code in Ques11.java is:

```
1 package datatypesprograms;
2
3 public class Ques11 {
4
5     public static void main(String[] args) {
6         // demonstrate the wrapper classes in java
7         System.out.println(Byte.MAX_VALUE+Byte.MIN_VALUE);
8         System.out.println(Short.MAX_VALUE+Short.MIN_VALUE);
9         System.out.println(Integer.MAX_VALUE +integer.MIN_VALUE);
10        System.out.println(Long.MAX_VALUE+Long.MIN_VALUE);
11        System.out.println(Character.MAX_VALUE+Character.MIN_VALUE);
12        System.out.println(Float.MAX_VALUE+Float.MIN_VALUE);
13        System.out.println(Double.MAX_VALUE+Double.MIN_VALUE);
14    }
15
16 }
17
```

The Console output shows the results of the println statements:

```
-1
-1
65535
3.4028235E38
1.7976931348623157E308
```

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

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - COREJAVA/PROGRAMS375/src/datatypesprograms/Ques12.java - Eclipse IDE". The Project Explorer shows multiple Java files. The code in Ques12.java is:

```
1 package datatypesprograms;
2
3 public class Ques12 {
4
5     public static void main(String[] args) {
6         // convert string to integer
7         String s1="100";
8         int i1=Integer.parseInt(s1);
9         System.out.println("THE STRING TO INTEGER IS"+ " "+s1);
10        int i1=101;
11        System.out.println("the integer to string is"+ " "+Integer.toString(i1));
12
13    }
14
15 }
16
```

The Console output shows the results of the println statements:

```
THE STRING TO INTEGER IS 100
the integer to string is 101
```

## **13. Write a program to demonstrate the use of string methods such as substring, index Of, and equals**

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - COREJAVA/PROGRAMS375/src/datatypesprograms/Ques13.java - Eclipse IDE". The Project Explorer shows multiple Java files. The code in Ques13.java is:

```
1 package datatypesprograms;
2
3 public class Ques13 {
4
5     public static void main(String[] args) {
6         // demonstrate String built-in methods
7         String s="jayasurya";
8         System.out.println(s.substring(1,5));
9         System.out.println(s.charAt(4));
10        System.out.println(s.startsWith("ja"));
11        System.out.println(s.endsWith("a"));
12        System.out.println(s.indexOf(6));
13        System.out.println(s.contains("hi"));
14        System.out.println(s.length());
15    }
16 }
```

The Console output shows the results of the println statements:

```
jayas
false
false
true
11
true
11
```

## **14. Write a program to count the occurrences of a character in a string**

The screenshot shows the Eclipse IDE interface with a Java project named "Ques14". The code in the editor is as follows:

```
1 package datatypesprograms;
2 import java.util.Scanner;
3 public class Ques14 {
4
5     public static void main(String[] args) {
6         // count of occurrences of the character
7         Scanner scan = new Scanner(System.in);
8         String s="jayasurya";
9         System.out.println("enter any character");
10        char c=scan.next().charAt(0);
11        int count=0;
12        System.out.println("the number of the occurrences of the character is");
13        for(int i=0;i<s.length();i++) {
14            if(s.charAt(i)==c) {
15                count++;
16            }
17        }
18    }
19    System.out.print(count);
20 }
21 }
```

The console output shows the user entering 'a' and the program outputting the count of 'a', which is 3.

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

The screenshot shows the Eclipse IDE interface with a Java project named "Ques15". The code in the editor is as follows:

```
1 package datatypesprograms;
2
3 public class Ques15 {
4
5     public static void main(String[] args) {
6         // check whether given string is palindrome or not
7         String s="101";
8         int temp=Integer.parseInt(s);
9         int result=0;
10        while(s!=0) {
11            result=result*10+temp%10;
12            s/=10;
13        }
14        System.out.println(result);
15        if(temp==result)
16            System.out.println("IT IS A PALINDROME");
17        else
18            System.out.println("IT IS NOT A PALINDROME");
19    }
20 }
```

The console output shows the user entering '101' and the program outputting "IT IS A PALINDROME".

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

The screenshot shows the Eclipse IDE interface with a Java project named "Ques16". The code in the editor is as follows:

```
1 package datatypesprograms;
2
3 public class Ques16 {
4
5     public static void main(String[] args) {
6         // reverse a string without using any built in methods
7         String str="surya";
8         String res="";
9         for(int i=str.length()-1;i>=0;i--) {
10            res=res+str.charAt(i);
11        }
12        System.out.println(res);
13    }
14
15 }
16
```

The console output shows the user entering 'surya' and the program outputting "ayrus".

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

The screenshot shows the Eclipse IDE interface. The Project Explorer view contains two files: Ques16.java and Ques17.java. The code for Ques17.java is as follows:

```
1 package datatypeprograms;
2
3 public class Ques17 {
4
5     public static void main(String[] args) {
6         // convert String upper to lower case and vice versa
7         String s="jayasurya";
8         System.out.println(s.toUpperCase());
9         String s1="JAYASURYA";
10        System.out.println(s1.toLowerCase());
11    }
12}
13
14
15}
16
```

The Console view shows the output of the program:

```
JAYASURYA
jayasurya
```

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

The screenshot shows the Eclipse IDE interface. The Project Explorer view contains two files: Ques18.java and Ques19.java. The code for Ques18.java is as follows:

```
1 package datatypeprograms;
2
3 public class Ques18 {
4
5     public static void main(String[] args) {
6         // program to demonstrate string builder class
7         Stringbuilder str=new StringBuilder("JAYASURYA");
8         System.out.println(str.length());
9         System.out.println(str.charAt(4));
10    }
11}
12
13
14}
15
```

The Console view shows the output of the program:

```
JAYASURYA
4
```

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

The screenshot shows the Eclipse IDE interface. The Project Explorer view contains three files: Ques19.java, Ques20.java, and Ques21.java. The code for Ques19.java is as follows:

```
1 package datatypeprograms;
2
3 import java.math.BigInteger;
4 import java.util.Scanner;
5
6 public class Ques19 {
7
8     public static void main(String[] args) {
9         // program to write a factorial of number using Big integer
10        Scanner scan = new Scanner(System.in);
11        System.out.println("Enter any number");
12        int a=scan.nextInt();
13        BigInteger b=new BigInteger(Integer.toString(a));
14        fact=1;
15        for(int i=1;i<=a;i++) {
16            fact=fact*i;
17        }
18        System.out.println(fact);
19    }
20}
```

The Console view shows the output of the program:

```
Enter any number
15
1004310016
```

## 20-Write a program to demonstrate the use of arrays of objects

```

1 package datatypeprograms;
2 public class Ques20 {
3     public static void main(String[] args) {
4         // demonstrate use of arrays objects
5         int a[] = {1,2,3,4,5};
6         String s[] = {"jaya","surya","mani","soma","varun","bharath","vinay"};
7         for(int i=0;i<a.length-1;i++) {
8             System.out.println(a[i]+" ");
9         }
10        for(String s1:s) {
11            System.out.println(s1);
12        }
13    }
14 }

```

Console <terminated> Ques20 (2) [Java Application] C:\eclipse\jee-2022-12\bin\java.exe -Dfile.encoding=UTF-8 -jar C:\eclipse\jee-2022-12\bin\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.5.v20221102-0935\run\bin\javaw.exe (Feb 28, 2025, 7:32:56 PM - 7:32:57 PM) [pid: 26588]

```

1
2
3
4
5 jaya
6 surya
7 mani
8 soma
9 varun
10 bharath
11 vinay

```

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

```

1 package datatypeprograms;
2 import java.util.Arrays;
3 public class Ques21 {
4     public static void main(String[] args) {
5         // sort an array of integer in ascending and descending order
6         int [] a = {1,3,2,7,4,8,5};
7         Arrays.sort(a);
8         for(int i=0;i<a.length-1;i++) {
9             System.out.print(a[i]+" ");
10        }
11        System.out.println();
12        for(int i=a.length-1;i>0;i--) {
13            System.out.print(a[i]+" ");
14        }
15    }
16 }

```

Console <terminated> Ques21 (2) [Java Application] C:\eclipse\jee-2022-12\bin\java.exe -Dfile.encoding=UTF-8 -jar C:\eclipse\jee-2022-12\bin\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.5.v20221102-0935\run\bin\javaw.exe (Feb 28, 2025, 7:41:15 PM - 7:41:16 PM) [pid: 17196]

```

1 2 3 4 5 7
8 7 5 4 3 2

```

## 22: Write a program to find the second largest and second smallest elements in an array

```

1 package datatypeprograms;
2 import java.util.Arrays;
3 public class Ques22 {
4     public static void main(String[] args) {
5         // program to find second largest and second smallest in an array
6         int [] a = {1,2,7,3,6,9,8};
7         Arrays.sort(a);
8         System.out.println("second largest is"+ " "+a[a.length-2]);
9         System.out.println("second smallest is"+ " "+a[1]);
10    }
11 }

```

Console <terminated> Ques22 (2) [Java Application] C:\eclipse\jee-2022-12\bin\java.exe -Dfile.encoding=UTF-8 -jar C:\eclipse\jee-2022-12\bin\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.5.v20221102-0935\run\bin\javaw.exe (Feb 28, 2025, 7:46:41 PM - 7:46:42 PM) [pid: 21344]

```

Second largest is 8
Second smallest is 2

```

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

The screenshot shows the Eclipse IDE interface with a Java code editor. The code is as follows:

```
5 // find the sum of diagonal elements in the matrix
6 int arr[] = new int[3][3];
7 Scanner scan = new Scanner(System.in);
8 System.out.println("ENTER ARRAY ELEMENTS");
9 for(int i=0;i<arr.length;i++) {
10     for(int j=0;j<arr[i].length;j++) {
11         arr[i][j]=scan.nextInt();
12     }
13 }
14 System.out.println("the array elements is sum is"+ " ");
15 int sum=0;
16 for(int i=0;i<arr.length;i++) {
17     for(int j=0;j<arr[i].length;j++) {
18         if(i==j)
19             sum+=arr[i][j];
20     }
21 }
22 System.out.println(sum);
23 }
24 }
```

The console output shows the user entering the matrix elements and the resulting sum:

```
ENTER ARRAY ELEMENTS
1 2 3
4 5 6
7 8 9
the array elements is sum is
21
```

## 24. Write a program to transpose a matrix.

The screenshot shows the Eclipse IDE interface with a Java code editor. The code is as follows:

```
3 public class Ques24 {
4     public static void main(String[] args) {
5         // transpose of a matrix
6         int[][] a =new int [3][3];
7         Scanner scan = new Scanner(System.in);
8         System.out.println("enter array elements");
9         for(int i=0;i<a.length;i++) {
10             for(int j=0;j<a.length;j++) {
11                 a[i][j]=scan.nextInt();
12             }
13         }
14         System.out.println("the array matrix is");
15         for(int i=0;i<a.length;i++) {
16             for(int j=0;j<a.length;j++) {
17                 System.out.print(a[i][j]+" ");
18             }
19             System.out.println();
20         }
21         System.out.println("the array transpose matrix is");
22         for(int i=0;i<a.length;i++) {
23             for(int j=0;j<a.length;j++) {
24                 System.out.print(a[j][i]+" ");
25             }
26             System.out.println();
27 }
```

The console output shows the original matrix and its transpose:

```
7 8 9
the array matrix is
1 2 3
4 5 6
7 8 9
the array transpose matrix is
1 4 7
2 5 8
3 6 9
```

## 25. Write a program to multiply two matrices.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS375/src/datatypesprograms/Ques25.java - Eclipse IDE
- Project Explorer:** Shows multiple Java files: Ques16.java, Ques7.java, Ques18.java, Ques19.java, Ques20.java, Ques21.java, Ques22.java, Ques23.java, Ques24.java, and \*Ques25.java.
- Code Editor:** Displays the Java code for Ques25.java, which performs matrix multiplication. The code uses Scanner for input and System.out.println for output.
- Console:** Shows the execution of the program. It prompts for matrices A and B, then prints the result of their multiplication.
- System Tray:** Shows the date and time as 2/28/2025, 11:28 PM.

## STRINGS:

### 1. Write a program to reverse a string.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS375/src/datatypesprograms/Ques01.java - Eclipse IDE
- Project Explorer:** Shows multiple Java files: Ques01.java, Ques02.java, Ques03.java, Ques04.java, Ques05.java, Ques06.java, Ques07.java, Ques08.java, Ques09.java, Ques10.java, Ques11.java, Ques12.java, Ques13.java, Ques14.java, Ques15.java, Ques16.java, Ques17.java, Ques18.java, Ques19.java, Ques20.java, Ques21.java, Ques22.java, Ques23.java, Ques24.java, and Ques25.java.
- Code Editor:** Displays the Java code for Ques01.java, which reverses a string using a StringBuilder.
- Console:** Shows the execution of the program. It prompts for a string and prints its reverse.
- System Tray:** Shows the date and time as Mar 1, 2022, 11:05 AM.

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

```

2 import java.util.Scanner;
3 public class Ques02 {
4     public static void main(String[] args) {
5         // string is palindrome or not
6         Scanner scan = new Scanner(System.in);
7         System.out.println("enter a string");
8         String str=scan.nextLine();
9
10        StringBuilder Str1=new StringBuilder(str);
11        Str1.reverse();
12        System.out.println(Str1);
13        if(Str1.toString().equals(str)){
14            System.out.println("it is a plindrome");
15        }
16        else {
17            System.out.println("it is not a paindrome");
18        }
19    }
20    scan.close();
21 }
22
Data Source Explorer  Snippets  Terminal  Console
terminated < Ques02 (3) [Java Application] C:\eclipse\jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.jdt\openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (Mar. 17, 2023)
enter a string
urya
yru
t is not a paindrome

```

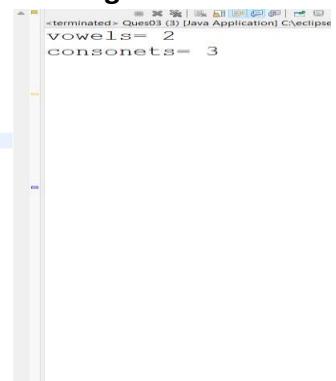
---

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

```

1 package stringsprograms;
2 public class Ques03 {
3     public static void main(String[] args) {
4         // count number of words and consonets in a string
5         String str="surya";
6         int len=str.length();
7         int vowels=0;
8         int consonets=0;
9         char ch;
10        for(int i=0;i<str.length();i++) {
11            ch=str.charAt(i);
12            if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u') {
13                vowels++;
14            }
15            else if(ch>='a'&&ch<='z') {
16                consonets++;
17            }
18        }
19        System.out.println("vowels="+ " "+vowles);
20        System.out.println("consonets="+" "+consonets);
21    }
22 }
23

```




---

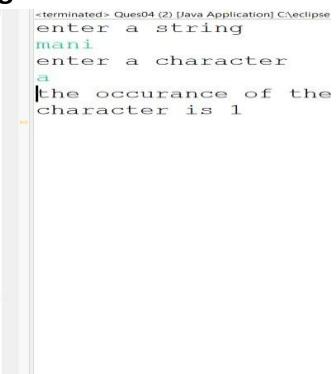
### 4. Write a program to count the occurrences of a character in a string.

```

import java.util.Scanner;

public class Ques04 {
    public static void main(String[] args) {
        // count of occurrences of the character
        Scanner scan =new Scanner(System.in);
        System.out.println("enter a string");
        String str=scan.nextLine();
        System.out.println("enter a character");
        char ch=scan.next().charAt(0);
        int count=0;
        for(int i=0;i<str.length();i++) {
            if(str.charAt(i)==ch) {
                count++;
            }
        }
        System.out.println("the occurrence of the character is"
    }
}

```




---

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

```

package stringsprograms;

import java.util.StringTokenizer;

public class Ques05 {

    public static void main(String[] args) {
        // remove all white spaces in string
        StringTokenizer s= new StringTokenizer("s u r y a", " ");
        while(s.hasMoreTokens()) {
            System.out.print(s.nextToken());
        }
    }
}

```

<terminated> Ques05 (3) [Java Application] C:\eclipse-jee-2022-06-SR1\Ques05\bin\surya

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

```

package stringsprograms;

public class Ques06 {

    public static void main(String[] args) {
        // length of the string
        String str="surya";
        int len=str.length();
        System.out.println("the length of the string is"+len);
    }
}

```

<terminated> Ques06 (3) [Java Application] C:\eclipse-jee-2022-06-SR1\Ques06\bin\the length of the string is 5

## 7. Write a program to concatenate two strings.

```

package stringsprograms;

public class Ques07 {

    public static void main(String[] args) {
        // program to concatenate 2 strings
        String s1="jaya";
        String s2="surya";
        String s3=s1.concat(s2);
        System.out.println("the concatenation of string is"+s3);
    }
}

```

<terminated> Ques07 (3) [Java Application] C:\eclipse-jee-2022-06-SR1\Ques07\bin\the concatenation of string is jayasurya

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA\PROGRAMS5375\src\stringsprograms\Ques08.java - Eclipse IDE
- Project Explorer:** Shows the project structure under "COREJAVA\PROGRAMS5375".
- Code Editor:** Displays the Java code for Ques08.java.
- Console:** Shows the output of the program execution.

```

1 package stringsprograms;
2
3 public class Ques08 {
4
5     public static void main(String[] args) {
6         // convert lowercase to uppercase
7         String s="surya";
8         System.out.println("the upper case is"+s.toUpperCase());
9         System.out.println("the lower case is"+s.toLowerCase());
10    }
11 }
12
13 }
14

```

Output in Console:

```

the upper case
isSURYA
the lower case
issurya

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA\PROGRAMS5375\src\stringsprograms\Ques09.java - Eclipse IDE
- Project Explorer:** Shows the project structure under "COREJAVA\PROGRAMS5375".
- Code Editor:** Displays the Java code for Ques09.java.
- Console:** Shows the output of the program execution.

```

1 package stringsprograms;
2
3 public class Ques09 {
4     private static String LongWord(String sentence) {
5         String[] word=sentence.split(" ");
6         String longestWord="";
7         for(int i=0;i<word.length;i++) {
8             if(word[i].length()>longestWord.length()) {
9                 longestWord=word[i];
10            }
11        }
12        return longestWord;
13    }
14    public static void main(String[] args) {
15        // find the longest word in the string
16        String sentence="global quest technologies";
17        System.out.println(LongWord(sentence));
18    }
19
20 }
21

```

Output in Console:

```

the upper case
isSURYA
the lower case
issurya

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA\PROGRAMS5375\src\stringsprograms\Ques10.java - Eclipse IDE
- Project Explorer:** Shows the project structure under "COREJAVA\PROGRAMS5375".
- Code Editor:** Displays the Java code for Ques10.java.
- Console:** Shows the output of the program execution.

```

1 package stringsprograms;
2
3 import java.util.Arrays;
4
5 public class Ques10 {
6     // check given string is anagram or not
7     public static boolean isAnagram(String str1, String str2) {
8         String str3=str1.replaceAll("\n\r", "");
9         String str4=str2.replaceAll("\n\r", "");
10        if(str3.length()!=str4.length()) {
11            return false;
12        }
13
14        else {
15            char[]ch1=str3.toLowerCase().toCharArray();
16            char[]ch2=str4.toLowerCase().toCharArray();
17            Arrays.sort(ch1);
18            Arrays.sort(ch2);
19
20            return Arrays.equals(ch1, ch2);
21        }
22
23
24    }
25
26    public static void main(String[] args) {
27        System.out.println(isAnagram("silent", "lisent"));
28    }
29

```

Output in Console:

```

true

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS275/src/stringsprograms/Ques11.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA PROGRAMS275" containing numerous Java files (Ques01.java through Ques24.java) under a package named "stringsprograms".
- Code Editor:** Displays the Java code for Ques11.java, which finds the first non-repeated character in a string.
- Console:** Shows the output "terminated> Ques11 (4) [Java Application] C:\eclipse\jee-2022-12\Ques11.java:20: D".
- System Tray:** Shows icons for battery, signal strength, and date/time (3/17/2025).

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS275/src/stringsprograms/Ques12.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA PROGRAMS275" containing numerous Java files (Ques01.java through Ques24.java) under a package named "stringsprograms".
- Code Editor:** Displays the Java code for Ques12.java, which checks if a string contains only digits.
- Console:** Shows the output "terminated> Ques12 (4) [Java Application] C:\eclipse\jee-2022-12\Ques12.java:20: enter string with numbers".
- System Tray:** Shows icons for battery, signal strength, and date/time (3/17/2025).

## 13. Write a program to find all permutations of a string.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS275/src/stringsprograms/Ques13.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA PROGRAMS275" containing numerous Java files (Ques01.java through Ques24.java) under a package named "stringsprograms".
- Code Editor:** Displays the Java code for Ques13.java, which prints all permutations of a string.
- Console:** Shows the output "terminated> Ques13 (4) [Java Application] C:\eclipse\jee-2022-12\Ques13.java:20: cd cd cd".
- System Tray:** Shows icons for battery, signal strength, and date/time (3/17/2025).

## 14. Write a program to find the frequency of each character in a string.

```

1 package stringsprograms;
2
3 import java.util.Scanner;
4
5 public class Ques14 {
6     public static void main(String[] args) {
7         // find the frequency of the each character in String
8         Scanner scan = new Scanner(System.in);
9         System.out.println("Enter a String");
10        String str=scan.nextLine();
11        int[] freq = new int[256];
12        for(char ch : str.toCharArray()) {
13            freq[ch]++;
14        }
15        System.out.println("character frequencies is");
16        for(int i=0; i<256; i++) {
17            if(freq[i]>0) {
18                System.out.println(i + " " + freq[i]);
19            }
20        }
21    }
22 }

```

## **15. Write a program to reverse words in a sentence.**

```

1 package stringsprograms;
2
3 public class Ques15 {
4
5     public static void main(String[] args) {
6         // write a program to reverse of words in a sentence
7         String str="Type your name";
8         String[] wordsname=str.split(" ");
9         for(int i=wordsname.length-1; i>-0; i--) {
10             System.out.print(wordsname[i]);
11         }
12     }
13 }

```

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

```

1 package stringsprograms;
2
3 public class Ques16 {
4
5     public static void main(String[] args) {
6         // find the duplicate characters in string
7         String str="jayasurya";
8         int[] freq=new int[256];
9         for(char ch:str.toCharArray()) {
10             freq[ch]++;
11         }
12         System.out.println("duplicates are");
13         for(int i=0; i<256; i++) {
14             if(freq[i]>1) {
15                 System.out.println((char)i + " " + freq[i]);
16             }
17         }
18     }
19 }

```

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

**Project Explorer:**

- File
- Edit
- Source
- Refactor
- Navigate
- Search
- Project
- Run
- Window
- Help

**Code Editor (Quest17.java):**

```

1 package stringsprograms;
2 import java.util.*;
3 public class Quest17 {
4
5     public static char findFirstRepeatingChar(String s) {
6         HashSet<Character> seen = new HashSet<>();
7         for (char ch : s.toCharArray()) {
8             if (seen.contains(ch)) {
9                 return ch; // Return the first repeating character
10            }
11            seen.add(ch);
12        }
13        return '\0'; // Return null character if no repeats
14    }
15
16    public static void main(String[] args) {
17        Scanner scan = new Scanner(System.in);
18        System.out.print("Enter a string: ");
19        String sl = scan.nextLine();
20        scan.close();
21
22        char result = findFirstRepeatingChar(sl);
23        if (result != '\0') {
24            System.out.println("First repeating character is " + result);
25        } else {
26            System.out.println("No repeating character found.");
27        }
28    }
}

```

**Console Output:**

```

terminated: Quest17 (Java Application) [Eclipse IDE - 2023-07-17T10:45:00Z]
Enter a string: mani
No repeating characters found.

```

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

**Project Explorer:**

- File
- Edit
- Source
- Refactor
- Navigate
- Search
- Project
- Run
- Window
- Help

**Code Editor (Quest18.java):**

```

1 package stringsprograms;
2 import java.util.Scanner;
3
4 public class Quest18 {
5     public static String capitalize(String Str) {
6         String[] words=Str.split(" ");
7         String res="";
8         for(String word:words) {
9             if(word.isEmpty())
10                 res+=Character.toUpperCase(word.charAt(0))+word.substring(1);
11             else
12                 res+=word.charAt(0)+word.substring(1);
13         }
14         return res.trim();
15     }
16
17     public static void main(String[] args) {
18         // Capitalize the first letter in each word in sentence
19         Scanner scan = new Scanner(System.in);
20         System.out.print("Enter a sentence: ");
21         String Sen=scan.nextLine();
22         System.out.println(capitalize(Sen));
23     }
24 }

```

**Console Output:**

```

terminated: Quest18 (Java Application) [Eclipse IDE - 2023-07-17T10:45:00Z]
enter a sentence
java surya
Java Surya

```

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

**Project Explorer:**

- File
- Edit
- Source
- Refactor
- Navigate
- Search
- Project
- Run
- Window
- Help

**Code Editor (Quest19.java):**

```

1 package stringsprograms;
2
3 import java.util.Scanner;
4
5 public class Quest19 {
6     public static void isRotation(String str1, String str2) {
7         if(str1.length()!=str2.length()) {
8             System.out.println("String is not rotation of another string");
9         } else {
10             String s=str1+str1;
11             if(s.contains(str2)) {
12                 System.out.println("String is rotation of another string");
13             } else {
14                 System.out.println("String is not rotation of another string");
15             }
16         }
17     }
18
19     public static void main(String[] args) {
20         // Check whether a string is a rotation
21         Scanner scan=new Scanner(System.in);
22         System.out.print("Enter a str1");
23         String str1=scan.nextLine();
24         System.out.print("Enter a str2");
25         String str2=scan.nextLine();
26         isRotation(str1,str2);
27     }
28 }

```

**Console Output:**

```

terminated: Quest19 (Java Application) [Eclipse IDE - 2023-07-17T10:45:00Z]
enter a str1
jaya
enter a str2
surya
string is not rotation of another string

```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA" containing several packages like "Assignment1", "Assignment2", and "COREJAVA PROGRAMS375".
- Code Editor:** Displays the Java code for "Ques20.java".
- Console:** Shows the output of the program: "string is a substring of another string".

```
package stringsprograms;
public class Ques20 {
    public static void main(String[] args) {
        // check whether a string is a substring of another string
        String str1="jayasurya";
        String str2="surya";
        if(str1.contains(str2)) {
            System.out.println("string is a substring of another string");
        } else {
            System.out.println("it contains other character also");
        }
    }
}
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA" containing several packages like "Assignment1", "Assignment2", and "COREJAVA PROGRAMS375".
- Code Editor:** Displays the Java code for "Ques21.java".
- Console:** Shows the output of the program: "array[1]surya".

```
import java.util.Arrays;
public class Ques21 {
    public static void main(String[] args) {
        // remove the duplicates from the string
        String str="jayasurya";
        char[]str1 = str.toCharArray();
        Arrays.sort(str1);
        int j=1;
        for(int i=0;i<str1.length-1;i++) {
            if(str1[i]==str1[i+1]) {
                str1[i]=str1[i+1];
                j++;
            }
        }
        for(int i=0;i<j;i++) {
            System.out.print(str1[i]);
        }
    }
}
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA" containing several packages like "Assignment1", "Assignment2", and "COREJAVA PROGRAMS375".
- Code Editor:** Displays the Java code for "Ques22.java".
- Console:** Shows the output of the program: "array[1]surya".

```
package stringsprograms;
public class Ques22 {
    public static void main(String[] args) {
        // find the common character in the two strings
        String str="surya";
        String str1="surya";
        String res="";
        for(int i=0;i<str.length();i++) {
            for(int j=0;j<str1.length();j++) {
                if(str.charAt(i)==str1.charAt(j)) {
                    if(!res.contains(str.charAt(i)+"")) {
                        res+=str.charAt(i);
                    }
                }
            }
        }
        System.out.println(res);
    }
}
```

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

```

1 package stringsprograms;
2
3 public class Ques23 {
4
5     public static void main(String[] args) {
6         // check whether a given string is pangram or not
7         String s1="The quick brown fox jumps over the lazy dog";
8         String s2=s1.toLowerCase();
9         for(char ch='a';ch<='z';ch++) {
10             if(!s2.contains(ch+"")) {
11                 System.out.println("it is not a pangram");
12                 return;
13             }
14         }
15         System.out.println("it is a pangram");
16     }
17 }
18
19
20
21

```

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

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

```

1 package stringsprograms;
2
3 import java.util.Arrays;
4
5 public class Ques25 {
6
7     public static void main(String[] args) {
8         // find the
9         String[] s = {"Mani", "Mal", "Mango", "Manish"};
10        Arrays.parallelSort(s, (a,b)->Integer.compare(a.length,
11        for(int i=0; i<s[0].length(); i++) {
12            for(int j=1; j<s.length+1; j++) {
13                if( s[0].length()==j ) {
14                    System.out.println(s[0].substring(0, j));
15                    return;
16                }
17                else if(s[0].charAt(i) != s[j].charAt(i)) {
18                    System.out.println(s[0].substring(0, i));
19                    return;
20                }
21            }
22        }
23    }
24
25 }
26

```

## **OVERLOADING:**

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS575" with several source files listed under "src/overloadingprograms".
- Code Editor:** Displays the Java code for "Ques01.java".
- Console:** Shows the output of the program execution: "sum=30".

```
1 package overloadingprograms;
2 //write a program to find the sum of the two integers(method o
3
4 class sum{
5     public void Sum(int a,int b) {
6         int c=a+b;
7         System.out.println("sum-"+c);
8     }
9     public static void Sum(int a,int b,int c) {
10        int d=a+b+c;
11        System.out.println("sum-"+d);
12    }
13 }
14
15 public class Ques01 {
16     public static void main(String[] args) {
17         sum s=new sum();
18         s.Sum(10, 20);
19     }
20 }
21
22 }
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS575" with several source files listed under "src/overloadingprograms".
- Code Editor:** Displays the Java code for "Ques02.java".
- Console:** Shows the output of the program execution: "sum=-31.1".

```
1 package overloadingprograms;
2 //write a program to find the sum of the two integers(method o
3
4 class Add{
5     public void Sum(float a,float b) {
6         float c=a+b;
7         System.out.println("sum-"+c);
8     }
9     public static void Sum(float a,float b,float c) {
10        float d=a+b+c;
11        System.out.println("sum-"+d);
12    }
13 }
14
15 public class Ques02 {
16     public static void main(String[] args) {
17         Add a=new Add();
18         a.Sum(10.5f, 20.6f);
19     }
20 }
21
22 }
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS575" with several source files listed under "src/overloadingprograms".
- Code Editor:** Displays the Java code for "Ques03.java".
- Console:** Shows the output of the program execution: "150.0".

```
1 package overloadingprograms;
2 //find the area of the rectangle (using method overloading)
3
4 class Area{
5     public void AreaOfRectangle(int length , int height) {
6         double res=length*height;
7         System.out.println(res);
8     }
9     public void AreaOfRectangle(float length , float height) {
10        double res=length*height;
11        System.out.println(res);
12    }
13 }
14
15 public class Ques03 {
16
17     public static void main(String[] args) {
18         Area a = new Area();
19         a.AreaOfRectangle(10, 15);
20     }
21
22 }
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVA PROGRAMS".
- Code Editor:** Displays the following Java code in the file "Ques04.java":
 

```

1 package overloadingprograms;
2 //Find the volume of cylinder(method overloading)
3 class Volume{
4     public void volumeOfCylinder(int radius,int height) {
5         double res=(3.14)*radius*radius*height;
6         System.out.println(res);
7     }
8     public void volumeOfCylinder(float radius,int height) {
9         double res=(3.14)*radius*radius*height;
10        System.out.println(res);
11    }
12 }
13
14 public class Ques04 {
15
16     public static void main(String[] args) {
17         // TODO Auto-generated method stub
18         Volume v=new Volume();
19         v.volumeOfCylinder(5.5f,5);
20     }
21 }
22
23
24
      
```
- Console:** Shows the output "474.925".
- System Bar:** Shows the date and time as "8/17/2025 11:05 PM".

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVA PROGRAMS".
- Code Editor:** Displays the following Java code in the file "Ques05.java":
 

```

1 package overloadingprograms;
2 //find the factorial of the number(method overloading)
3 class Fact {
4     public int isFact(int a) {
5         int fact=1;
6         for(int i=1;i<=a;i++) {
7             fact=fact*i;
8         }
9         System.out.println(fact);
10    }
11 }
12
13 public class Ques05 {
14
15     public static void main(String[] args) {
16         Fact f = new Fact();
17         f.isFact(5);
18     }
19 }
20
      
```
- Console:** Shows the output "120".
- System Bar:** Shows the date and time as "8/17/2025 11:05 PM".

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVA PROGRAMS".
- Code Editor:** Displays the following Java code in the file "Ques06.java":
 

```

1 package overloadingprograms;
2 //find the average of the three number(method overloading)
3 class Avgl {
4     public void avgOfNumbers(int a,int b,int c) {
5         int res=a+b+c/3;
6         System.out.println(res);
7     }
8     public void avgOfNumbers(float a,float b,float c) {
9         float res=a+b+c/3;
10        System.out.println(res);
11    }
12 }
13
14 public class Ques06 {
15
16     public static void main(String[] args) {
17         Avgl a = new Avgl();
18         a.avgOfNumbers(5.1f, 5.6f, 8.9f);
19     }
20 }
21
22
23
24
      
```
- Console:** Shows the output "13.66666".
- System Bar:** Shows the date and time as "8/17/2025 11:05 PM".

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS375' open. The 'Project Explorer' view on the left lists various Java files. The 'Ques07.java' file is selected and displayed in the central editor area. The code implements a class 'Maximum' with methods to find the maximum of two integers and two floats, and a main method to demonstrate it.

```
package overloadingprograms;
//Find the maximum of the two numbers
class Maximum{
    public void maximumOfTwoNumbers(int a,int b) {
        int res=Math.max(a, b);
        System.out.println(res);
    }
    public void maximumOfTwoNumbers(int a,float b) {
        float res=Math.max(a, b);
        System.out.println(res);
    }
}
public class Ques07 {
    public static void main(String[] args) {
        Maximum m=new Maximum();
        m.maximumOfTwoNumbers(9, 13.6f);
    }
}
```

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS375' open. The 'Project Explorer' view on the left lists various Java files. The 'Ques08.java' file is selected and displayed in the central editor area. The code implements a class 'Min' with methods to find the minimum of two integers and two floats, and a main method to demonstrate it.

```
package overloadingprograms;
// find the minimum of the two numbers
class Min{
    public void minOfNumbers(int a,int b) {
        int res=Math.min(a, b);
        System.out.println(res);
    }
    public void minOfNumbers(int a,Float b) {
        float res=Math.min(a, b);
        System.out.println(res);
    }
}
public class Ques08 {
    public static void main(String[] args) {
        Min m= new Min();
        m.minOfNumbers(5, 7.7f);
    }
}
```

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS375' open. The 'Project Explorer' view on the left lists various Java files. The 'Ques09.java' file is selected and displayed in the central editor area. The code implements a class 'Power' with methods to calculate the power of a number (base, power) using Math.pow() and to calculate the power of a double (base, power), and a main method to demonstrate it.

```
package overloadingprograms;
//find the power of number
class Power{
    public void powerOfNumber(int base,double power) {
        double res=Math.pow(base, power);
        System.out.println(res);
    }
    public void powerOfNumber(double base,double power) {
        double res=Math.pow(base, power);
        System.out.println(res);
    }
}
public class Ques09 {
    public static void main(String[] args) {
        Power p= new Power();
        p.powerOfNumber(5, 5);
    }
}
```

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

The screenshot shows the Eclipse IDE interface with a Java project named "CORLLJAVA". The code editor displays a file named "Ques10.java" containing the following Java code:

```

package overloadingprograms;
class Prime{
    public void isPrime(int num) {
        int count=0;
        for(int i=1;i<=num;i++) {
            if(num%i==0) {
                count++;
            }
        }
        if(count==2) {
            System.out.println("it is a prime number");
        } else {
            System.out.println("it is not a prime number");
        }
    }
}
public class Ques10 {
    public static void main(String[] args) {
        Prime p= new Prime();
        p.isPrime(5);
    }
}

```

The Java console output shows the result: "it is a prime number".

## 11. Write a program

The screenshot shows the Eclipse IDE interface with a Java project named "CORLLJAVA". The code editor displays a file named "Ques11.java" containing the following Java code:

```

package overloadingprograms;
class Square{
    // find the square root of the number
    public void squareRootIs(int a) {
        double res=Math.sqrt(a);
        System.out.println(res);
    }
    public void squareRootIs(double a) {
        double res=Math.sqrt(a);
        System.out.println(res);
    }
}
public class Ques11 {
    public static void main(String[] args) {
        Square s= new Square();
        s.squareRootIs(4);
    }
}

```

The Java console output shows the result: "2.0".

to find the square root of a number.

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

The screenshot shows the Eclipse IDE interface with a Java project named "CORLLJAVA". The code editor displays a file named "Ques12.java" containing the following Java code:

```

package overloadingprograms;
class peri{
    //cal perimeter of rectangle
    public void perimeterOfRectangle(int l,int b) {
        int res=2*(l+b);
        System.out.println(res);
    }
    public void perimeterOfRectangle(int l,float b) {
        float res=2*(l+b);
        System.out.println(res);
    }
}
public class Ques12 {
    public static void main(String[] args) {
        peri p=new peri();
        p.perimeterOfRe
    }
}

```

A tooltip is visible in the code editor area, showing the method signature: `void overloadingprograms.Ques12.main(String[] args)`.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS5375" containing several source files: Ques01.java through Ques35.java.
- Code Editor:** Displays Java code for calculating the area of a circle. The code defines a class Circle with two methods: areaOfCircle(float r) and areaOfCircle(int r). It also contains a main method that creates a Circle object and calls its areaOfCircle method with the argument 5.
- Console:** Shows the output of the program: 78.5

```
1 package overloadingprograms;
2 // find the area of the circle
3 class Circle{
4     public void areaOfCircle(float r) {
5         double res=3.14*r*r;
6         System.out.println(res);
7     }
8     public void areaOfCircle(int r) {
9         double res=3.14*r*r;
10        System.out.println(res);
11    }
12 }
13 public class Ques13 {
14
15     public static void main(String[] args) {
16         Circle c= new Circle();
17         c.areaOfCircle(5);
18     }
19 }
20 }
21 
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS5375" containing several source files: Ques01.java through Ques35.java.
- Code Editor:** Displays Java code for converting Celsius to Fahrenheit. The code defines a class temp with a method celsiusToFahrenheit that takes an int parameter celsius and prints the result. It also contains a main method that creates a temp object and calls its celsiusToFahrenheit method with the argument 50.
- Console:** Shows the output of the program: 82.0

```
1 package overloadingprograms;
2 //convert temprature from celsius to Fahrenheat
3 class temp{
4     public void celsiusToFahrenheit(int celsius) {
5         double fahernheat=(celsius*(9/5))+32;
6         System.out.println(fahernheat);
7     }
8 }
9
10 public class Ques15 {
11
12
13     public static void main(String[] args) {
14         temp t=new temp();
15         t.celsiusToFahrenheit(50);
16     }
17 }
18
19
20 
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS5375" containing several source files: Ques01.java through Ques35.java.
- Code Editor:** Displays Java code for calculating the area of a triangle. The code defines a class Triangle with a method areaATriangle that takes two int parameters b and h, calculates the area as (0.5)\*b\*h, and prints the result. It also contains a main method that creates a Triangle object and calls its areaATriangle method with the arguments 2 and 5.
- Console:** Shows the output of the program: 5.0

```
1 package overloadingprograms;
2 //area a triangle
3 class Triangle{
4     public void areaATriangle(int b,int h) {
5         double res=(0.5)*b*h;
6         System.out.println(res);
7     }
8 }
9
10 public class Ques14 {
11
12     public static void main(String[] args) {
13         Triangle t= new Triangle();
14         t.areaATriangle(2, 5);
15     }
16 }
17
18
19 
```

#### 16. Write a program to convert temperature from Fahrenheit to Celsius.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS275/src/overloadingprograms/Ques16.java - Eclipse IDE
- Project Explorer:** Shows a large list of Java files under the package `COREJAVA/PROGRAMS275`, including `Ques01.java` through `Ques25.java`.
- Code Editor:** Displays the Java code for `Ques16.java`. The code defines a class `Temp1` with a method `fahernToCelsius` that converts Fahrenheit to Celsius using the formula  $(\text{fahern} - 32) * 5/9$ . It also contains a main method that creates an instance of `Temp1` and calls its `fahernToCelsius` method with the value 5.
- Console:** Shows the output of the program: `15.0`.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS275/src/overloadingprograms/Ques17.java - Eclipse IDE
- Project Explorer:** Shows a large list of Java files under the package `COREJAVA/PROGRAMS275`, including `Ques01.java` through `Ques25.java`.
- Code Editor:** Displays the Java code for `Ques17.java`. The code defines a class `Sphere` with a method `volumeOfSphere` that calculates the volume of a sphere using the formula  $\frac{4}{3}\pi r^3$ . It also contains a main method that creates an instance of `Sphere` and calls its `volumeOfSphere` method with the value 5.
- Console:** Shows the output of the program: `392.5`.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS275/src/overloadingprograms/Ques18.java - Eclipse IDE
- Project Explorer:** Shows a large list of Java files under the package `COREJAVA/PROGRAMS275`, including `Ques01.java` through `Ques25.java`.
- Code Editor:** Displays the Java code for `Ques18.java`. The code defines a class `array` with a method `avgOfArray` that calculates the average of an integer array. It also contains a main method that creates an array and calls its `avgOfArray` method with the value `{1, 2, 3, 4, 5}`.
- Console:** Shows the output of the program: `3`.

### 19. Write a program to calculate the compound interest.

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

```

package overloadingprograms;
// calculate the compound interest
class Interest{
    public void compoundInterest(int p,double r,int years) {
        double res = p * Math.pow((1 + r / 100.0), years);
        System.out.println(res);
    }
}
public class Ques19 {
    public static void main(String[] args) {
        Interest i=new Interest();
        i.compoundInterest(1000, 5, 2);
    }
}

```

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

```

package overloadingprograms;
import java.util.*;
// find the area of trapezoid
class Trapezoid {
    public void calculateArea(double basel, double base2,
        double area = 0.5 * (basel + base2) * height;
        System.out.println("Area of the trapezoid: " + area);
    }
}
public class Ques20 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Base 1: ");
        double basel = scanner.nextDouble();
        System.out.print("Enter Base 2: ");
        double base2 = scanner.nextDouble();
        System.out.print("Enter Height: ");
        double height = scanner.nextDouble();
        Trapezoid t = new Trapezoid();
        t.calculateArea(basel, base2, height);
        scanner.close();
    }
}

```

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

```

package overloadingprograms;
// area of parallelogram
import java.util.Scanner;
class Parallelogram {
    public void calculateArea(double base, double height) {
        double area = base * height;
        System.out.println("Area of the parallelogram: " + area);
    }
}
public class Ques21 {
    public static void main(String[] args) {
        Parallelogram p = new Parallelogram();
        p.calculateArea(10,20);
    }
}

```

## 22. Write a program to calculate the simple interest.

```

package overloadingprograms;
class Unrest {
    public void calculateArea(float p, float t, float x) {
        double area = p*t*x;
        System.out.println(area);
    }
    public void calculateArea(int p, int t, float x) {
        double area = p*t*x;
        System.out.println(area);
    }
}
public class Ques22 {
    public static void main(String[] args) {
        Unrest t = new Unrest();
        t.calculateArea(100, 20, 350);
    }
}

```

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

```

package overloadingprograms2;
class Area {
    public void calculateArea(double side) {
        double area = side * side;
        System.out.println("Area of the rhombus: " + area);
    }
    public void calculateArea(int side) {
        double area = side*side;
        System.out.println("Area of the rhombus : " + area);
    }
}
public class Ques23 {
    public static void main(String[] args) {
        Area a = new Area();
        a.calculateArea(20);
    }
}

```

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

```

package overloadingprograms;
class Rhombus {
    public void calculateArea(double diagonal1, double diagonal2) {
        double area = 0.5 * diagonal1 * diagonal2;
        System.out.println("Area of the rhombus (using diagonals): " + area);
    }
    public void calculateArea(double base, double height, boolean b) {
        double area = base * height;
        System.out.println("Area of the rhombus (using base & height): " + area);
    }
}
public class Ques24 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Rhombus r = new Rhombus();
        System.out.print("Enter Diagonal 1: ");
        double d1 = scanner.nextDouble();
        System.out.print("Enter Diagonal 2: ");
        double d2 = scanner.nextDouble();
        r.calculateArea(d1, d2);
    }
}

```

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

```

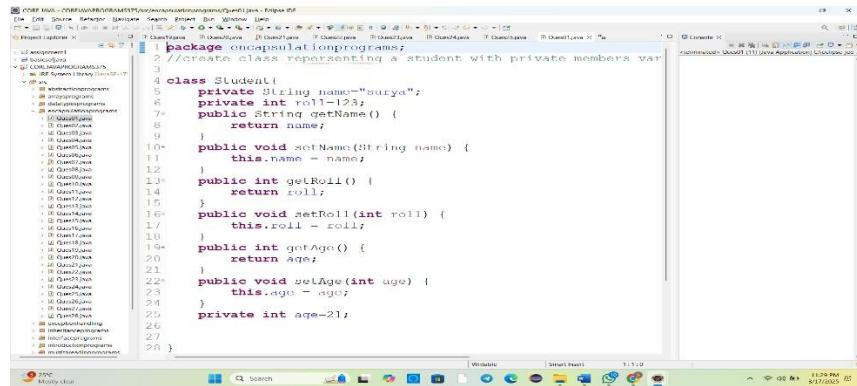
package overloadingprograms;
class polygon {
    public void areaOfPolygon(int n, int l, int a) {
        double rem = l*a*n;
        System.out.println(rem);
    }
    public void areaOfPolygon(int n, float l, int a) {
        double rem=n/2*l*a;
        System.out.println(rem);
    }
}
public class Ques25 {
    public static void main(String[] args) {
        polygon p = new polygon();
        p.areaOfPolygon(6, 5, 6);
    }
}

```

**ENCAPSULATION:**

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

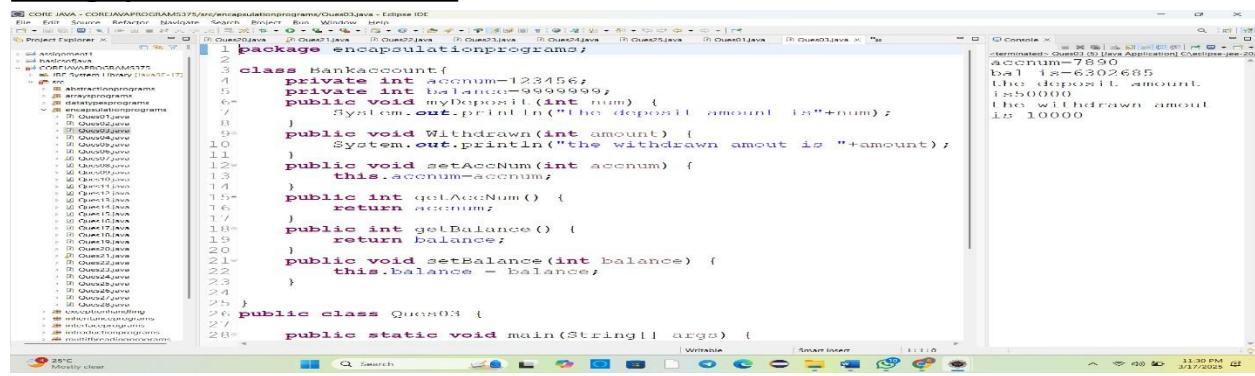
**2. Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods**



```
package encapsulationprograms;
//create class representing a student with private members var
class Student {
    private String name="surya";
    private int roll=123;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getRoll() {
        return roll;
    }
    public void setRoll(int roll) {
        this.roll = roll;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    private int age=21;
}
```

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

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



```
package encapsulationprograms;
class Bankaccount {
    private int accnum=123456;
    private int balance=6302.685;
    public void Deposit(int num) {
        System.out.println("The deposit amount is "+num);
    }
    public void Withdrawn(int amount) {
        System.out.println("the withdrawn amount is "+amount);
    }
    public void setAccNum(int accnum) {
        this.accnum=accnum;
    }
    public int getAccNum() {
        return accnum;
    }
    public int getBalance() {
        return balance;
    }
    public void setBalance(int balance) {
        this.balance = balance;
    }
}
public class Ques03 {
    public static void main(String[] args) {
    }
}
```

Terminal output:

```
terminal: Ques03 [3] User Application C:\eclipse\jee-2020-06\bin\Ques03.java
bal 123456
the deposit amount
150000
the withdrawn amount
is 10000
```

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

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

```

package encapsulationprograms;
class car{
    private String model="kia sonet";
    private String color="red";
    private int price=12345;
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}
public class Ques05 {
}

```

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

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

```

package encapsulationprograms;
class book{
    private String title="java";
    private String author="james";
    private int price=2000;
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
}
public class Ques07 {
    public static void main(String[] args) {
        book b= new book();
        b.setAuthor("gosling");
        b.setTitle("corejava");
        System.out.println(b.getAuthor());
        System.out.println(b.getTitle());
    }
}

```

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

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

```

1 package encapsulationprograms;
2 class CollegeStudent{
3     private String name="surya";
4     private int roll=123;
5     private int age=21;
6     public String getName() {
7         return name;
8     }
9     public void setName(String name) {
10        this.name = name;
11    }
12    public int getRoll() {
13        return roll;
14    }
15    public void setRoll(int roll) {
16        this.roll = roll;
17    }
18    public int getAge() {
19        return age;
20    }
21    public void setAge(int age) {
22        this.age = age;
23    }
24
25 }
26
27 public class Ques09 {
28

```

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

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

```

1 package encapsulationprograms;
2 class Circle{
3     private float radius=3.14f;
4     private int area=3000;
5     private int circum =35;
6     public float getRadius() {
7         return radius;
8     }
9     public void setRadius(float radius) {
10        this.radius = radius;
11    }
12    public int getArea() {
13        return area;
14    }
15    public void setArea(int area) {
16        this.area = area;
17    }
18    public int getCircum() {
19        return circum;
20    }
21    public void setCircum(int circum) {
22        this.circum = circum;
23    }
24
25 }
26
27
28

```

**13. Create a class representing a employee with private member variables (name, id, salary) and public methods (getters and setters).**

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

```
package encapsulationprograms;
class Employee{
    private String name="surya";
    private int id=12;
    private int salary=21000;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getSalary() {
        return salary;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
}
public class Ques13 {
    public static void main(String[] args) {
        Employee e = new Employee();
        System.out.println("Employee Name : " + e.getName());
        System.out.println("Employee ID : " + e.getId());
        System.out.println("Employee Salary : " + e.getSalary());
        e.setName("Rahul");
        e.setId(13);
        e.setSalary(22000);
        System.out.println("Employee Name : " + e.getName());
        System.out.println("Employee ID : " + e.getId());
        System.out.println("Employee Salary : " + e.getSalary());
    }
}
```

**15. Create a class representing a mobile phone with private member variables (brand, model, price) and public methods (getters and setters).**

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

```
package encapsulationprograms;
class mobile{
    private String brand="surya";
    private int model=123;
    private int price=21;
    public String getBrand() {
        return brand;
    }
    public void setBrand(String brand) {
        this.brand = brand;
    }
    public int getModel() {
        return model;
    }
    public void setModel(int model) {
        this.model = model;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}
public class Ques15 {
    public static void main(String[] args) {
        mobile m = new mobile();
        System.out.println("Mobile Brand : " + m.getBrand());
        System.out.println("Mobile Model : " + m.getModel());
        System.out.println("Mobile Price : " + m.getPrice());
        m.setBrand("Samsung");
        m.setModel(124);
        m.setPrice(22);
        System.out.println("Mobile Brand : " + m.getBrand());
        System.out.println("Mobile Model : " + m.getModel());
        System.out.println("Mobile Price : " + m.getPrice());
    }
}
```

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

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS375' open. The 'src' folder contains several Java files, including 'Ques17.java'. The code for 'Ques17.java' is as follows:

```
package encapsulationprograms;
class Bankaccount {
    private int accnum=123456;
    private int balance=9999999;
    public void myDeposit(int num) {
        System.out.println("the deposit amount is"+num);
    }
    public void Withdrawn(int amount) {
        System.out.println("the withdrawn amout is "+amount);
    }
    public void setAccNum(int accnum) {
        this.accnum=accnum;
    }
    public int getAccNum() {
        return accnum;
    }
    public int getBalance() {
        return balance;
    }
    public void setBalance(int balance) {
        this.balance = balance;
    }
}
public class Ques17 {
    public static void main(String[] args) {
    }
}
```

The right side of the screen shows the 'Console' tab with the following output:

```
accnum=7890
bal is=6302685
the deposit amount
is50000
the withdrawn amout
is 10000
```

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

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS375' open. The 'src' folder contains several Java files, including 'Ques19.java'. The code for 'Ques19.java' is as follows:

```
package encapsulationprograms;
class Computer{
    private String brand="surya";
    private String model="123";
    private int price=21;
    public String getBrand() {
        return brand;
    }
    public void setBrand(String brand) {
        this.brand = brand;
    }
    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}
public class Ques19 {
    public static void main(String[] args) {
    }
}
```

The right side of the screen shows the 'Console' tab with the following output:

```
dell
insp15
10000
```

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

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

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - COREJAVA PROGRAMS537/src/encapsulationprograms/Ques21.java - Eclipse IDE". The Project Explorer view on the left shows a Java project structure with various packages and source files. The main editor window displays the following Java code for class `Ques21`:

```
1 package encapsulationprograms;
2 class libaraybook{
3     private String tittle="java";
4     private String author="james";
5     private int price=2000;
6     public String getTitle() {
7         return tittle;
8     }
9     public void setTitle(String tittle) {
10        this.tittle = tittle;
11    }
12    public String getAuthor() {
13        return author;
14    }
15    public void setAuthor(String author) {
16        this.author = author;
17    }
18}
19
20 }
21 public class Ques21 {
22
23    public static void main(String[] args) {
24        book b= new book();
25        b.setAuthor("gosling");
26        b.setTitle("corejava");
27        System.out.println(b.getAuthor());
28        System.out.println(b.getTitle());
29    }
30 }
31
32 }
33
34 }
35
36 }
37
38 }
39
40 }
41
42 }
43
44 }
45
46 }
47
48 }
49
50 }
51
52 }
53
54 }
55
56 }
57
58 }
59
59 }
```

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

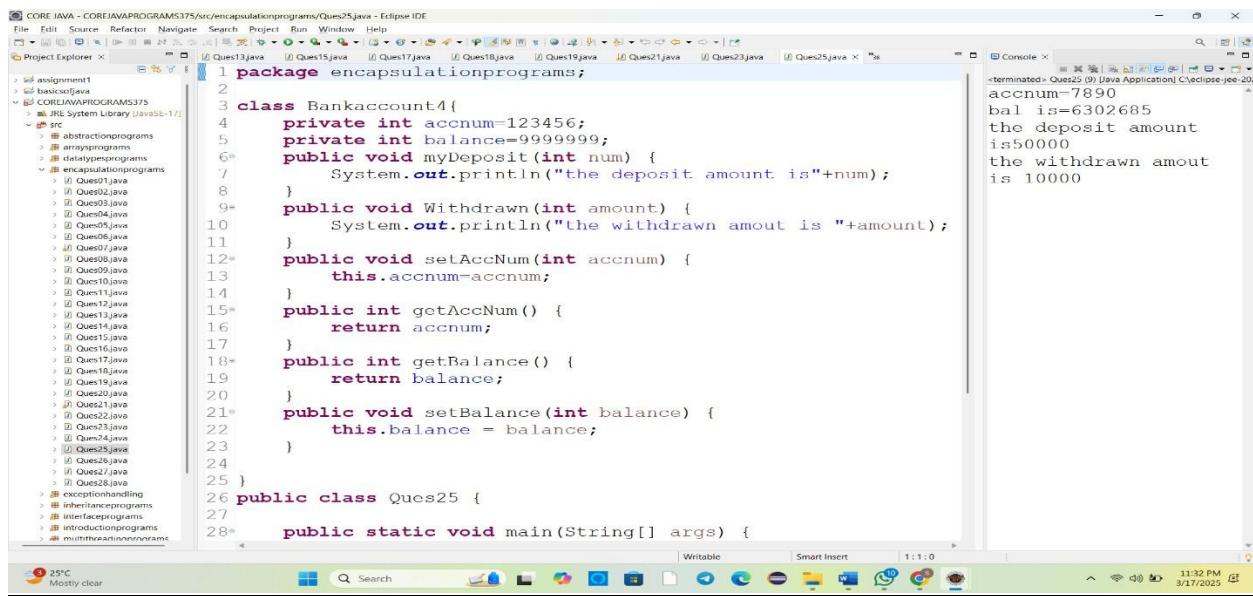
**24. Write a program to demonstrate encapsulation by accessing private member variables through public accessor methods.**

The screenshot shows the Eclipse IDE interface with the title bar "CORE JAVA - COREJAVA PROGRAMS537/src/encapsulationprograms/Ques23.java - Eclipse IDE". The Project Explorer view on the left shows a Java project structure with various packages and source files. The main editor window displays the following Java code for class `Ques23`:

```
1 package encapsulationprograms;
2 class Rectangle{
3     private int length=123;
4     private int width=987;
5     private int area=0764;
6     public int getLength() {
7         return length;
8     }
9     public void setLength(int length) {
10        this.length = length;
11    }
12    public int getWidth() {
13        return width;
14    }
15    public void setWidth(int width) {
16        this.width = width;
17    }
18    public int getArea() {
19        return area;
20    }
21    public void setArea(int area) {
22        this.area = area;
23    }
24
25 }
26 public class Ques23 {
27
28    public static void main(String[] args) {
29
30    }
31
32 }
33
34 }
35
36 }
37
38 }
39
40 }
41
42 }
43
44 }
45
46 }
47
48 }
49
49 }
```

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

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



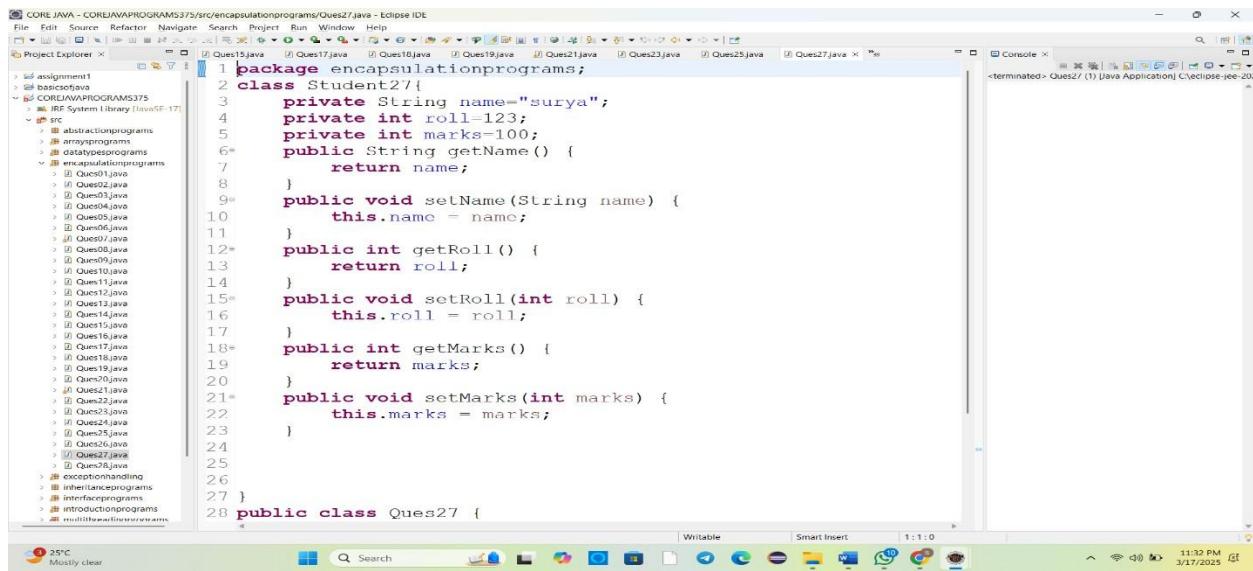
```

1 package encapsulationprograms;
2
3 class Bankaccount4{
4     private int accnum=123456;
5     private int balance=9999999;
6     public void myDeposit(int num) {
7         System.out.println("the deposit amount is"+num);
8     }
9     public void Withdrawn(int amount) {
10        System.out.println("the withdrawn amout is "+amount);
11    }
12    public void setAccNum(int accnum) {
13        this.accnum=accnum;
14    }
15    public int getAccNum() {
16        return accnum;
17    }
18    public int getBalance() {
19        return balance;
20    }
21    public void setBalance(int balance) {
22        this.balance = balance;
23    }
24
25 }
26 public class Ques25 {
27
28     public static void main(String[] args) {

```

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

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



```

1 package encapsulationprograms;
2 class Student27{
3     private String name="surya";
4     private int roll=123;
5     private int marks=100;
6     public String getName() {
7         return name;
8     }
9     public void setName(String name) {
10        this.name = name;
11    }
12    public int getRoll() {
13        return roll;
14    }
15    public void setRoll(int roll) {
16        this.roll = roll;
17    }
18    public int getMarks() {
19        return marks;
20    }
21    public void setMarks(int marks) {
22        this.marks = marks;
23    }
24
25 }
26 public class Ques27 {
27
28

```

**STATIC:**

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

```
1 package staticprograms;
2 //write a program to count the number of objects created for t
3 class objects{
4     static int count =0;
5     objects(){
6         count++;
7     }
8 }
9
10 public class Ques01 {
11     public static void main(String[] args) {
12         // Create objects
13         objects ob1=new objects();
14         objects ob2=new objects();
15         System.out.println(objects.count);
16     }
17 }
18
19
20 }
```

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

```
1 package staticprograms;
2 //factorial of a number using static method
3 class Factorial{
4     public static void factorial(int num) {
5         int fact=1;
6         for(int i=1;i<=num;i++) {
7             fact=fact*i;
8         }
9         System.out.println(fact);
10    }
11 }
12
13 public class Ques02 {
14     public static void main(String[] args) {
15         Factorial.factorial(5);
16     }
17 }
18
19 }
```

## **3. Write a program to calculate the area of a circle using a static method.**

```
1 package staticprograms;
2 //area of a circle using static method
3 class Area{
4     public static void areaOfCircle(int r) {
5         double area=3.14*r*r;
6         System.out.println(area);
7     }
8 }
9
10 public class Ques03 {
11     public static void main(String[] args) {
12         Area.areaOfCircle(20);
13     }
14 }
15
16
17 }
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS275/src/staticprograms/Ques04.java - Eclipse IDE
- Project Explorer:** Shows a large number of Java files under the package staticprograms, including Ques01.java through Ques25.java.
- Code Editor:** Displays the following Java code:

```
1 package staticprograms;
2 //sum of elements in an array
3 class Sum {
4     public static void SumOfElementsIs(int[]a) {
5         int sum=0;
6         for(int i=0;i<a.length;i++) {
7             sum=sum+a[i];
8         }
9         System.out.println(sum);
10    }
11 }
12 public class Ques04 {
13
14     public static void main(String[] args) {
15         Sum.SumOfElementsIs(new int[]{1,2,3,4,5});
16     }
17 }
18
19 }
20
```

The code defines a static method `SumOfElementsIs` that takes an integer array as an argument and prints the sum of its elements. The `main` method calls this static method with an array containing the values 1, 2, 3, 4, and 5.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS275/src/staticprograms/Ques05.java - Eclipse IDE
- Project Explorer:** Shows a large number of Java files under the package staticprograms, including Ques01.java through Ques25.java.
- Code Editor:** Displays the following Java code:

```
1 package staticprograms;
2 //maximum of two numbers using static method
3 class max{
4     public static void maxOfNum(int a,int b) {
5         int res=Math.max(a, b);
6         System.out.println(res);
7     }
8     public class Ques05 {
9
10        public static void main(String[] args) {
11            // TODO Auto-generated method stub
12            max.maxOfNum(10, 20);
13        }
14    }
15 }
16
17 }
```

The code defines a static method `maxOfNum` that takes two integers as arguments and prints the maximum value using the `Math.max` method. The `main` method calls this static method with the arguments 10 and 20.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS275/src/staticprograms/Ques06.java - Eclipse IDE
- Project Explorer:** Shows a large number of Java files under the package staticprograms, including Ques01.java through Ques25.java.
- Code Editor:** Displays the following Java code:

```
1 package staticprograms;
2 //minimum of two numbers using static method
3 class min{
4     public static void minOfNum(int a,int b) {
5         int res=Math.min(a, b);
6         System.out.println(res);
7     }
8     public class Ques06 {
9
10        public static void main(String[] args) {
11            // TODO Auto-generated method stub
12            min.minOfNum(10, 98);
13        }
14    }
15 }
16
17 }
```

The code defines a static method `minOfNum` that takes two integers as arguments and prints the minimum value using the `Math.min` method. The `main` method calls this static method with the arguments 10 and 98.

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA-COREJAVA PROGRAMS375" containing several Java files, including "Ques07.java".
- Code Editor:** Displays the content of "Ques07.java":

```
1 package staticprograms;
2 //find the power of number using static
3 class power{
4     public static void powerOFNum(double base,double pow) {
5         double res=Math.pow(base,pow);
6         System.out.println(res);
7     }
8     public class Ques07 {
9         public static void main(String[] args) {
10             power.powerOFNum(5, 5);
11         }
12     }
13 }
14 }
15 }
16 }
```

- Console:** Shows the output of the program: "3125.0".

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA-COREJAVA PROGRAMS375" containing several Java files, including "Ques08.java".
- Code Editor:** Displays the content of "Ques08.java":

```
1 package staticprograms;
2 //write a program to calculate the square root of number using
3 class Square{
4     public static void squareRootIs(int num) {
5         double res=Math.sqrt(num);
6         System.out.println(res);
7     }
8 }
9 public class Ques08 {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13         Square.squareRootIs(20);
14     }
15 }
16 }
17 }
```

- Console:** Shows the output of the program: "4.47213595499958".

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA-COREJAVA PROGRAMS375" containing several Java files, including "Ques09.java".
- Code Editor:** Displays the content of "Ques09.java":

```
1 package staticprograms;
2 //area of triangle using static method
3 class Areal{
4     public static void areaOfTriangleIs(int b,int h) {
5         double res=0.5*b*h;
6         System.out.println(res);
7     }
8 }
9 public class Ques09 {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13         Areal.areaOfTriangleIs(5, 5);
14     }
15 }
16 }
17 }
```

- Console:** Shows the output of the program: "12.5".

## 10. Write a program to calculate the simple interest using a static method.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS375/src/staticprograms/Ques10.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA PROGRAMS375" containing a "src" folder with many Java files (Ques01.java to Ques25.java) and a "staticprograms" package.
- Code Editor:** Displays the Java code for "Ques10.java".
- Console:** Shows the output of the application: "terminated> Ques10 (6) [Java Application] C:\eclipse-jee-2022-06\Ques10.java 100.08".
- System Tray:** Shows the date and time as 11:47 PM 3/17/2025.

```
1 package staticprograms;
2 //cal simple interst using static method
3 class interest{
4     public static void simpleInterst(int p,int t,float r) {
5         float res=(p+t+r)/100;
6         System.out.println(res);
7     }
8 }
9 public class Ques10 {
10
11     public static void main(String[] args) {
12         interest.simpleInterst(10000, 5, 3);
13     }
14 }
15
16
17
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS375/src/staticprograms/Ques11.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA PROGRAMS375" containing a "src" folder with many Java files (Ques01.java to Ques25.java) and a "staticprograms" package.
- Code Editor:** Displays the Java code for "Ques11.java".
- Console:** Shows the output of the application: "terminated> Ques11 (7) [Java Application] C:\eclipse-jee-2022-06\Ques11.java 674.3934909415245".
- System Tray:** Shows the date and time as 11:47 PM 3/17/2025.

```
1 package staticprograms;
2 //find the volume of cylinder using static method
3 class volume{
4     public static void volumeOfCylinderIs(float r,float h) {
5         double res=3.14*r*r*h;
6         System.out.println(res);
7     }
8 }
9 public class Ques11 {
10
11     public static void main(String[] args) {
12         volume.volumeOfCylinderIs(5.5f, 7.1f);
13     }
14 }
15
16
17
18
```

### 12. Write a program to calculate the compound interest using a static method.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA PROGRAMS375/src/staticprograms/Ques12.java - Eclipse IDE
- Project Explorer:** Shows a project named "CORE JAVA PROGRAMS375" containing a "src" folder with many Java files (Ques01.java to Ques25.java) and a "staticprograms" package.
- Code Editor:** Displays the Java code for "Ques12.java".
- Console:** Shows the output of the application: "terminated> Ques12 (8) [Java Application] C:\eclipse-jee-2022-06\Ques12.java 20000.113024889.328".
- System Tray:** Shows the date and time as 11:47 PM 3/17/2025.

```
1 package staticprograms;
2 //cal compound interest using static method
3 class compound{
4     public static void compundInterestIS(int p,float r,int y) {
5         double res = p*Math.pow((1+r/100), y);
6         System.out.println(res);
7     }
8 }
9 public class Ques12 {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13         compound.compundInterestIS(20000, 5.5f, 2);
14     }
15
16
17
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "COREJAVA/COREJAVA/PROGRAMS275/src/staticprograms".
- Code Editor:** Displays the Java code for Question 13. The code defines a static method `areaOfRect` that calculates the area of a rectangle by multiplying length (`l`) and breadth (`b`). It also contains a main method that calls this static method with parameters 20 and 10.
- Console:** Shows the output of the program: `200`.

```
1 package staticprograms;
2 //Area of rectangle
3 class rectangle {
4     public static void areaOfRect(int l,int b) {
5         int res=l*b;
6         System.out.println(res);
7     }
8 }
9 public class Ques13 {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13         rectangle.areaOfRect(20, 10);
14     }
15
16 }
17
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "COREJAVA/COREJAVA/PROGRAMS275/src/staticprograms".
- Code Editor:** Displays the Java code for Question 14. The code defines a static method `areaOfSquare` that calculates the area of a square by squaring its side (`s`). It also contains a main method that calls this static method with parameter 20.
- Console:** Shows the output of the program: `400`.

```
1 package staticprograms;
2 //Area of rectangle
3 class square {
4     public static void areaOfSquare(int s) {
5         int res=s*s;
6         System.out.println(res);
7     }
8 }
9
10 public class Ques14 {
11
12     public static void main(String[] args) {
13         // TODO Auto-generated method stub
14         square.areaOfSquare(20);
15     }
16 }
17
```

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "COREJAVA/COREJAVA/PROGRAMS275/src/staticprograms".
- Code Editor:** Displays the Java code for Question 15. The code defines a static method `areaOfRhombus` that calculates the area of a rhombus by multiplying its base (`b`) and height (`h`). It also contains a main method that calls this static method with parameters 4 and 5.
- Console:** Shows the output of the program: `400`.

```
1 package staticprograms;
2 //Area of rhombus
3 class rhombus {
4     public static void areaOfRhombus(int b,int h) {
5         int res=b*h;
6         System.out.println(res);
7     }
8 }
9
10 public class Ques15 {
11
12     public static void main(String[] args) {
13         // TODO Auto-generated method stub
14         rhombus.areaOfRhombus(4, 5);
15     }
16 }
17
```

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

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS' open. The code editor displays a Java program named 'Ques16.java'. The code defines a static method 'areaOfparallelogram' that calculates the area of a parallelogram given its base and height. It also contains a main method that calls this static method with specific values.

```
package staticprograms;
//area of parallelogram
class parallelogram {
    public static void areaOfparallelogram(int b,int h) {
        int res=b*h;
        System.out.println(res);
    }
}
public class Ques16 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        parallelogram.areaOfparallelogram(20, 70);
    }
}
```

### **17. Write a program to find the area of a trapezoid using a static method.**

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS' open. The code editor displays a Java program named 'Ques17.java'. The code defines a static method 'areaOftrapezoid' that calculates the area of a trapezoid given its top and bottom bases and height. It also contains a main method that calls this static method with specific values.

```
package staticprograms;
//area of trapezoid
class trapezoid {
    public static void areaOftrapezoid(int a,int b,int h) {
        int res=(a+b)/2*h;
        System.out.println(res);
    }
}
public class Ques17 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        trapezoid.areaOftrapezoid(5, 5, 10);
    }
}
```

### **18. Write a program to find the area of a regular polygon using a static method.**

The screenshot shows the Eclipse IDE interface with the project 'CORE JAVA - COREJAVA PROGRAMS' open. The code editor displays a Java program named 'Ques18.java'. The code defines a static method 'areaOfpolygon' that calculates the area of a regular polygon given the number of sides, side length, and radius. It also contains a main method that calls this static method with specific values.

```
package staticprograms;
//area of regular polygon
class polygon {
    public static void areaOfpolygon(int n,int l,int r) {
        int res=n/2*l*r;
        System.out.println(res);
    }
}
public class Ques18 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        polygon.areaOfpolygon(6, 4, 6);
    }
}
```

### **19. Write a program to convert temperature from Celsius to Fahrenheit using a static method.**

```
1 package staticprograms;
2 //celsius to Fahrenheit using static method
3 class Temp{
4     public static void celtoFah(float celsius) {
5         float res= (celsius*(9/5))+32;
6         System.out.println(res);
7     }
8 }
9 public class Quec19 {
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12         float temp;
13         temp=30;
14         celtoFah(temp);
15     }
16 }
17
```

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

```
1 package staticprograms;
2 //fahrenheit to Celsius using static method
3 class Temp{
4     public static void fahToCel(float fah) {
5         float res= (fah-32)*(5/9);
6         System.out.println(res);
7     }
8 }
9 public class Quec20 {
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12         float fah;
13         fah=98;
14         fahToCel(fah);
15     }
16 }
17
```

**21. Write a program to find the factorial of a number using recursion and a static method.**

**22. Write a program to find the sum of digits of a number using recursion and a static method.**

```
1 package staticprograms;
2 //factorial of a number using recursion and static
3 class fact {
4     public static int isFact(int num) {
5         if(num==0) {
6             return 1;
7         }
8         else {
9             return num*isFact(num-1);
10        }
11    }
12 }
13 public class Quec21 {
14     public static void main(String[] args) {
15         // TODO Auto-generated method stub
16         int a=fact.isFact(5);
17         System.out.println(a);
18     }
19 }
20 }
```

**23. Write a program to find the power of a number using recursion and a static method.**

```

1 package staticprograms;
2 //sum of digits of a number using recursion and static method
3 class sum1{
4     public static int sumOfDigits(int num) {
5         if(num==0) {
6             return 0;
7         }
8         else {
9             return num%10+sumOfDigits(num/10);
10        }
11    }
12 }
13
14 public class Ques22 {
15
16     public static void main(String[] args) {
17         // TODO Auto-generated method stub
18         int su= sum1.sumOfDigits(123);
19         System.out.println(su);
20     }
21 }
22

```

## 24. Write a program to find the Fibonacci series using recursion and a static method.

```

1 package staticprograms;
2 //fibonacci series using recursion and static method
3 class fib{
4     public static int isFib(int num) {
5         if(num==0) {
6             return 0;
7         }
8         else if(num==1) {
9             return 1;
10        }
11        else {
12            return isFib(num-1)+isFib(num-2);
13        }
14    }
15 }
16
17 public class Ques24 {
18
19     public static void main(String[] args) {
20         // TODO Auto-generated method stub
21         int fibo=fib.isFib();
22         System.out.println(fibo);
23     }
24 }
25

```

## 25. Write a program to reverse a string using recursion and a static method.

```

1 package staticprograms;
2 //reverse a string using recursion
3 class reverse{
4     public static String reverseIs(String str) {
5         if(str.isEmpty()) {
6             return "";
7         }
8         else {
9             return reverseIs(str.substring(1)) + str.charAt(0);
10        }
11    }
12 }
13
14 public class Ques25 {
15
16     public static void main(String[] args) {
17         // TODO Auto-generated method stub
18         String s = reverse.reverseIs("surya");
19         System.out.println(s);
20     }
21 }
22

```

**ABSTRACTION:**

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project "abstractionprograms" with files "Shape.java", "Circle.java", and "Rectangle.java".
- Code Editor:** Displays the content of Shape.java. The code defines an abstract class Shape with two abstract methods: calculateArea() and calculatePerimeter(). It then defines two concrete subclasses, Circle and Rectangle, each overriding these methods.
- Console View:** Shows the output of the application, which includes:
  - 78.5
  - 31.400000000000002
  - 25.0
  - 20.0
- Bottom Status Bar:** Shows the date and time as 12/16/2025 1:49 AM.

```
1 package abstractionprograms;
2
3 //write a abstract class shape with abstract methods "calculateArea" and
4 //abstract class shape {
5     abstract public void calculateArea();
6     abstract public void calculateperimeter();
7 }
8 class circle extends shape{
9     int r=5;
10    @Override
11    public void calculateArea() {
12        double area=3.14*r*r;
13        System.out.println(area);
14    }
15    public void calculateperimeter() {
16        double perimeter=2*3.14*r;
17        System.out.println(perimeter);
18    }
19 }
20 class rectangle extends shape{
21     int l=5;
22     int b=5;
23     @Override
24     public void calculateArea() {
25         double area=l*b;
26         System.out.println(area);
27     }
28 }
29 public void calculateperimeter() {
30     double perimeter=2*(l+b);
31 }
```

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer**: Shows a project named "Ques03" containing packages "assignment1", "basicsjava", "COMPARISONANDSWI", and "src". The "src" folder contains a package "abstractionprograms" which holds files from "Ques01.java" to "Ques44.java".
- Editor View (Ques03.java)**: Displays the Java code for "Ques03.java". The code defines an abstract class "Animal" with abstract methods "eat()" and "sleep()". It then defines two subclasses, "Dog" and "Cat", each overriding the abstract methods to print specific messages ("dog is eating" and "cat is sleeping"). Finally, it defines a main class "Ques03" with a static main method.
- Console View**: Shows the output of the Java application, displaying the printed messages: "dog is eating", "cat is sleeping", and "Ques03".
- Bottom Bar**: Includes icons for file operations (New, Open, Save, etc.), search, and system status (CPU usage, battery, network).

```
1 package abstractionprograms;
2 //write a abstract class animal with abstract methods "eat" and "sleep".impl
3 abstract class Animal{
4     abstract public void eat();
5     abstract public void sleep();
6 }
7 class Dog extends Animal{
8     public void eat(){
9         System.out.println("dog is eating");
10    }
11    public void sleep(){
12        System.out.println("dog is sleeping");
13    }
14
15 }
16 class Cat extends Animal{
17     public void eat(){
18         System.out.println("cat is eating");
19    }
20    public void sleep(){
21        System.out.println("cat is sleeping");
22    }
23 }
24
25
26
27
28
29
30 public class Ques03 {
31     public static void main(String[] args) {
```

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

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

```

1 package abstractionprograms;
2 //write a abstract class employee with abstract methods "calculatesalary" and
3 abstract class employee{
4     abstract public void calculatesalary();
5     abstract public void calculatebonus();
6 }
7 class manager extends employee{
8     public void calculatesalary() {
9         System.out.println("salary is 100000");
10    }
11    public void calculatebonus() {
12        System.out.println("bonus is 20000");
13    }
14 }
15 class clerk extends employee{
16     public void calculatesalary() {
17         System.out.println("salary is 100000");
18    }
19    public void calculatebonus() {
20        System.out.println("bonus is 20000");
21    }
22 }
23
24 public class Ques05 {
25     public static void main(String[] args) {
26         manager m = new manager();

```

**7. Write an abstract class "BankAccount with abstract methods "deposit" and "withdraw" Implement it in subclasses "Savings Account" and "CurrentAccount"**

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

```

1 package abstractionprograms;
2 //write a abstract class employee with abstract methods "calculatesalary" and
3 abstract class employee{
4     abstract public void calculatesalary();
5     abstract public void calculatebonus();
6 }
7 class manager extends employee{
8     public void calculatesalary() {
9         System.out.println("salary is 100000");
10    }
11    public void calculatebonus() {
12        System.out.println("bonus is 20000");
13    }
14 }
15 class clerk extends employee{
16     public void calculatesalary() {
17         System.out.println("salary is 100000");
18    }
19    public void calculatebonus() {
20        System.out.println("bonus is 20000");
21    }
22 }
23
24 public class Ques05 {
25     public static void main(String[] args) {
26         manager m = new manager();

```

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "abstractionprograms". Inside, there are several source files: Ques01.java through Ques34.java, and arrayprograms.java, datatypesprograms.java.
- Code Editor:** The main editor window displays the Java code for "Ques05.java". The code defines an abstract class "employee" with abstract methods "calculatesalary()" and "calculatebonus()". It then defines two subclasses: "manager" (extending "employee") and "clerk" (extending "employee"). Each subclass overrides the abstract methods to print specific salary and bonus values. Finally, the "main" method creates an instance of "manager" and calls its "calculatebonus()" method.
- Console:** On the right, the console output shows the results of running the program: "salary is 100000", "bonus is 20000", "salary is 100000", and "bonus is 20000".

```
package abstractionprograms;
//write a abstract class employee with abstract methods "calculatesalary" and
abstract class employee{
    abstract public void calculatesalary();
    abstract public void calculatebonus();
}

class manager extends employee{
    public void calculatesalary() {
        System.out.println("salary is 100000");
    }
    public void calculatebonus() {
        System.out.println("bonus is 20000");
    }
}

class clerk extends employee{
    public void calculatesalary() {
        System.out.println("salary is 100000");
    }
    public void calculatebonus() {
        System.out.println("bonus is 20000");
    }
}

public class Ques05 {
    public static void main(String[] args) {
        manager m = new manager();
    }
}
```

**11. Write an abstract class "Shape" with abstract methods "calculate Area" and "calculate Perimeter Implement it in subclasses "Triangle" and "Circle".**

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "CORE JAVA - COREJAVA PROGRAMS375".
- Ques05.java:** The active file contains Java code for an abstract class `employee` and its subclasses `manager` and `clerk`. It also includes a main method for testing.
- Console:** Displays the output of the program, showing salary and bonus values for each employee type.

```
package abstractionprograms;
//write a abstract class employee with abstract methods "calculatesalary" and
abstract class employee{
    abstract public void calculatesalary();
    abstract public void calculatebonus();
}
class manager extends employee{
    public void calcualtesalary() {
        System.out.println("salary is 100000");
    }
    public void calculatebonus() {
        System.out.println("bonus is 20000");
    }
}
class clerk extends employee{
    public void calcualtesalary() {
        System.out.println("salary is 100000");
    }
    public void calculatebonus() {
        System.out.println("bonus is 20000");
    }
}
public class Ques05 {
    public static void main(String[] args) {
        manager m = new manager();
    }
}
```

Output in Console:

```
salary is 100000
bonus is 20000
salary is 100000
bonus is 20000
```

**13. Write an abstract class "Bank" with abstract methods open Account" and "closeAccount" Implement it in subclasses "Savings Bank" and "CurrentBank"**

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

```

1 package abstractionprograms;
2 //write a abstract class bankaccount with abstract methods "deposit" and "withdrawn"
3 abstract class bank{
4     abstract public void deposit();
5     abstract public void withdrawn();
6 }
7 class savingsaccount11 extends bank{
8     public void deposit() {
9         System.out.println("deposit is 7000");
10    }
11    public void withdrawn() {
12        System.out.println("withdrawn is 2000");
13    }
14 }
15 class currentaccount11 extends bank{
16     public void deposit() {
17         System.out.println("deposit is 9000");
18    }
19    public void withdrawn() {
20        System.out.println("withdrawn is 2000");
21    }
22 }
23
24
25
26
27
28 public class Ques13 {
29     public static void main(String[] args) {
30         savingsaccount11 s= new savingsaccount11();
31         currentaccount11 c= new currentaccount11();

```

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

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

```

1 package abstractionprograms;
2 //write a abstract class figure with abstract methods "draw" and "erase".impl
3 abstract class figure{
4     abstract public void draw();
5     abstract public void erase();
6 }
7 class Rectangles extends figure{
8     public void draw() {
9         System.out.println("rectangle is drawn");
10    }
11    public void erase() {
12        System.out.println("rectangle is erased");
13    }
14 }
15 class circles extends figure{
16     public void draw() {
17         System.out.println("circle is drawn");
18    }
19    public void erase() {
20        System.out.println("circle is erased");
21    }
22 }
23
24
25
26
27
28
29
30
31 public class Ques15 {

```

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

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

```

1 package abstractionprograms;
2 //write a abstract class vecicle with abstract methods "drive" and "stop".impl
3 abstract class vecicle23{
4     abstract public void drive();
5     abstract public void stop();
6 }
7 class car22 extends vecicle23{
8     public void drive() {
9         System.out.println("car is driving state");
10    }
11    public void stop() {
12        System.out.println("car is stopeed");
13    }
14 }
15 class truck extends vecicle23{
16     public void drive() {
17         System.out.println("truck is started");
18    }
19    public void stop() {
20        System.out.println("truck is stopped");
21    }
22 }
23
24 public class Ques17 {
25     public static void main(String[] args) {
26         vecicle23 v = new car22();
27         vecicle23 v1 = new truck();
28     }
29 }

```

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

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

```

1 package abstractionprograms;
2 //write a abstract class vecicle with abstract methods "drive" and "stop".impl
3 abstract class vecicle23{
4     abstract public void drive();
5     abstract public void stop();
6 }
7 class car22 extends vecicle23{
8     public void drive() {
9         System.out.println("car is driving state");
10    }
11    public void stop() {
12        System.out.println("car is stopeed");
13    }
14 }
15 class truck extends vecicle23{
16     public void drive() {
17         System.out.println("truck is started");
18    }
19    public void stop() {
20        System.out.println("truck is stopped");
21    }
22 }
23
24 public class Ques17 {
25     public static void main(String[] args) {
26         vecicle23 v = new car22();
27         vecicle23 v1 = new truck();
28     }
29 }

```

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like assignment1, basicjava, and COREJAVAPROGRAMS375.
- Code Editor:** Displays the Java code for `Ques21.java`. The code defines an abstract class `Animal8` with abstract methods `eat()` and `sleep()`. It then defines two subclasses, `Dog8` and `Cat8`, which extend `Animal8` and implement the `eat()` and `sleep()` methods. The `main` method calls these methods for a dog and a cat.
- Console:** Shows the output of the program, which prints "dog is eating", "dog is sleeping", "cat is eating", and "cat is sleeping".
- System Bar:** Includes icons for battery, signal, and system status.

```

1 package abstractionprograms;
2 //write a abstract class animal with abstract methods "eat" and "sleep".im
3 abstract class Animal8{
4     abstract public void eat();
5     abstract public void sleep();
6 }
7 class Dog8 extends Animal8{
8     public void eat() {
9         System.out.println("dog is eating");
10    }
11    public void sleep() {
12        System.out.println("dog is sleeping");
13    }
14 }
15 class Cat8 extends Animal8{
16     public void eat() {
17         System.out.println("cat is eating");
18    }
19    public void sleep() {
20        System.out.println("cat is sleeping");
21    }
22 }
23
24
25
26
27 }
28
29
30 public class Ques21 {
31     public static void main(String[] args) {

```

**23. Write an abstract class "Shape" with abstract methods "calculateArea" and "calculate Perimeter" implement it in subclasses "Rectangle" and "Square".**

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like assignment1, basicjava, and COREJAVAPROGRAMS375.
- Code Editor:** Displays the Java code for `Ques23.java`. The code defines an abstract class `shape24` with abstract methods `calculateArea()` and `calculateperimeter()`. It then defines two subclasses, `circle24` and `rectangle24`, which extend `shape24` and implement these methods. The `main` method creates objects of both classes and prints their area and perimeter.
- Console:** Shows the output of the program, which prints the area and perimeter for a circle and a rectangle.
- System Bar:** Includes icons for battery, signal, and system status.

```

1 package abstractionprograms;
2 //write a abstract class shape with abstract methods "calculateArea" and "calcu
3 abstract class shape24 {
4     abstract public void calculateArea();
5     abstract public void calculateperimeter();
6 }
7 class circle24 extends shape24{
8     int r=5;
9     @Override
10    public void calculateArea() {
11        double area=3.14*r*r;
12        System.out.println(area);
13    }
14 }
15 class rectangle24 extends shape24{
16     int l=5;
17     int b=5;
18     @Override
19    public void calculateArea() {
20        double area=l*b;
21        System.out.println(area);
22    }
23    public void calculateperimeter() {
24        double perimeter=2*(l+b);
25    }
26 }
27
28
29
30 public class Ques23 {
31     public static void main(String[] args) {

```

**25. Write an abstract class "Bank" with abstract methods "openAccount" and "close Account". Implement it in subclasses "Savings Bank" and "CurrentBank"**

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

```

package abstractionprograms;
//write a abstract class bankaccount with abstract methods "deposit" and "withdraw"
abstract class bank13{
    abstract public void deposit();
    abstract public void withdraw();
}
class savingsaccount13 extends bank13{
    public void deposit(){
        System.out.println("deposit is 7000");
    }
    public void withdraw(){
        System.out.println("withdrawn is 2000");
    }
}
class currentaccount13 extends bank13{
    public void deposit(){
        System.out.println("deposit is 9000");
    }
    public void withdraw(){
        System.out.println("withdrawn is 2000");
    }
}
public class Ques25 {
    public static void main(String[] args) {
        bank13 b = new savingsaccount13();
        bank13 bl= new currentaccount13();
    }
}

```

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

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

```

package abstractionprograms;
//write an abstract class vehicle with abstract methods "start" and "stop", implement it
abstract class vehicle88{
    abstract public void start();
    abstract public void stop();
}
class car88 extends vehicle88{
    public void start(){
        System.out.println("car is started");
    }
    public void stop(){
        System.out.println("car is stopped");
    }
}
class motorcycle88 extends vehicle88{
    public void start(){
        System.out.println("motorcycle is started");
    }
    public void stop(){
        System.out.println("motorcycle is stopped");
    }
}
public class Ques27 {
    public static void main(String[] args) {
        car c = new car();
        motorcycle c= new motorcycle();
    }
}

```

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

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

```

1 package Abstractionprograms;
2 //write a program to demonstrate abstraction with abstract class shape56
3 abstract class shape56 {
4     abstract public void calculateArea();
5     abstract public void calculatePerimeter();
6 }
7
8 class circle56 extends shape56 {
9     int r=5;
10    @Override
11    public void calculateArea() {
12        double area=3.14*r*r;
13        System.out.println(area);
14    }
15
16    public void calculatePerimeter() {
17        double perimeter=2*3.14*r;
18        System.out.println(perimeter);
19    }
20 }
21
22 class rectangle56 extends shape56 {
23     int l=5;
24     int b=5;
25     @Override
26     public void calculateArea() {
27         double area=l*b;
28         System.out.println(area);
29     }
30
31     public void calculatePerimeter() {
32         double perimeter=2*(l+b);
33     }

```

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

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

```

1 package abstractionprograms;
2 //write a program to demonstrate abstraction with abstract class bankaccount with abstract methods "deposit" and "withdraw".
3 abstract class bank111 {
4     abstract public void deposit();
5     abstract public void withdraw();
6 }
7
8 class savingsaccount111 extends bank111 {
9     public void deposit() {
10         System.out.println("deposit is 7000");
11     }
12     public void withdraw() {
13         System.out.println("withdrawn is 2000");
14     }
15 }
16
17 class currentaccount111 extends bank111 {
18     public void deposit() {
19         System.out.println("deposit is 9000");
20     }
21     public void withdraw() {
22         System.out.println("withdrawn is 2000");
23     }
24 }
25
26
27 public class Ques31 {
28     public static void main(String[] args) {
29         savingsaccount111 s = new savingsaccount111();
30         currentaccount111 c = new currentaccount111();
31     }
32 }

```

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

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

```

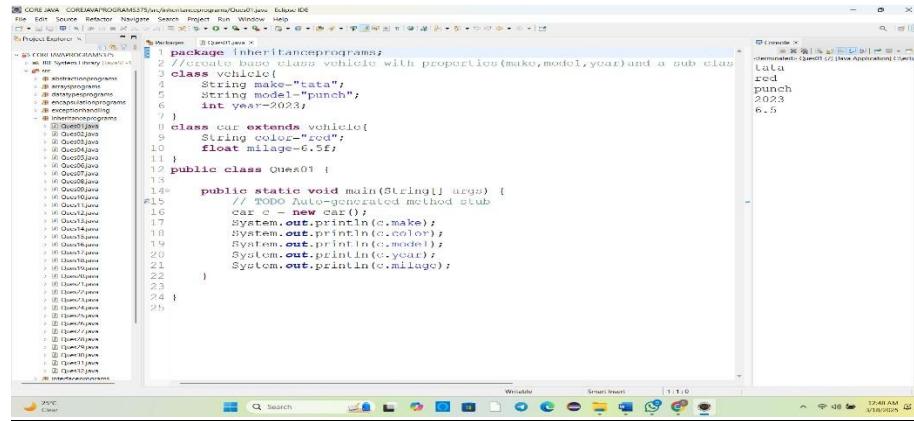
1 package abstractionprograms;
2 //write a program to demonstrate abstraction with abstract class animal with abstract methods "eat" and "sleep".Implement it in subclasses dog and cat.
3 abstract class Animal333 {
4     abstract public void eat();
5     abstract public void sleep();
6 }
7
8 class Dog333 extends Animal333 {
9     public void eat() {
10         System.out.println("dog is eating");
11     }
12     public void sleep() {
13         System.out.println("dog is sleeping");
14     }
15 }
16
17 class Cat333 extends Animal333 {
18     public void eat() {
19         System.out.println("cat is eating");
20     }
21     public void sleep() {
22         System.out.println("cat is sleeping");
23     }
24 }
25
26
27 public class Ques33 {
28     public static void main(String[] args) {
29     }
30 }

```

## INHERITANCE:

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

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

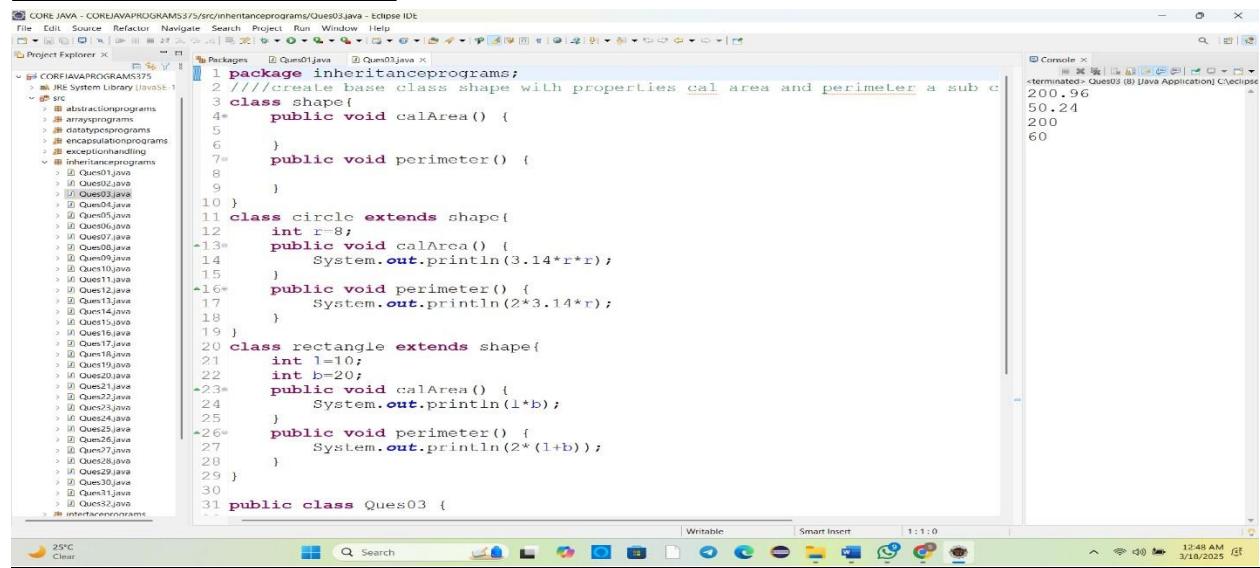


The screenshot shows the Eclipse IDE interface with the following code in the editor:

```
1 package inheritanceprograms;
2 //create base class vehicle with properties(make,model,year)and a sub class
3 class vehicle {
4     String make="tata";
5     String model="punch";
6     int year=2023;
7 }
8 class car extends vehicle{
9     String color="red";
10    float mileage=6.5f;
11 }
12 public class Ques01 {
13
14     public static void main(String[] args) {
15         TODO Auto-generated method stub
16         car c=new car();
17         System.out.println(c.make);
18         System.out.println(c.color);
19         System.out.println(c.model);
20         System.out.println(c.year);
21         System.out.println(c.mileage);
22     }
23 }
24 }
```

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

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



The screenshot shows the Eclipse IDE interface with the following code in the editor:

```
1 package inheritanceprograms;
2 //create base class shape with properties cal area and perimeter a sub c
3 class shape{
4     public void calArea() {
5
6     }
7     public void perimeter() {
8
9     }
10 }
11 class circle extends shape{
12     int r=8;
13     public void calArea() {
14         System.out.println(3.14*r*r);
15     }
16     public void perimeter() {
17         System.out.println(2*3.14*r);
18     }
19 }
20 class rectangle extends shape{
21     int l=10;
22     int b=20;
23     public void calArea() {
24         System.out.println(l*b);
25     }
26     public void perimeter() {
27         System.out.println(2*(l+b));
28     }
29 }
30
31 public class Ques03 {
```

**5. Create a base class "Animal" with properties (name, age) and subclasses "Dog" and cat**

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

```

package inheritanceprograms;
// create base class animal with properties name and age a sub class dog
class animal {
    public void name() {
        System.out.println("snowbel");
    }
    public void age() {
        System.out.println("5years");
    }
}
class dog extends animal{
    public void name() {
        System.out.println("leo");
    }
    public void age() {
        System.out.println("5years");
    }
}
class cat extends animal{
    public void name() {
        System.out.println("snowbel");
    }
    public void age() {
        System.out.println("2years");
    }
}
public class Ques05 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}

```

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

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

```

package inheritanceprograms;
// create base class animal with properties name and age a sub class dog
class animal {
    public void name() {
        System.out.println("snowbel");
    }
    public void age() {
        System.out.println("5years");
    }
}
class dog extends animal{
    public void name() {
        System.out.println("leo");
    }
    public void age() {
        System.out.println("5years");
    }
}
class cat extends animal{
    public void name() {
        System.out.println("snowbel");
    }
    public void age() {
        System.out.println("2years");
    }
}
public class Ques05 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}

```

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

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

```

package inheritanceprograms;
//create base class person with properties(name,age) and a sub class student
class person{
    String name="surya";
    int age=23;
}
class student extends person{
    int rollno =146;
    String subject="english";
}
class teacher extends person{
    int rollno =147;
    String subject "maths";
}
public class Ques09 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        student s= new student();
        teacher t=new teacher();
        System.out.println(s.age);
        System.out.println(s.name);
        System.out.println(s.rollno);
        System.out.println(s.subject);
        System.out.println(t.age);
        System.out.println(t.name);
        System.out.println(t.rollno);
        System.out.println(t.subject);
    }
}

```

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

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

```

package inheritanceprograms;
//create base class bankaccount with properties(accountno,balance) and a s
class bankaccount{
    int accono=123456;
    int bal=2333445;
}
class savingsaccount extends bankaccount{
}
class currentaccount extends bankaccount{
}
public class Ques11{
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        savingsaccount sa= new savingsaccount();
        currentaccount cu=new currentaccount();
        System.out.println(sa.accono);
        System.out.println(sa.bal);
        System.out.println(cu.accono);
        System.out.println(cu.bal);
    }
}

```

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS375/src/inheritanceprograms/Ques13.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project management.
- Project Explorer:** Shows the project structure under C:\COREJAVA\PROGRAMS375\src\inheritanceprograms. It contains a package named inheritanceprograms and several Java files (Ques01.java through Ques32.java).
- Code Editor:** The main editor window displays the Java code for `Ques13.java`. The code defines two classes, `shape13` and `rectangle13`, which inherit from `shape13`. Both classes implement methods `calArea()` and `perimeter()`.
- Output View:** The right-hand margin shows the output of the `Ques13` application. It prints:
  - 200.96
  - 50.24
  - 200
  - 60
  - red
  - rectangle
- System Tray:** Shows icons for battery, signal strength, and system status.
- Bottom Bar:** Shows the date and time (12:49 AM 3/18/2025) and standard Windows-style taskbar icons.

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

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

CORE JAVA - COREJAVA/PROGRAMS375/src/inheritanceprograms/Ques15.java - Eclipse IDE

File Edit Source Refactor Navigate Project Run Window Help

Project Explorer X

CoreJavaPrograms375

JRE System Library JavaSE-1

src

- abstractionprograms
- arrayprograms
- datatypeprograms
- encapsulationprograms
- exceptionhandling
- interfaceprograms
- Ques01.java
- Ques02.java
- Ques03.java
- Ques04.java
- Ques05.java
- Ques06.java
- Ques07.java
- Ques08.java
- Ques09.java
- Ques10.java
- Ques11.java
- Ques12.java
- Ques13.java
- Ques14.java
- Ques15.java
- Ques16.java
- Ques17.java
- Ques18.java
- Ques19.java
- Ques20.java
- Ques21.java
- Ques22.java
- Ques23.java
- Ques24.java
- Ques25.java
- Ques26.java
- Ques27.java
- Ques28.java
- Ques29.java
- Ques30.java
- Ques31.java
- Ques32.java

Package inheritanceprograms;

//create base class vehicle with properties (make,model,year) and a sub class

class vehicle15{

String make="tata";

String model="punch";

int year=2023;

}

class car15 extends vehicle15{

String color="red";

float milage=6.5f;

}

public class Ques15 {

public static void main(String[] args) {

// TODO Auto-generated method stub

car15 c15 = new car15();

System.out.println(c15.make);

System.out.println(c15.color);

System.out.println(c15.model);

System.out.println(c15.year);

System.out.println(c15.milage);

}

}

Console X Ques07.java

<terminated> Ques15 (Java Application) C:\Users\DELL

tata

red

punch

2023

6.5

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

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

```

1 package inheritanceprograms;
2 //create base class fruit with properties(name,color) and a sub class apple
3 class fruit{
4     String name="orange";
5     String color="orange";
6 }
7 class apple extends fruit{
8     String taste="super";
9     float size=6.5f;
10 }
11 class banana extends fruit{
12     String taste="super";
13     float size=6.5f;
14 }
15
16 public class Ques17 {
17
18     public static void main(String[] args) {
19         // TODO Auto-generated method stub
20         apple a = new apple();
21         banana b=new banana();
22         System.out.println(a.color);
23         System.out.println(a.name);
24         System.out.println(b.color);
25         System.out.println(b.name);
26         System.out.println(a.taste);
27         System.out.println(b.taste);
28         System.out.println(a.size);
29         System.out.println(b.size);
30     }
31 }

```

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

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

```

1 package inheritanceprograms;
2 //create base class animal with properties name and age a sub class dog
3 class animal19{
4     public void name() {
5
6     }
7     public void age() {
8
9     }
10 }
11 class dog19 extends animal19{
12     public void name() {
13         System.out.println("leo");
14     }
15     public void age() {
16         System.out.println("5years");
17     }
18 }
19 class cat19 extends animal19{
20     public void name() {
21         System.out.println("snowbel");
22     }
23     public void age() {
24         System.out.println("2years");
25     }
26 }
27
28 public class Ques19 {
29
30     public static void main(String[] args) {
31         // TODO Auto-generated method stub

```

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

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

```

CORE JAVA - COREJAVA PROGRAMS375/src/inheritanceprograms/Ques21.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Project Explorer Packages Ques01.java Ques03.java Ques05.java Ques09.java Ques11.java Ques13.java Ques15.java Ques21.java
COREJAVA PROGRAMS375 JRE System Library [JavaSE-1.8]
abstractionprograms arrayprograms datatypesprograms encapsulationsprograms exceptionhandling inheritanceprograms
Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java Ques27.java Ques28.java Ques29.java Ques30.java Ques31.java Ques32.java interfaceprograms
1 package inheritanceprograms;
2 //create base class person with properties(name,age) and a sub class student
3 class person21{
4     String name="surya";
5     int age=23;
6 }
7 class student21 extends person21{
8     int rollno =146;
9     String subject="english";
10
11 }
12 class teacher21 extends person21{
13     int rollno =147;
14     String subject="maths";
15
16 }
17
18 public class Ques21 {
19
20     public static void main(String[] args) {
21         // TODO Auto-generated method stub
22         student21 s= new student21();
23         teacher21 t=new teacher21();
24         System.out.println(s.age);
25         System.out.println(s.name);
26         System.out.println(s.rollno);
27         System.out.println(s.subject);
28         System.out.println(t.age);
29         System.out.println(t.name);
30         System.out.println(t.rollno);
31         System.out.println(t.subject);
32     }
33 }

```

Console > Ques21.java  
terminated> Ques21 (9) [Java Application] Eclipse IDE  
23  
surya  
146  
english  
23  
surya  
147  
maths

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

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

```

CORE JAVA - COREJAVA PROGRAMS375/src/inheritanceprograms/Ques23.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Project Explorer Packages Ques01.java Ques03.java Ques05.java Ques09.java Ques23.java
COREJAVA PROGRAMS375 JRE System Library [JavaSE-1.8]
abstractionprograms arrayprograms datatypesprograms encapsulationsprograms exceptionhandling inheritanceprograms
Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java Ques27.java Ques28.java Ques29.java Ques30.java Ques31.java Ques32.java interfaceprograms
1 package inheritanceprograms;
2 //create base class shape with properties type,color cal area and perimeter
3 class shape23{
4     String type="rectangle";
5     String color="red";
6     public void calArea() {
7
8     }
9     public void perimeter() {
10
11     }
12 }
13
14 class circle23 extends shape23{
15     int r=8;
16     public void calArea() {
17         System.out.println(3.14*r*r);
18     }
19     public void perimeter() {
20         System.out.println(2*3.14*r);
21     }
22 }
23 class rectangle23 extends shape23{
24     int l=10;
25     int b=20;
26     public void calArea() {
27         System.out.println(l*b);
28     }
29     public void perimeter() {
30         System.out.println(2*(l+b));
31     }
32 }

```

Console > Ques23.java  
terminated> Ques23 (0) [Java Application] Eclipse IDE  
200.96  
50.24  
200  
60  
red  
rectangle

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a package named "inheritanceprograms" containing several Java files: Ques01.java, Ques03.java, Ques05.java, Ques09.java, Ques23.java, and Ques25.java.
- Code Editor:** Displays the content of Ques25.java. The code defines a base class `vehicle25` with properties `make`, `model`, and `year`. It extends this class in the `car25` class, adding `color` and `milage`. The `main` method creates a `car25` object and prints its properties.
- Console:** Shows the output of the program:

```
tata
red
punch
2023
6.5
```
- Taskbar:** Includes icons for various applications like File Explorer, Task View, and Taskbar.
- System Tray:** Shows system status icons.

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a package named "inheritanceprograms" containing several Java files: Ques01.java, Ques03.java, Ques05.java, Ques09.java, Ques23.java, and Ques27.java.
- Code Editor:** Displays the content of Ques27.java. It defines a base class `animal27` with `name` and `age` methods. It extends this class in the `dog27` and `cat27` subclasses, adding specific implementations for each.
- Console:** Shows the output of the program:

```
2years
snowbel
leo
5years
```
- Taskbar:** Includes icons for various applications like File Explorer, Task View, and Taskbar.
- System Tray:** Shows system status icons.

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "COREJAVA PROGRAMS375" with a package named "inheritanceprograms". Inside the package, there are several Java files: Ques01.java, Ques02.java, Ques03.java, Ques04.java, Ques05.java, Ques07.java, Ques08.java, Ques09.java, Ques10.java, Ques11.java, Ques12.java, Ques13.java, Ques14.java, Ques15.java, Ques16.java, Ques17.java, Ques18.java, Ques19.java, Ques20.java, Ques21.java, Ques22.java, Ques23.java, Ques24.java, Ques25.java, Ques26.java, Ques27.java, Ques28.java, Ques29.java, and Ques32.java.
- Code Editor:** Displays the content of the file "Ques29.java". The code defines three classes: "shape29", "circle29", and "rectangle29", which inherit from the base class "shape29". The "shape29" class has properties "type" and "color". The "circle29" class has a property "r=8". The "rectangle29" class has properties "l=10" and "b=20". Both subclasses implement methods "calArea()" and "perimeter()".
- Console:** Shows the output of running the program. It prints:
  - 200.96
  - 50.24
  - 200
  - 60
  - red
  - rectangle
- System Bar:** Shows the date and time as 3/18/2025 at 12:52 AM.

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

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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "COREJAVA PROGRAMS375" with a "src" folder containing numerous Java files (Ques01.java, Ques03.java, etc.) and several subfolders like "abstractionprograms", "arraysprograms", "datatypeprograms", "encapsulationprograms", "exceptionhandling", and "inheritanceprograms".
- Code Editor:** Displays the content of the file "Ques31.java". The code defines a class "employee31" with properties name, id, and salary. It then defines a subclass "manager31" extending "employee31" with additional properties dep and desg. Finally, it defines a public class "Ques31" with a main method that creates an instance of "manager31", prints its properties, and outputs "m" (likely a reference to the manager object) at the end.
- Console:** Shows the output of the program. The output is:

```
It
software developer
123
surya
20233
```
- Taskbar:** At the bottom, there is a taskbar with various icons and a system tray showing the date and time (12:52 AM, 3/18/2025).

## EXCEPTION HANDLING:

### 1. Write a program to handle ArrayIndexOutOfBoundsException.

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - COREJAVA PROGRAMS375". The code editor displays a file named "Ques01.java" which contains the following code:

```
1 package exceptionhandling;
2 //program to handle ArrayIndexOutOfBoundsException
3 public class Ques01 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int a[]={1,2,3,4,5};
8         try {
9             for(int i=0;i<a.length;i++) {
10                 System.out.println(a[i]);
11             }
12         } catch(ArrayIndexOutOfBoundsException e) {
13             System.out.println("array size is more");
14         }
15     }
16 }
17 }
```

The console output shows the numbers 1, 2, 3, 4, and 5, followed by the message "array size is more".

## 2. Implement a program to handle Arithmetic Exception (division by zero).

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - COREJAVA PROGRAMS375". The code editor displays a file named "Ques02.java" which contains the following code:

```
1 package exceptionhandling;
2 //program to handle Arithmetic Exception
3 public class Ques02 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         try {
8             int a=10;
9             int b=0;
10            int c=a/b;
11            System.out.println(c);
12        } catch(ArithmaticException e) {
13            System.out.println("enter non zero value");
14        }
15    }
16 }
17 }
```

The console output shows the message "enter non zero value".

## 3. Write a program to handle NullPointerException.

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - COREJAVA PROGRAMS375". The code editor displays a file named "Ques03.java" which contains the following code:

```
1 package exceptionhandling;
2
3 import java.util.Scanner;
4
5 //program to handle null pointer Exception
6 public class Ques03 {
7
8     public static void main(String[] args) {
9         // TODO Auto-generated method stub
10        Scanner scan = new Scanner(System.in);
11        try {
12            String str= null;
13            int length=str.length();
14            System.out.println(length);
15        } catch(NullPointerException e) {
16            System.out.println("null pointer");
17        }
18    }
19 }
20 }
```

The console output shows the message "null pointer".

## 4. Implement a program to handle FileNotFoundException.

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - COREJAVA PROGRAMS375". The "src" folder contains several Java files, including "Ques4.java". The code in "Ques4.java" demonstrates how to handle a `FileNotFoundException`. It attempts to read from a file named "example.txt" using a `Scanner`. If the file is not found, it prints an error message and continues execution. The "Console" tab shows the output: "File not found at path: example.txt" and "Operation complete.".

```

package exceptionhandling;
//program to handle file not found exception
import java.io.File;
public class Ques4 {
    public static void main(String[] args) {
        String filePath = "example.txt"; // Change this to your file
        try {
            File file = new File(filePath);
            Scanner scanner = new Scanner(file);
            System.out.println("File contents:");
            while (scanner.hasNextLine()) {
                System.out.println(scanner.nextLine());
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found at path: " + filePath);
        } finally {
            System.out.println("Operation complete.");
        }
    }
}

```

## 5. Write a program to handle NumberFormatException.

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - COREJAVA PROGRAMS375". The "src" folder contains several Java files, including "Ques5.java". The code in "Ques5.java" demonstrates how to handle a `NumberFormatException`. It tries to parse a string "123abc" into an integer. Since "123abc" is not a valid integer, a `NumberFormatException` is caught, and an error message is printed. The "Console" tab shows the output: "Error: Invalid number format for input: 123abc" and "Operation complete.".

```

package exceptionhandling;
//handle number format exception
public class Ques5 {
    public static void main(String[] args) {
        String numberString = "123abc"; // Invalid number string
        try {
            int number = Integer.parseInt(numberString);
            System.out.println("Number is: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format for input: " + numberString);
        } finally {
            System.out.println("Operation complete.");
        }
    }
}

```

## 6. Implement a program to handle IOException.

The screenshot shows the Eclipse IDE interface with a Java project named "CORE JAVA - COREJAVA PROGRAMS375". The "src" folder contains several Java files, including "Ques6.java". The code in "Ques6.java" demonstrates how to handle an `IOException`. It reads lines from a file named "example.txt" using a `BufferedReader`. If an I/O error occurs, an error message is printed. The "Console" tab shows the output: "Error: An I/O error occurred while reading example.txt (The system cannot find the file specified)" and "Operation complete.".

```

package exceptionhandling;
//handle exception
import java.io.BufferedReader;
public class Ques6 {
    public static void main(String[] args) {
        String filePath = "example.txt"; // Change this to your file
        BufferedReader reader = null;
        try {
            reader = new BufferedReader(new FileReader(filePath));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("Error: An I/O error occurred - " + e.getMessage());
        } finally {
            try {
                if (reader != null) {
                    reader.close();
                }
            } catch (IOException e) {
                System.out.println("Error closing the file: " + e.getMessage());
            }
        }
        System.out.println("Operation complete.");
    }
}

```

## 7. Write a program to handle ClassNotFoundException.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS375". Inside, there's a package named "exceptionhandling" containing a file "Ques07.java".
- Code Editor:** Displays the code for "Ques07.java":
 

```
1 package exceptionhandling;
2 //handle class not found exception
3 public class Ques07 {
4     public static void main(String[] args) {
5         try {
6             // Attempting to load a class that doesn't exist
7             Class<?> myClass = Class.forName("com.example.NonExiste
8             System.out.println("Class found: " + myClass.getName());
9         } catch (ClassNotFoundException e) {
10             System.out.println("Error: Class not found - " + e.getMessage());
11         } finally {
12             System.out.println("Operation complete.");
13         }
14     }
15 }
```
- Console:** Shows the output of the program, indicating that the class "com.example.NonExiste" was not found.
- System Tray:** Shows the date and time as "24°C Clear" and "1:14 AM 3/18/2025".

## 8. Implement a program to handle StackOverflowError.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS375". Inside, there's a package named "exceptionhandling" containing a file "Ques08.java".
- Code Editor:** Displays the code for "Ques08.java":
 

```
1 package exceptionhandling;
2 //handle stack over flow exception
3 public class Ques08 {
4
5     // Recursive method that causes stack overflow
6     public static void recursiveMethod(int count) {
7         System.out.println("Recursion depth: " + count);
8         recursiveMethod(count + 1); // No base case = infinite recursion
9     }
10
11
12     public static void main(String[] args) {
13         try {
14             recursiveMethod(1);
15         } catch (StackOverflowError e) {
16             System.out.println("Error: Stack overflow occurred! Recur
17         } finally {
18             System.out.println("Operation complete.");
19         }
20     }
21 }
```
- Console:** Shows the output of the program, indicating that the recursion depth reached 11170 before an error occurred.
- System Tray:** Shows the date and time as "24°C Clear" and "1:14 AM 3/18/2025".

## 9. Write a program to handle NegativeArraySizeException.

**10. Implement a program to handle NegativeArraySizeException.**

```

CORE JAVA - COREJAVA PROGRAMS375/src/exceptionhandling/Ques09.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer × Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java
src assignment1 basicjavaprograms COREJAVA PROGRAMS375 JRE System Library [JavaSE-1]
  abstractionprograms arraysprograms datatypesprograms encapsulationprograms exceptionhandling
    Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java
    Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java
    Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java inheritanceprograms
    introductionprograms multithreadingprograms operatorsprograms overloadingprograms staticenvironments
Console × terminated - Ques09 (6) Java Application | Eclipse jee-2022-12-Win32
Recursion depth: 11170
Recursion depth: 11171
Recursion depth: 11172
Recursion depth: 11173
Recursion depth: 11174
Recursion depth: 11175
Recursion depth: 11176
Recursion depth: 11177
Recursion depth: 11178
Recursion depth: 11179
Recursion depth: 11180
Recursion depth: 11181
Recursion depth: 11182
Recursion depth: 11183
Recursion depth: 11184
Recursion depth: 11185
Recursion depth: 11186
Recursion depth: 11187
Recursion depth: 11188
Recursion depth: 11189
Recursion depth: 11190
Recursion depth: 11191
Recursion depth: 11192
Recursion depth: 11193
Recursion depth: 11194
Error: Stack overflow occurred! Recursion went too deep.
Operation complete.

```

## 10. Implement a program to handle InterruptedException.

```

CORE JAVA - COREJAVA PROGRAMS375/src/exceptionhandling/Ques10.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer × Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java Ques10.java
src assignment1 basicjavaprograms COREJAVA PROGRAMS375 JRE System Library [JavaSE-1]
  abstractionprograms arraysprograms datatypesprograms encapsulationprograms exceptionhandling
    Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java
    Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java
    Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java inheritanceprograms
    introductionprograms multithreadingprograms operatorsprograms overloadingprograms staticenvironments
Console × terminated - Ques10 (7) Java Application | Eclipse jee-2022-12-Win3
Thread going to sleep...

```

## 11. Write a program to handle ArrayStoreException.

```

CORE JAVA - COREJAVA PROGRAMS375/src/exceptionhandling/Ques11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer × Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java Ques11.java
src assignment1 basicjavaprograms COREJAVA PROGRAMS375 JRE System Library [JavaSE-1]
  abstractionprograms arraysprograms datatypesprograms encapsulationprograms exceptionhandling
    Ques01.java Ques02.java Ques03.java Ques04.java Ques05.java Ques06.java Ques07.java Ques08.java Ques09.java
    Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java
    Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java inheritanceprograms
    introductionprograms multithreadingprograms operatorsprograms overloadingprograms staticenvironments
Console × terminated - Ques11 (8) Java Application | Eclipse jee-2022-12-Win3
Error: Cannot store incompatible type in array
array = java.lang.Integer
Operation complete.

```

## 12. Implement a program to handle IllegalStateException.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVAPROGRAMS375" containing several Java files (Ques01.java through Ques13.java) and a "src" folder with various package structures.
- Code Editor:** Displays the content of `Ques13.java`, which contains Java code demonstrating the use of an Iterator and handling of `NoSuchElementException`.
- Console:** Shows the output of running the program, which includes a stack trace for an `UnsupportedOperationException` occurring at `Iterator.remove()` due to calling `remove()` without first calling `next()`.

### 13. Write a program to handle NoSuchElementException.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "CORE JAVA - COREJAVAPROGRAMS375" containing several Java files (Ques01.java through Ques13.java) and a "src" folder with various package structures.
- Code Editor:** Displays the content of `Ques13.java`, which contains Java code demonstrating the use of an Iterator and handling of `NoSuchElementException`.
- Console:** Shows the output of running the program, which includes a stack trace for an `UnsupportedOperationException` occurring at `Iterator.remove()` due to calling `remove()` without first calling `next()`.

### 14. Implement a program to handle UnsupportedOperationException.

**14. Write a program to handle UnsupportedOperationException.**

```

package exceptionhandling;
import java.util.*;
public class Ques14 {
    public static void main(String[] args) {
        try {
            // Example: Trying to modify an unmodifiable list
            List<String> fruits = Arrays.asList("Apple", "Banana", "Orange");
            System.out.println("Fruits: " + fruits);
            // Attempting to add a new item - causes UnsupportedOperationException
            fruits.add("Grapes");
        } catch (UnsupportedOperationException e) {
            System.out.println("Error: Operation not supported - " + e);
        } finally {
            System.out.println("Operation complete.");
        }
    }
}

```

The screenshot shows the Eclipse IDE interface with the code editor open to Ques14.java. The code demonstrates attempting to add an element to an unmodifiable list, which results in a UnsupportedOperationException. The output window shows the expected error message and the final output.

## 15. Write a program to handle UnsupportedOperationException.

```

package exceptionhandling;
import java.util.*;
public class Ques15 {
    public static void main(String[] args) {
        try {
            // Example: Trying to modify an unmodifiable list
            List<String> fruits = Arrays.asList("Apple", "Banana", "Orange");
            System.out.println("Fruits: " + fruits);
            // Attempting to add a new item - causes UnsupportedOperationException
            fruits.add("Grapes");
        } catch (UnsupportedOperationException e) {
            System.out.println("Error: Operation not supported - " + e);
        } finally {
            System.out.println("Operation complete.");
        }
    }
}

```

The screenshot shows the Eclipse IDE interface with the code editor open to Ques15.java. The code is identical to Ques14.java, demonstrating the handling of an UnsupportedOperationException when trying to modify an unmodifiable list. The output window shows the error message and the final output.

## 16. Implement a program to handle ConcurrentModificationException.

```

package exceptionhandling;
//handle concurrentmodification exception
import java.util.*;
public class Ques16 {
    public class ConcurrentModificationExceptionExample {
        public static void main(String[] args) {
            List<String> fruits = new ArrayList<>(Arrays.asList("Apple", "Banana", "Orange"));
            try {
                // Attempt to modify the list while iterating (causes ConcurrentModificationException)
                for (String fruit : fruits) {
                    if (fruit.equals("Banana")) {
                        fruits.remove(fruit); // Throws ConcurrentModificationException
                    }
                }
            } catch (ConcurrentModificationException e) {
                System.out.println("Error: Concurrent modification detected - " + e);
            } finally {
                System.out.println("Operation complete.");
            }
            System.out.println("Fruits after modification: " + fruits);
        }
    }
}

```

The screenshot shows the Eclipse IDE interface with the code editor open to Ques16.java. The code implements a ConcurrentModificationExceptionExample class that attempts to remove elements from a list while it is being iterated over. This causes a ConcurrentModificationException. The output window shows the error message and the final output.

## 17. Write a program to handle IllegalArgumentException.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS375" with a "src" folder containing various Java files (Ques01.java to Ques25.java) and a "JRE System Library [JavaSE-1.8]" entry.
- Code Editor:** Displays the content of Ques17.java, which handles an `IllegalArgumentException`. The code includes a method `calculateSquareRoot` that throws an exception if the input is negative, and a `main` method that catches this exception and prints the result or error message.
- Console:** Shows the output of the application running in the background, indicating an error due to a non-negative number being passed to the program.

## 18. Implement a program to handle SecurityException.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "CORE JAVA - COREJAVA PROGRAMS375" with a "src" folder containing various Java files (Ques01.java to Ques25.java) and a "JRE System Library [JavaSE-1.8]" entry.
- Code Editor:** Displays the content of Ques18.java, which handles a `SecurityException`. It attempts to set a security manager and catches any resulting exception, printing a success message or an error message if it occurs.
- Console:** Shows the output of the application running in the background, indicating that the security manager was successfully set.

## 19. Write a program to handle DateTimeParseException.

```

package exceptionhandling;
import java.time.format.DateTimeParseException;
import java.time.LocalDate;

public class Ques19 {
    public static void main(String[] args) {
        String dateStr = "2024-02-30"; // Invalid date
        try {
            // Define date format
            DateTimeFormatter formatter = DateTimeFormatter.ISO_LOCAL_DATE;
            // Attempt to parse the date string
            LocalDate date = LocalDate.parse(dateStr, formatter);
            System.out.println("Parsed date: " + date);
        } catch (DateTimeParseException e) {
            System.out.println("Error: Invalid date format or value");
        } finally {
            System.out.println("Operation complete.");
        }
    }
}

```

## 20. Implement a program to handle PatternSyntaxException.

```

package exceptionhandling;
//handle pattern pattern syntax exception
import java.util.regex.*;
public class Ques20 {
    public static void main(String[] args) {
        String regex = "[A-Z+"; // Invalid regex (missing closing )
        try {
            // Compile the regex pattern - throws PatternSyntaxException
            Pattern pattern = Pattern.compile(regex);
            System.out.println("Pattern compiled successfully.");
        } catch (PatternSyntaxException e) {
            System.out.println("Error: Invalid regex pattern - " + e);
            System.out.println("Index: " + e.getIndex());
            System.out.println("Pattern: " + e.getPattern());
        } finally {
            System.out.println("Operation complete.");
        }
    }
}

```

## 21. Write a program to handle MissingResourceException.

```

package exceptionhandling;
//handle missing resource exception
import java.util.ResourceBundle;
public class Ques21 {
    public static void main(String[] args) {
        try {
            // Load a non-existent resource bundle (causes MissingResourceException)
            ResourceBundle bundle = ResourceBundle.getBundle("messages");
            // Access a key (won't reach here if bundle is missing)
            String message = bundle.getString("greeting");
            System.out.println("Message: " + message);
        } catch (MissingResourceException e) {
            System.out.println("Error: Resource not found -");
        }
    }
}

```

## 22. Implement a program to handle FormatterClosedException.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVA\PROGRAMS\27\src\exceptionhandling\Ques22.java - Eclipse IDE
- Project Explorer:** Shows a project named "exceptionhandling" containing files like Ques22.java, Ques1.java, and Ques2.java.
- Code Editor:** Displays Java code for exception handling, specifically demonstrating the use of try-catch-finally blocks to handle missing resource exceptions.
- Toolbars:** Standard Eclipse toolbars for file operations, search, and navigation.
- Bottom Status Bar:** Shows the current file as "Ques22.java", the line number as "1 1:0", and the date/time as "3/18/2023 1:20:00 PM".

```
package exceptionhandling;
//handle formatter closed exception
import java.util.*;
public class Ques22 {
    public class MissingResourceExceptionExample {
        public static void main(String[] args) {
            try {
                // Load a non-existent resource bundle (causes MissingResourceBundle exception)
                ResourceBundle.getBundle("missing");
                // Access a key (won't reach here if bundle is missing)
                String message = bundle.getString("greeting");
                System.out.println("Message: " + message);
            } catch (MissingResourceException e) {
                System.out.println("Error: Resource not found = " + e.getMessage());
                System.out.println("Key: " + e.getKey());
            } finally {
                System.out.println("Operation complete.");
            }
        }
    }
}
```

**23. Write a program to handle BufferOverflowException.**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a tree view of Java projects and source files. Projects include "assignments", "COREJAVA PROGRAMS 375", and "exceptionhandling". Source files under "exceptionhandling" include Ques01Java through Ques23Java.
- Code Editor:** Displays the Java code for Ques23.java. The code demonstrates exception handling, specifically catching a `BufferOverflowException`.
- Terminal/Console:** Shows the output of running the program, which ends with an error message about a buffer overflow and then continues with "Operation complete."

```
package exceptionhandling;
import java.nio.*;
public class Ques23 {
    public static void main(String[] args) {
        try {
            // Create a ByteBuffer with capacity 5
            ByteBuffer buffer = ByteBuffer.allocate(5);
            // Put 5 bytes into the buffer - this is fine
            for (int i = 0; i < 5; i++) {
                buffer.put((byte) i);
            }
            // Try to add another byte - this will cause BufferOverflowException
            buffer.put((byte) 100);
        } catch (BufferOverflowException e) {
            System.out.println("Error: Buffer overflow detected - " + e);
        } finally {
            System.out.println("Operation complete.");
        }
    }
}
```

**24. Implement a program to handle Buffer UnderflowException.**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "CORE JAVA - COREJAVA PROGRAMS75".
- Editor:** Displays the Java code for "Ques24.java". The code demonstrates reading from a ByteBuffer and catching BufferUnderflowException.
- Terminal (Console):** Shows the output of running the program, which includes three successful byte reads (Byte 1: 10, Byte 2: 20, Byte 3: 30) followed by an error message indicating a buffer underflow detected during the fourth read.

```
package exceptionhandling;
//handle buffer under flow exception
import java.nio.*;
public class Ques24 {
    public static void main(String[] args) {
        try {
            // Create a ByteBuffer with capacity 5 and add 3 bytes
            ByteBuffer buffer = ByteBuffer.allocate(5);
            buffer.put((byte) 10);
            buffer.put((byte) 20);
            buffer.put((byte) 30);

            // Flip the buffer to switch to reading mode
            buffer.flip();

            // Read 3 bytes - fine
            System.out.println("Byte 1: " + buffer.get());
            System.out.println("Byte 2: " + buffer.get());
            System.out.println("Byte 3: " + buffer.get());

            // Try to read another byte - causes BufferUnderflowException
            System.out.println("Byte 4: " + buffer.get());
        } catch (BufferUnderflowException e) {
            System.out.println("Error: Buffer underflow detected - ");
        } finally {
            System.out.println("Operation complete.");
        }
    }
}
```

## 25. Write a program to handle DateTimeException

```

package exceptionhandling;
// handle date and time exception
import java.time.*;
public class Ques25 {
    public static void main(String[] args) {
        try {
            // Invalid date: February 30 does not exist
            LocalDate invalidDate = LocalDate.of(2024, 2, 30);
            System.out.println("Date: " + invalidDate);
        } catch (DateTimeException e) {
            System.out.println("Error: Invalid date or time - " + e);
        }
    }
}

```

## INTERFACES:

### 1.Create interfaces "Drawable" and "Resizable" with methods "draw()" and "resize()".

Implement them in a class representing a shape

```

package interfaceprograms;
//create interfaces "drawable" and "resizable" with methods draw()
//implement them in a class representing shape
interface drawable{
    public void draw();
    public void resize();
}
interface resizable{
    public void draw();
    public void resize();
}
class shape implements drawable,resizable{
    public void draw() {
        System.out.println("shape is drawn");
    }
    public void resize() {
        System.out.println("shape is resize");
    }
}
public class Ques01 {
    public static void main(String[] args) {
        shape s=new shape();
        s.draw();
        s.resize();
    }
}

```

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

### 3. Create interfaces "Flyable" and "Swimable" with methods "fly()" and "swim()".

Implement them in classes representing a bird and a fish.

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

```

1 package interfaceprograms;
2 //create interfaces "flyable" and "swimable" with methods fly() and
3 //implement them in a class representing bird and fish
4 interface flyable{
5     public void fly();
6     public void swim();
7 }
8 interface swimable{
9     public void fly();
10    public void swim();
11 }
12
13 class bird implements flyable,swimable{
14     public void fly() {
15         System.out.println("bird is flyn");
16     }
17     public void swim() {
18         System.out.println("bird is swim");
19     }
20 }
21
22 class fish implements flyable,swimable{
23     public void fly() {
24         System.out.println("fish is flyn");
25     }
26     public void swim() {
27         System.out.println("swim is swim");
28     }
29 }
30 public class Ques03 {
31

```

## 5. Create interfaces "Comparable" and "Cloneable" with methods "compareTo()" and "clone()" Implement them in classes representing a number and a person.

## 6. Write a program to demonstrate interface implementation by creating objects of the number and person classes and invoking interface methods 7. Create interfaces "List" and "Set" with methods "add()", "remove()", and "contains()"

```

1 package interfaceprograms;
2 //create interfaces "comparable" and "clonable" with methods comparable
3 //implement them in a class representing number and person
4 interface comparable{
5     public void compareTo();
6     public void cloneObject();
7 }
8 interface cloneable{
9     public void compareTo();
10    public void cloneObject();
11 }
12
13 class number implements comparable,cloneable{
14     public void compareTo() {
15         System.out.println("number is compareton");
16     }
17     public void cloneObject() {
18         System.out.println("number is clone");
19     }
20 }
21
22 class person implements comparable,cloneable{
23     public void compareTo() {
24         System.out.println("person is compareton");
25     }
26     public void cloneObject() {
27         System.out.println("clone is clone");
28     }
29 }
30 public class Ques05 {
31

```

## 8.Implement them in classes representing an array list and a hash set 8. Write a program to demonstrate interface implementation by creating objects of the array list and hash set classes and invoking interface methods.

## 9. Create Interfaces "Printable" and "Scannable" with methods "print()" and "scan()" Implement them in classes representing a printer and a scanner.

**10. Write a program to demonstrate interface implementation by creating objects of the printer and scanner classes and invoking interface methods.**

The screenshot shows the Eclipse IDE interface with the Project Explorer and Editor tabs open. The code in the editor is as follows:

```
1 package interfaceprograms;
2 //create interfaces "list" and "set" with methods add() and remove
3 //Implement them in a class representing arraylist and hashset
4 interface list {
5     public void add();
6     public void remove();
7     public void contains();
8 }
9
10 interface set {
11     public void add();
12     public void remove();
13     public void contains();
14 }
15 class arraylist implements list, set {
16     public void add() {
17         System.out.println("arraylist is added");
18     }
19     public void remove() {
20         System.out.println("arraylist is removed");
21     }
22     public void contains() {
23         System.out.println("arraylist contains heterogenous elements");
24     }
25 }
26 class hashset implements list, set {
27     public void add() {
28         System.out.println("arraylist is added");
29     }
30     public void remove() {
31         System.out.println("arraylist is removed");
32     }
33 }
```

The Console tab shows the output of the program:

```
terminated: Quest7 (1) [Java Application] C:\eclipse\jee-2022-12-R-win32-x86_64\workspace\interfaceprograms\Quest7\src\interfaceprograms\Quest7.java
arraylist is added
arraylist is removed
arraylist contains heterogenous elements
arraylist is added
arraylist is removed
arraylist contains heterogenous elements
```

**11. Create interfaces "Sortable" and "Searchable" with methods "sort()" and "search()**  
Implement them in classes representing a list and a dictionary

**12. Write a program to demonstrate interface implementation by creating objects of the list and dictionary classes and invoking interface methods.**

The screenshot shows the Eclipse IDE interface with the Project Explorer and Editor tabs open. The code in the editor is as follows:

```
1 package interfaceprograms;
2 //create interfaces "sortable" and "searchable" with methods sort() and search()
3 //Implement them in a class representing list and dictionary
4 interface sortable {
5     public void sort();
6     public void search();
7 }
8
9 interface searchable {
10     public void sort();
11     public void search();
12 }
13
14 class list1 implements sortable, searchable {
15     public void sort() {
16         System.out.println("printer is sorted");
17     }
18     public void search() {
19         System.out.println("printer is scan");
20     }
21 }
22
23 class dictionary implements sortable, searchable {
24     public void sort() {
25         System.out.println("printer is sorted");
26     }
27     public void search() {
28         System.out.println("printer is scan");
29     }
30 }
31
32 public class Quest11 {
33     public static void main(String[] args) {
34         list1 l=new list1();
35     }
36 }
```

The Console tab shows the output of the program:

```
terminated: Quest11 (1) [Java Application] C:\eclipse\jee-2022-12-R-win32-x86_64\workspace\interfaceprograms\Quest11\src\interfaceprograms\Quest11.java
printer is scan
printer is scan
printer is scan
printer is scan
printer is sorted
```

**13. Create interfaces "Serializable" and "DE serializable" with methods "serialize()" and "deserialize()". Implement them in classes representing a file and a database.**

**14. Write a program to demonstrate interface implementation by creating objects of the file and database classes and invoking interface methods.**

```

package interfaceprograms;
//create interfaces "Serializable" and "Deserializable" with methods serialize()
//implement them in a class representing file and database
interface Serializable{
    public void serialize();
    public void deserialize();
}
interface Deserializable{
    public void serialize();
    public void deserialize();
}
class File implements Serializable, Deserializable{
    public void serialize(){
        System.out.println("file is serialized");
    }
    public void deserialize(){
        System.out.println("file is deserialized");
    }
}
class Database implements Serializable, Deserializable{
    public void serialize(){
        System.out.println("database is serialized");
    }
    public void deserialize(){
        System.out.println("database is deserialized");
    }
}
public class Ques13 {
    public static void main(String[] args) {
        File f=new File();
    }
}

```

**15. Create interfaces "Encryptable" and "Decryptable" with methods "encrypt()" and "decrypt()". Implement them in classes representing an encoder and a decoder.**

**16. Write a program to demonstrate interface implementation by creating objects of the encoder and decoder classes and invoking interface methods.**

```

package interfaceprograms;
//create interfaces "Encryptable" and "Decryptable" with methods encrypt()
//implement them in a class representing encoder and decoder
interface Encryptable{
    public void encrypt();
    public void decrypt();
}
interface Decryptable{
    public void encrypt();
    public void decrypt();
}
class Encoder implements Encryptable, Decryptable{
    public void encrypt(){
        System.out.println("encoder is encrypted");
    }
    public void decrypt(){
        System.out.println("encoder is deserializable");
    }
}
class Decoder implements Encryptable, Decryptable{
    public void encrypt(){
        System.out.println("decoder is encrypted");
    }
    public void decrypt(){
        System.out.println("decoder is deserializable");
    }
}
public class Ques15 {
    public static void main(String[] args) {
        Encoder e=new Encoder();
    }
}

```

**17. Create interfaces "Runnable" and "Walkable" with methods "run()" and "walk()". Implement them in classes representing a cheetah and a tortoise.**

**18. Write a program to demonstrate interface implementation by creating objects of the cheetah and tortoise classes and invoking interface methods.**

```

1 package interfaceprograms;
2 //create interfaces "runnable" and "walkable" with methods run() and walk()
3 //implement them in a class representing cheetah and tortoise
4 interface runnable{
5     public void run();
6     public void walk();
7 }
8 interface walkable{
9     public void run();
10    public void walk();
11 }
12 class cheetah implements runnable,walkable{
13     public void run() {
14         System.out.println("cheetah is runnabe!");
15     }
16     public void walk() {
17         System.out.println("cheetah is deserizable!");
18     }
19 }
20 class tortise implements runnable,walkable{
21     public void run() {
22         System.out.println("tortise is runnabe!");
23     }
24     public void walk() {
25         System.out.println("tortise is deserizable!");
26     }
27 }
28 public class Ques17 {
29     public static void main(String[] args) {
30         cheetah l=new cheetah();
31     }
}

```

**19. Create interfaces "Playable" and "Recordable" with methods "play()" and "record()".  
Implement them in classes representing a music player and a recorder.**

**20. Write a program to demonstrate interface implementation by creating objects of the  
music player and recorder classes and invoking interface methods.**

```

1 package interfaceprograms;
2 //create interfaces "playable" and "recordable" with methods play() and record()
3 //implement them in a class representing music player and recorder
4 interface playable{
5     public void run();
6     public void walk();
7 }
8 interface recordable{
9     public void run();
10    public void walk();
11 }
12 class musicplayer implements playable,recordable{
13     public void run() {
14         System.out.println("music player is runnabe!");
15     }
16     public void walk() {
17         System.out.println("music player is deserizable!");
18     }
19 }
20 class recorder implements playable,recordable{
21     public void run() {
22         System.out.println("recorder is runnabe!");
23     }
24     public void walk() {
25         System.out.println("recorder is deserizable!");
26     }
27 }
28 public class Ques19 {
29     public static void main(String[] args) {
30         musicplayer l=new musicplayer();
31     }
}

```

**21. Create interfaces "Drawable" and "Erasable" with methods "draw()" and "erase()".  
Implement them in classes representing a whiteboard and a chalkboard.**

**22. Write a program to demonstrate interface implementation by creating objects of the  
whiteboard and chalkboard classes and invoking interface methods.**

```

1 package interfaceprograms;
2 //create interfaces "drawable" and "erasable" with methods draw() and erase()
3 //implement them in a class representing music whiteboard and chalkboard
4 interface drawable{
5     public void draw();
6     public void erase();
7 }
8 interface erasable{
9     public void draw();
10    public void erase();
11 }
12 class whiteboard implements drawable,erasable{
13*     public void draw() {
14         System.out.println("music player is drawn");
15     }
16*     public void erase() {
17         System.out.println("music player is erased");
18     }
19 }
20 class chalkboard implements drawable,erasable{
21*     public void draw() {
22         System.out.println("rchalkboard is drawn");
23     }
24*     public void erase() {
25         System.out.println("rchalkboard is erased");
26     }
27 }
28 public class Ques21{
29
30*     public static void main(String[] args) {
31         whiteboard l=new whiteboard();

```

## 23. Create interfaces "Sendable" and "Receivable" with methods "send()" and "receive()". Implement them in classes representing a transmitter and a receiver.

## 24. Write a program to demonstrate interface implementation by creating objects of the transmitter and receiver classes and invoking interface methods.

```

1 package interfaceprograms;
2 //create interfaces "sendable" and "receivable" with methods send() and receive()
3 //implement them in a class representing music transmitter and receiver
4 interface sendable{
5     public void send();
6     public void receive();
7 }
8 interface receivable{
9     public void send();
10    public void receive();
11 }
12 class transmitter implements sendable,receivable{
13*     public void send() {
14         System.out.println("music player is sent");
15     }
16*     public void receive() {
17         System.out.println("music player is received");
18     }
19 }
20 class receiver implements sendable,receivable{
21*     public void send() {
22         System.out.println("receiver is sent");
23     }
24*     public void receive() {
25         System.out.println("receiver is received");
26     }
27 }
28 public class Ques23{
29
30*     public static void main(String[] args) {
31         transmitter l=new transmitter();

```

## 25. Create interfaces "Encryptable" and "Decryptable" with methods "encrypt()" and "decrypt()". Implement them in classes representing an encryption algorithm

**26. decryption Write a program to demonstrate interface implementation by creating objects of the encryption and decryption classes and invoking interface methods.**

The screenshot shows the Eclipse IDE interface with the code for Question 26. The code defines two interfaces: `sendable` and `receivable`, and two classes: `transmitter` and `receiver`. The `transmitter` class implements both interfaces and has methods to send and receive messages. The `receiver` class also implements both interfaces and has methods to send and receive messages. The `main` method creates a `transmitter` object and calls its `send` and `receive` methods. The output in the console shows the results of these operations.

```
package interfaceprograms;
//create interfaces "sendable" and "receivable" with methods send() and receive()
interface sendable{
    public void send();
    public void receive();
}
interface receivable{
    public void send();
    public void receive();
}
class transmitter implements sendable,receivable{
    public void send() {
        System.out.println("music player is sendeed");
    }
    public void receive() {
        System.out.println("music player is deserizable");
    }
}
class receiver implements sendable,receivable{
    public void send() {
        System.out.println("rreciver issended");
    }
    public void receive() {
        System.out.println("rreciver is deserizable");
    }
}
public class Ques23 {
    public static void main(String[] args) {
        transmitter t=new transmitter();
        t.send();
        t.receive();
    }
}
```

Console output:

```
terminated - Ques23 [3] Java Application | Eclipse IDE
music player is
deserizable
music player is
deserizable
rreciver is
deserizable
rreciver issended
```

**27. Create interfaces "Writable" and "Readable" with methods "write()" and "read()".**

Implement them in classes representing a text file and a 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.**

The screenshot shows the Eclipse IDE interface with the code for Question 28. The code defines two interfaces: `writable` and `readable`, and two classes: `textfile` and `databasetable`. The `textfile` class implements both interfaces and has methods to write and read data. The `databasetable` class also implements both interfaces and has methods to write and read data. The `main` method creates a `textfile` object and calls its `write` and `read` methods. The output in the console shows the results of these operations.

```
package interfaceprograms;
//create interfaces "writable" and "readable" with methods write() and read()
interface writable{
    public void write();
    public void read();
}
interface readable{
    public void write();
    public void read();
}
class textfile implements writable,readable{
    public void write() {
        System.out.println("encoder iswritied");
    }
    public void read() {
        System.out.println("encoder is deserizable");
    }
}
class databasetable implements writable,readable{
    public void write() {
        System.out.println("decoder iswritied");
    }
    public void read() {
        System.out.println("decoder is deserizable");
    }
}
public class Ques27 {
    public static void main(String[] args) {
        textfile t=new textfile();
        t.write();
        t.read();
    }
}
```

Console output:

```
terminated - Ques27 [3] Java Application | Eclipse IDE
encoder is
deserizable
encoder is
deserizable
decoder is
deserizable
decoder iswritied
```

**29. Create interfaces "Drawable" and "Printable" with methods "draw()" and "print()".**

Implement them in classes representing a canvas and a printer.

**30. Write a program to demonstrate interface implementation by creating objects of the canvas and printer classes and invoking interface methods.**

```

1 package interfaceprograms;
2 //create interfaces "drawable" and "reziable" with methods draw() and res
3 //implement them in a class representing shape
4 interface drawable29{
5     public void draw();
6     public void resize();
7 }
8 interface reziable29{
9     public void draw();
10    public void resize();
11 }
12
13 class shape29 implements drawable29,reziable29{
14     public void draw() {
15         System.out.println("shape is drawn");
16     }
17     public void resize() {
18         System.out.println("shape is resize");
19     }
20 }
21
22 public class Ques29 {
23
24     public static void main(String[] args) {
25         shape s=new shape();
26         s.draw();
27         s.resize();
28     }
29 }
30
31 }

```

**31. Create interfaces "Runnable" and "Callable" with methods "run()" and "call()". Implement them in classes representing a thread and a task.**

**32. Write a program to demonstrate interface implementation by creating objects of the thread and task classes and invoking interface methods.**

```

1 package interfaceprograms;
2 //create interfaces "runnable" and "callable" with methods run() and call
3 //implement them in a class representing thread and task
4 interface runnable31{
5     public void run();
6     public void call();
7 }
8
9 interface callable31{
10    public void run();
11    public void call();
12 }
13
14 class thread31 implements runnable31,callable31{
15     public void run() {
16         System.out.println("thread is runn");
17     }
18     public void call() {
19         System.out.println("thread is call");
20     }
21 }
22
23 class task31 implements runnable31,callable31{
24     public void run() {
25         System.out.println("task is runn");
26     }
27     public void call() {
28         System.out.println("call is call");
29     }
30 }
31

```

## **MULTITHREADING:**

**1. Write a Java program to create multiple threads and display their names.**

**Ques01.java**

```

1 package multithreadingprograms;
2 //create multiple threads display there names
3 class demo extends Thread {
4     public void run() {
5         System.out.println(currentThread().getName()+"first");
6         try {
7             Thread.sleep(5000);
8         } catch (InterruptedException e) {
9             // TODO Auto-generated catch block
10            e.printStackTrace();
11        }
12        System.out.println(currentThread().getName()+"second");
13        try {
14            Thread.sleep(5000);
15        } catch (InterruptedException e) {
16            // TODO Auto-generated catch block
17            e.printStackTrace();
18        }
19        System.out.println(currentThread().getName()+"third");
20        try {
21            Thread.sleep(5000);
22        } catch (InterruptedException e) {
23            // TODO Auto-generated catch block
24            e.printStackTrace();
25        }
26    }
27 }
28
29 public class Ques01 {
30
31     public static void main(String[] args) throws Exception {
32         // TODO Auto-generated method stub
33         demo d1 = new demo();
34         d1.setName("surya");
35         d1.setName("surya");
36         d1.setName("ram");
37         d1.start();
38
39     }
40 }
41
42 }

```

## 2. Implement a program to demonstrate thread synchronization using synchronized blocks.

**Ques02.java**

```

1 package multithreadingprograms;
2 //thread synchroization using synchronized blocks
3 class mull {
4     private int count = 0;
5
6     // Lock object to synchronize on
7     private final Object lock = new Object();
8
9     public void increment() {
10         // Synchronized block to ensure only one thread modifies 'count' at a time
11         synchronized (lock) {
12             count++;
13             System.out.println(Thread.currentThread().getName() + " - Count: " + count);
14         }
15     }
16
17     public int getCount() {
18         return count;
19     }
20
21
22     public class Ques02{
23
24         public static void main(String[] args) {
25             SharedResource resource = new SharedResource();
26
27             // Create two threads that increment the shared counter
28             Thread t1 = new Thread(() -> {
29                 for (int i = 0; i < 5; i++) {
30                     resource.increment();
31                     try { Thread.sleep(100); } catch (InterruptedException e) { e.printStackTrace();
32                 }
33                 , "Thread 1";
34             });
35
36             Thread t2 = new Thread(() -> {
37                 for (int i = 0; i < 5; i++) {
38                     resource.increment();
39                     try { Thread.sleep(100); } catch (InterruptedException e) { e.printStackTrace();
40                 }
41                 , "Thread 2";
42             });
43
44             // Start threads
45         }
46     }
47 }

```

## 3. Write a Java program to create multiple threads and display their priorities.

**CORE JAVA - COREJAVA PROGRAMS375/src/multithreadingprograms/Ques03.java - Eclipse IDE**

```

1 package multithreadingprograms;
2 //create multiple threads and dispaly their priorities
3 class MyThread extends Thread {
4     public MyThread(String name, int priority) {
5         super(name);
6         setPriority(priority);
7     }
8
9     @Override
10    public void run() {
11        System.out.println(getName() + " is running with priority " + getPriority());
12    }
13
14
15 public class Ques03 {
16     public static void main(String[] args) {
17         // Create multiple threads with different priorities
18         MyThread t1 = new MyThread("Thread 1", Thread.MIN_PRIORITY); // Priority 1
19         MyThread t2 = new MyThread("Thread 2", Thread.NORM_PRIORITY); // Priority 5
20         MyThread t3 = new MyThread("Thread 3", Thread.MAX_PRIORITY); // Priority 10
21
22         // Start threads
23         t1.start();
24         t2.start();
25         t3.start();
26     }
27
28
29 }
30

```

Console <terminated> Ques03 (11) [Java Application] Eclipse-JEE-202

```

Thread 1 is running with
priority 1
Thread 2 is running with
priority 5
Thread 3 is running with
priority 10

```

#### 4. Implement a program to create a thread pool and execute multiple tasks using Executor Service

**CORE JAVA - COREJAVA PROGRAMS375/src/multithreadingprograms/Ques04.java - Eclipse IDE**

```

1 package multithreadingprograms;
2 //create thread pool and execute multiple tasks using executor service
3 import java.util.concurrent.*;
4
5 public class Ques04 {
6     private int taskId;
7
8     public void Task(int taskId) {
9         this.taskId = taskId;
10    }
11
12    public void run() {
13        System.out.println("Task " + taskId + " is running on: " +
14        try {
15            Thread.sleep(1000); // Simulate some work
16        } catch (InterruptedException e) {
17            System.out.println("Task " + taskId + "');");
18        }
19    }
20}
21
22

```

Console <terminated> Ques03 (11) [Java Application] Eclipse-JEE-202

```

Thread 1 is running with
priority 1
Thread 3 is running with
priority 10
Thread 2 is running with
priority 5

```

## 5. Write a Java program to create multiple threads and join them

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows files like Ques01.java through Ques26.java.
- Code Editor:** Displays the code for Ques05.java, which creates three threads (t1, t2, t3) and waits for them to finish using the join() method.
- Console:** Shows the output of the program, indicating the start and finish of each thread and the main thread exiting.
- Taskbar:** Shows the date and time as 3/18/2025 2:02 AM.

```
1 package multithreadingprograms;
2 //create multiple threads and join them
3 class MyThread123 extends Thread {
4     private final int threadId;
5
6     public MyThread123(int threadId) {
7         this.threadId = threadId;
8     }
9
10    @Override
11    public void run() {
12        System.out.println("Thread " + threadId + " is running.");
13        try {
14            Thread.sleep(1000); // Simulating work
15        } catch (InterruptedException e) {
16            System.out.println("Thread " + threadId + " was interrupted.");
17        }
18        System.out.println("Thread " + threadId + " has finished.");
19    }
20}
21
22 public class Ques05 {
23     public static void main(String[] args) {
24         // Create multiple threads
25         MyThread123 t1 = new MyThread123(1);
26         MyThread123 t2 = new MyThread123(2);
27         MyThread123 t3 = new MyThread123(3);
28
29         // Start threads
30         t1.start();
31         t2.start();
32         t3.start();
33
34         // Join threads - main thread waits for them to finish
35         try {
36             t1.join();
37             System.out.println("Thread 1 joined.");
38             t2.join();
39             System.out.println("Thread 2 joined.");
40             t3.join();
41             System.out.println("Thread 3 joined.");
42         } catch (InterruptedException e) {
43             System.out.println("Main thread interrupted.");
44         }
45
46         System.out.println("All threads have finished. Main thread exiting.");
47     }
48 }
49
```

## 6. Implement a program to demonstrate deadlock condition in multithreading.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows files like Ques01.java through Ques26.java.
- Code Editor:** Displays the code for Ques06.java, which demonstrates a deadlock between two threads (Thread 1 and Thread 2) waiting for shared resources.
- Console:** Shows the output of the program, indicating the acquisition of locks and the resulting deadlock.
- Taskbar:** Shows the date and time as 3/18/2025 2:02 AM.

```
1 package multithreadingprograms;
2
3 static final Object lock1 = new Object();
4 static final Object lock2 = new Object();
5
6 public void produce() {
7     // TODO Auto-generated method stub
8 }
9
10 public void accessResource(String name) {
11     // TODO Auto-generated method stub
12 }
13
14 public void consume99() {
15     // TODO Auto-generated method stub
16 }
17
18 public void produce99(String message) {
19     // TODO Auto-generated method stub
20 }
21
22 public void increment() {
23     // TODO Auto-generated method stub
24 }
25
26 class Thread1 extends Thread {
27     @Override
28     public void run() {
29         synchronized (SharedResource.lock1) {
30             System.out.println("Thread 1: Acquired lock1, waiting for lock2...");
31             try { Thread.sleep(100); } catch (InterruptedException e) {}
32             synchronized (SharedResource.lock2) {
33                 System.out.println("Thread 1: Acquired lock2");
34             }
35         }
36     }
37 }
38
39 class Thread2 extends Thread {
40     @Override
41     public void run() {
42         synchronized (SharedResource.lock2) {
43             System.out.println("Thread 2: Acquired lock2, waiting for lock1...");
44             try { Thread.sleep(100); } catch (InterruptedException e) {}
45             synchronized (SharedResource.lock1) {
46                 System.out.println("Thread 2: Acquired lock1");
47             }
48         }
49     }
50 }
51
52 public class Ques06 {
53 }
```

## 7. Write a Java program to create multiple threads and interrupt them.

```

1 package multithreadingprograms;
2 //multiple threads and interrupt them
3 public class Ques07 {
4     class MyTask extends Thread {
5         public MyTask(String name) {
6             super(name);
7         }
8         @Override
9         public void run() {
10            System.out.println(getName() + " started.");
11            try {
12                // Simulating long-running task
13                for (int i = 1; i <= 10; i++) {
14                    System.out.println(getName() + " is working... " + i);
15                    Thread.sleep(500); // Simulate work with sleep
16                }
17            } catch (InterruptedException e) {
18                System.out.println(getName() + " was interrupted!");
19            } finally {
20                System.out.println(getName() + " finished.");
21            }
22        }
23    }
24 }
25
26 public class Ques07 {
27     public static void main(String[] args) {
28         MyTask t1 = new MyTask("Thread 1");
29         MyTask t2 = new MyTask("Thread 2");
30         MyTask t3 = new MyTask("Thread 3");
31         t1.start();
32         t2.start();
33         t3.start();
34
35         // Interrupt all threads after 2 seconds
36         try {
37             Thread.sleep(2000);
38             System.out.println("Interrupting threads...");
39             t1.interrupt();
40             t2.interrupt();
41             t3.interrupt();
42         } catch (InterruptedException e) {
43             System.out.println("Main thread interrupted!");
44         }
45     }
46 }
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

```

## 8. Implement a program to demonstrate thread local variables

```

1 package multithreadingprograms;
2 //demonstrate thread local variables
3 public class Ques08 {
4     class MyTask extends Thread {
5         private static ThreadLocal<Integer> threadLocalValue = ThreadLocal
6
7             @Override
8             public void run() {
9                 System.out.println(Thread.currentThread().getName() + " in
10
11                     threadLocalValue.set(threadLocalValue.get() + 100);
12                     System.out.println(Thread.currentThread().getName() + " mo
13
14             }
15         }
16     }
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
387
388
389
389
390
391
392
393
394
395
396
397
397
398
399
399
400

```

## 9. Write a Java program to create multiple threads and wait for them to complete using Count Down Latch

**Project Explorer**

```

1 package multithreadingprograms;
2 //create multiple threads and wait for them to complete using count down latch
3 import java.util.concurrent.CountDownLatch;
4
5 public class Ques09 {
6     class Worker extends Thread {
7         private final CountDownLatch latch;
8         private final int workerId;
9
10        public Worker(int workerId, CountDownLatch latch) {
11            this.workerId = workerId;
12            this.latch = latch;
13        }
14
15        @Override
16        public void run() {
17            System.out.println("Worker " + workerId + " is starting...");
18
19            try {
20                // Simulate some work
21                Thread.sleep(1000 * workerId);
22            } catch (InterruptedException e) {
23                System.out.println("Worker " + workerId + " was interrupted.");
24            }
25
26            System.out.println("Worker " + workerId + " has finished.");
27            latch.countDown(); // Decrease the latch count after task completion
28        }
29    }
30
31
32    public static void main(String[] args) {
33        int numWorkers = 5;
34        CountDownLatch latch = new CountDownLatch(numWorkers);
35
36        // Create and start worker threads
37        for (int i = 1; i <= numWorkers; i++) {
38            new Worker(i, latch).start();
39        }
40
41        try {
42            latch.await(); // Wait until all workers have completed
43        } catch (InterruptedException e) {
44            e.printStackTrace();
45        }
46    }
47}

```

**Console**

```

Ques09 (7) [Java Application] Eclipse JEE 2022-12 R win32 x
Thread 2: Acquired lock2, waiting for lock1...
Thread 1: Acquired lock1, waiting for lock2...

```

## 10. Implement a program to demonstrate thread priorities in Java.

**Project Explorer**

```

1 package multithreadingprograms;
2 //demonstrates thread priorities in java
3 class MyThread1 extends Thread {
4     public MyThread1(String name, int priority) {
5         super(name);
6         setPriority(priority);
7     }
8
9     @Override
10    public void run() {
11        for (int i = 1; i <= 5; i++) {
12            System.out.println(getName() + " (Priority: " + getPriority() +
13                try {
14                    Thread.sleep(500);
15                } catch (InterruptedException e) {
16                    System.out.println(getName() + " was interrupted.");
17                }
18            System.out.println(getName() + " finished.");
19        }
20    }
21
22
23    public class Ques10 {
24        public static void main(String[] args) {
25
26            MyThread t1 = new MyThread("Thread 1 (Min Priority)", Thread.MIN_PRIORITY);
27            MyThread t2 = new MyThread("Thread 2 (Normal Priority)", Thread.NORMAL_PRIORITY);
28            MyThread t3 = new MyThread("Thread 3 (Max Priority)", Thread.MAX_PRIORITY);
29
30            t1.start();
31            t2.start();
32            t3.start();
33        }
34    }
35}

```

**Console**

```

<terminated> Ques10 (0) [Java Application] Eclipse JEE-2022
Thread 1 (Min Priority) is running with priority 1
Thread 2 (Normal Priority) is running with priority 5
Thread 3 (Max Priority) is running with priority 10

```

## 11. Write a Java program to create multiple threads and use thread group.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques11.java - Eclipse IDE
- Project Explorer:** Shows a project named "COREJAVA PROGRAMS375" with a "src" folder containing numerous Java files (Ques01.java through Ques30.java) and several package folders like "abstractionprograms", "datatypesprograms", etc.
- Code Editor:** Displays Java code for a program that demonstrates thread communication using `wait()` and `notify()`. It includes a `MyTask` class that extends `Thread` and a `main` method that creates a thread group and multiple threads within it.
- Console:** Shows the output from the last run: "terminated> Ques11 (12) [Java Application] C:\eclipse-jee-2022-06\Ques11\bin\Ques11.jar".
- Bottom Bar:** Shows the Windows taskbar with various pinned icons and the system clock indicating 2:00 AM on 3/18/2022.

## 12. Implement a program to demonstrate thread communication using wait() and notify() methods.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques12.java - Eclipse IDE
- Project Explorer:** Shows a project named "COREJAVA PROGRAMS375" with a "src" folder containing numerous Java files (Ques01.java through Ques30.java) and several package folders like "abstractionprograms", "datatypesprograms", etc.
- Code Editor:** Displays Java code for a producer-consumer problem using `wait()` and `notify()`. It features a `SharedResource` class with `produce` and `consume` synchronized methods, and a `main` method that creates threads to perform these actions.
- Console:** Shows the output from the last run: "terminated> Ques12 (8) [Java Application] C:\eclipse-jee-2022-06\Ques12\bin\Ques12.jar".
- Bottom Bar:** Shows the Windows taskbar with various pinned icons and the system clock indicating 19:53:682 on 3/18/2022.

## 13. Write a Java program to create multiple threads and use thread local variables

CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques13.java - Eclipse IDE

File Edit Source Refactor Search Project Run Window Help

Project Explorer X

Ques05java Ques06java Ques07java Ques08java Ques09java Ques10java Ques11java Ques12java Ques13java

> assignment1  
  ↳ basicsofjava  
  ↳ COREJAVAPROGRAMS375  
    ↳ IRE System Library [JavaSE-1]  
      ↳ src  
        ↳ abstractionprograms  
        ↳ arraysprograms  
        ↳ datatypesprograms  
        ↳ encapsulationprograms  
        ↳ exceptionhandling  
        ↳ inheritanceprograms  
        ↳ interfaceprograms  
        ↳ introductionprograms  
      ↳ multithreadingprograms  
        ↳ Ques01java  
        ↳ Ques02java  
        ↳ Ques03java  
        ↳ Ques04java  
        ↳ Ques05java  
        ↳ Ques06java  
        ↳ Ques07java  
        ↳ Ques08java  
        ↳ Ques09java  
        ↳ Ques10java  
        ↳ Ques11java  
        ↳ Ques12java  
        ↳ Ques13java  
        ↳ Ques14java  
        ↳ Ques15java  
        ↳ Ques16java  
        ↳ Ques17java  
        ↳ Ques18java  
        ↳ Ques19java  
        ↳ Ques20java  
        ↳ Ques21java  
        ↳ Ques22java  
        ↳ Ques23java  
        ↳ Ques24java  
        ↳ Ques25java  
        ↳ Ques26java  
      ↳ operatorsprograms  
      ↳ overloadingfunctions

Console X

<terminated> Ques13 [11] [Java Application] C:\eclipse-jee-2023-03\workspace\multithreadingprograms\Ques13\bin\Ques13

Thread 1 initial value: 0  
Thread 1 modified value: 100  
Thread 3 initial value: 0  
Thread 3 modified value: 100  
Thread 2 initial value: 0  
Thread 2 modified value: 100  
Thread 3 final value: 100  
Thread 1 final value: 100  
Thread 2 final value: 100

```
1 package multithreadingprograms;
2 //create multiple threads and use thread local variables
3 class MyTask1 extends Thread {
4     // ThreadLocal variable - each thread gets its own copy
5     private static ThreadLocal<Integer> threadLocalValue = ThreadLocal.withInitial(() -> 0
6
7     public MyTask1(String name) {
8         super(name);
9     }
10
11    @Override
12    public void run() {
13        System.out.println(getName() + " initial value: " + threadLocalValue.get());
14
15        // Modify the ThreadLocal value (only for this thread)
16        threadLocalValue.set(threadLocalValue.get() + 100);
17        System.out.println(getName() + " modified value: " + threadLocalValue.get());
18
19        // Simulate some work
20        try {
21            Thread.sleep(1000);
22        } catch (InterruptedException e) {
23            System.out.println(getName() + " was interrupted.");
24        }
25
26        // Final value (still unique per thread)
27        System.out.println(getName() + " final value: " + threadLocalValue.get());
28    }
29
30
31    public class Ques13 {
32        public static void main(String[] args) {
33            // Create and start multiple threads
34            MyTask1 t1 = new MyTask1("Thread 1");
35            MyTask1 t2 = new MyTask1("Thread 2");
36            MyTask1 t3 = new MyTask1("Thread 3");
37
38            t1.start();
39            t2.start();
40            t3.start();
41        }
42    }
}
```

Top Stories PM Modi joins Tr... Writable Smart Insert 1:1:0

Search

207 AM 3/18/2025

**14. Implement a program to demonstrate thread communication using volatile keyword.**

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer**: Shows the project structure under "CORE JAVA - COREJAVAPROGRAMS375". The "src" folder contains numerous Java files named Ques01.java through Ques26.java, as well as Abstractionprograms, Arrayprograms, Datatypesprograms, Encapsulationprograms, Exceptionhandling, Inheritanceprograms, Interfaceprograms, Introductionprograms, and Multithreadingprograms.
- Code Editor**: Displays the Java code for "Ques14.java". The code implements a producer-consumer pattern using the `volatile` keyword to ensure thread communication. It includes a producer method that sleeps for 1000ms and sets a flag to true, and a consumer method that waits until the flag is set to true before printing a message.
- Console**: Shows the output from the Java application, indicating that the producer has produced data and the consumer has consumed it.

```
package multithreadingprograms;
//thread communication using volatile keyword
class SharedResource3 {
    // Volatile keyword ensures visibility across threads
    private volatile boolean flag = false;

    public void producer() {
        System.out.println("Producer: Producing data...");
        try {
            Thread.sleep(1000); // Simulate some work
        } catch (InterruptedException e) {
            System.out.println("Producer interrupted.");
        }
        flag = true; // Set the flag to true, visible to
        System.out.println("Producer: Data produced, flag " + flag);
    }

    public void consumer() {
        System.out.println("Consumer: Waiting for data...");
        while (!flag) {
            // Busy-wait until flag is set to true by the producer
        }
        System.out.println("Consumer: Data consumed, flag " + flag);
    }
}

public class Ques14 {
    public static void main(String[] args) {
        SharedResource3 sharedResource = new SharedResource3();
        new Thread(sharedResource::producer).start();
        new Thread(sharedResource::consumer).start();
    }
}
```

## 15. Write a Java program to create multiple threads and use Executors framework

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows files like Ques07.java, Ques08.java, Ques09.java, Ques10.java, Ques11.java, Ques12.java, Ques13.java, Ques14.java, Ques15.java, and Ques16.java.
- Code Editor:** Displays the Java code for Ques15.java, which creates multiple threads using the Executors framework.
- Console Output:** Shows the execution of the program, indicating tasks running on different threads.
- Taskbar:** Shows the operating system taskbar with various application icons.

```
1 package multithreadingprograms;
2 //create multiple threads and use executors frameworks
3 import java.util.concurrent.ExecutorService;
4
5
6 class MyTask11 implements Runnable {
7     private final int taskId;
8
9     public MyTask11(int taskId) {
10         this.taskId = taskId;
11     }
12
13     @Override
14     public void run() {
15         System.out.println("Task " + taskId + " is running");
16         try {
17             Thread.sleep(1000); // Simulate some work
18         } catch (InterruptedException e) {
19             System.out.println("Task " + taskId + " was interrupted");
20         }
21         System.out.println("Task " + taskId + " has finished");
22     }
23
24
25 public class Ques15 {
26     public static void main(String[] args) {
27         // Create a fixed thread pool with 3 threads
28         ExecutorService executor = Executors.newFixedThrea
29 }
```

## 16. Implement a program to demonstrate thread interruption in Java.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows files like Ques07.java, Ques08.java, Ques09.java, Ques10.java, Ques11.java, Ques12.java, Ques13.java, Ques14.java, Ques15.java, Ques16.java, and Ques17.java.
- Code Editor:** Displays the Java code for Ques16.java, which demonstrates thread interruption.
- Console Output:** Shows the execution of the program, indicating the thread's state at various steps.
- Taskbar:** Shows the operating system taskbar with various application icons.

```
1 package multithreadingprograms;
2 //demonstration thread interruption in java
3 class MyTask22 extends Thread {
4     public MyTask22(String name) {
5         super(name);
6     }
7
8     @Override
9     public void run() {
10         System.out.println(getName() + " started.");
11
12         for (int i = 1; i <= 10; i++) {
13             if (Thread.interrupted()) {
14                 System.out.println(getName() + " was interrupted. Exiting...");
15                 return;
16             }
17             System.out.println(getName() + " is working... Step " + i);
18             try {
19                 Thread.sleep(500); // Simulate work
20             } catch (InterruptedException e) {
21                 System.out.println(getName() + " was interrupted during sleep. Exiting...");
22                 return;
23             }
24         }
25         System.out.println(getName() + " completed.");
26     }
27
28
29 public class Ques16 {
30     public static void main(String[] args) {
31         MyTask22 task = new MyTask22("Worker Thread");
32
33         // Start the thread
34         task.start();
35
36         // Give the thread some time to start
37         try {
38             Thread.sleep(1000);
39         } catch (InterruptedException e) {
40             e.printStackTrace();
41         }
42     }
43 }
```

## 17. Write a Java program to create multiple threads and use Callable and Future.

**CORE JAVA - COREJAVA PROGRAMS375/src/multithreadingprograms/Ques17.java - Eclipse IDE**

```

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X
CORE JAVA - COREJAVA PROGRAMS375
src
abstractionprograms
arraysprograms
datatypeprograms
encapsulationprograms
exceptionhandling
inheritanceprograms
interfaceprograms
introductionprograms
multithreadingprograms
Ques01java
Ques02java
Ques03java
Ques04java
Ques05java
Ques06java
Ques07java
Ques08java
Ques09java
Ques10java
Ques11java
Ques12java
Ques13java
Ques14java
Ques15java
Ques16java
Ques17java
Ques18java
Ques19java
Ques20java
Ques21java
Ques22java
Ques23java
Ques24java
Ques25java
Ques26java
operatorsprograms
overloadingprograms
overloadingnonname
Ques09.java Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java

1 package multithreadingprograms;
2 //create multiple threads and use callable and future
3 import java.util.concurrent.*;
4
5 class MyTask33 implements Callable<String> {
6     private final int taskId;
7
8     public MyTask33(int taskId) {
9         this.taskId = taskId;
10    }
11
12    @Override
13    public String call() throws Exception {
14        System.out.println("Task " + taskId + " is running on thread: " + Thread.currentThread());
15        Thread.sleep(1000); // Simulate some work
16        return "Task " + taskId + " completed.";
17    }
18}
19
public class Ques17 {
20    public static void main(String[] args) {
21        // Create a fixed thread pool
22        ExecutorService executor = Executors.newFixedThreadPool(3);
23
24        // Submit multiple tasks and collect futures
25        Future<String> future1 = executor.submit(new MyTask33(1));
26        Future<String> future2 = executor.submit(new MyTask33(2));
27        Future<String> future3 = executor.submit(new MyTask33(3));
28
29        try {
30            // Retrieve results from futures
31            System.out.println(future1.get());
32            System.out.println(future2.get());
33            System.out.println(future3.get());
34        } catch (InterruptedException | ExecutionException e) {
35            System.out.println("Exception: " + e.getMessage());
36        } finally {
37            executor.shutdown();
38        }
39    }
40}
41
42

```

Console X <terminated> Ques17 [12] [Java Application] [C:\eclipse-jee-202]
Task 1 is running on thread: pool-1-thread-1
Task 2 is running on thread: pool-1-thread-2
Task 3 is running on thread: pool-1-thread-3
Task 1 completed.
Task 2 completed.
Task 3 completed.

## 18. Implement a program to demonstrate thread communication using BlockingQueue

**CORE JAVA - COREJAVA PROGRAMS375/src/multithreadingprograms/Ques18.java - Eclipse IDE**

```

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X
CORE JAVA - COREJAVA PROGRAMS375
src
abstractionprograms
arraysprograms
datatypeprograms
encapsulationprograms
exceptionhandling
inheritanceprograms
interfaceprograms
introductionprograms
multithreadingprograms
Ques01java
Ques02java
Ques03java
Ques04java
Ques05java
Ques06java
Ques07java
Ques08java
Ques09java
Ques10java
Ques11java
Ques12java
Ques13java
Ques14java
Ques15java
Ques16java
Ques17java
Ques18java
Ques19java
Ques20java
Ques21java
Ques22java
Ques23java
Ques24java
Ques25java
Ques26java
operatorsprograms
overloadingprograms
overloadingnonname
Ques09.java Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java

1 package multithreadingprograms;
2
3 import java.util.concurrent.BlockingQueue;
4
5 //thread communication using blocking queue
6 class Producer implements Runnable {
7     private final BlockingQueue<String> queue;
8
9     public Producer(BlockingQueue<String> queue) {
10        this.queue = queue;
11    }
12
13    @Override
14    public void run() {
15        try {
16            for (int i = 1; i <= 5; i++) {
17                String item = "Item " + i;
18                System.out.println("Producing: " + item);
19                queue.put(item); // Adds item to the queue (blocks if full)
20                Thread.sleep(500); // Simulate work
21            }
22        } catch (InterruptedException e) {
23            System.out.println("Producer interrupted.");
24        }
25    }
26}
27
28 // Consumer Class
29 class Consumer implements Runnable {
30     private final BlockingQueue<String> queue;
31
32     public Consumer(BlockingQueue<String> queue) {
33        this.queue = queue;
34    }
35
36    @Override
37    public void run() {
38        try {
39            while (true) {
40                String item = queue.poll(2, TimeUnit.SECONDS); // Retrieves item (blocks if item == null) break; // Exit if no item in 2 seconds
41                if (item == null) break;
42                System.out.println("Consuming: " + item);
43            }
44        } catch (InterruptedException e) {
45            System.out.println("Consumer interrupted.");
46        }
47    }
48}
49
50

```

Console X <terminated> Ques17 [12] [Java Application] [C:\eclipse-jee-202]
Task 1 is running on thread: pool-1-thread-1
Task 2 is running on thread: pool-1-thread-2
Task 3 is running on thread: pool-1-thread-3
Task 1 completed.
Task 2 completed.
Task 3 completed.

## 19. Write a Java program to create multiple threads and use Phaser.

**CORE JAVA - COREJAVA PROGRAMS375/src/multithreadingprograms/Ques19.java - Eclipse IDE**

```

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer X
src
> assignment1
> basicsofjava
> COREJAVA PROGRAMS375
  > IRE System Library [JavaSE-1]
    > src
      > abstractionprograms
      > arraysprograms
      > datatypesprograms
      > encapsulationprograms
      > exceptionhandling
      > interfaceprograms
      > introductionprograms
      > multithreadingprograms
        > Ques01.java
        > Ques02.java
        > Ques03.java
        > Ques04.java
        > Ques05.java
        > Ques06.java
        > Ques07.java
        > Ques08.java
        > Ques09.java
        > Ques10.java
        > Ques11.java
        > Ques12.java
        > Ques13.java
        > Ques14.java
        > Ques15.java
        > Ques16.java
        > Ques17.java
        > Ques18.java
        > Ques19.java
        > Ques20.java
        > Ques21.java
        > Ques22.java
        > Ques23.java
        > Ques24.java
        > Ques25.java
        > Ques26.java
        > operatorsprograms
        > overloadingnonoverridding
Ques19.java Ques2.java Ques3.java Ques4.java Ques5.java Ques6.java Ques7.java Ques8.java Ques9.java Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java operatorsprograms overloadingnonoverridding
Console X
<terminated> Ques17 (12) [Java Application] Eclipse-jee-2022
Task 3 is running on thread:
pool-1-thread-3
Task 1 is running on thread:
pool-1-thread-1
Task 2 is running on thread:
pool-1-thread-2
Task 1 completed.
Task 2 completed.
Task 3 completed.

```

24°C Clear 2:09 AM 3/18/2025

## 20. Implement a program to demonstrate thread communication using CyclicBarrier.

**CORE JAVA - COREJAVA PROGRAMS375/src/multithreadingprograms/Ques20.java - Eclipse IDE**

```

File Edit Source Refactor Navigate Search Project Run Windows Help
Project Explorer X
src
> assignment1
> basicsofjava
> COREJAVA PROGRAMS375
  > IRE System Library [JavaSE-1]
    > src
      > abstractionprograms
      > arraysprograms
      > datatypesprograms
      > encapsulationprograms
      > exceptionhandling
      > interfaceprograms
      > introductionprograms
      > multithreadingprograms
        > Ques01.java
        > Ques02.java
        > Ques03.java
        > Ques04.java
        > Ques05.java
        > Ques06.java
        > Ques07.java
        > Ques08.java
        > Ques09.java
        > Ques10.java
        > Ques11.java
        > Ques12.java
        > Ques13.java
        > Ques14.java
        > Ques15.java
        > Ques16.java
        > Ques17.java
        > Ques18.java
        > Ques19.java
        > Ques20.java
        > Ques21.java
        > Ques22.java
        > Ques23.java
        > Ques24.java
        > Ques25.java
        > Ques26.java
        > operatorsprograms
        > overloadingnonoverridding
Ques19.java Ques2.java Ques3.java Ques4.java Ques5.java Ques6.java Ques7.java Ques8.java Ques9.java Ques10.java Ques11.java Ques12.java Ques13.java Ques14.java Ques15.java Ques16.java Ques17.java Ques18.java Ques19.java Ques20.java Ques21.java Ques22.java Ques23.java Ques24.java Ques25.java Ques26.java operatorsprograms overloadingnonoverridding
Console X
<terminated> Ques20 (9) [Java Application] Eclipse-jee-2022
Thread-2 is working...
Thread-3 is working...
Thread-1 is working...
The threads reached the barrier.
Thread-2 reached the barrier.
Thread-1 reached the barrier.
All threads reached the barrier.
Proceeding further...
Thread-1 crossed the barrier.
Thread-2 crossed the barrier.
Thread-3 crossed the barrier.

```

24°C Clear 2:10 AM 3/18/2025

## 21. Write a Java program to create multiple threads and use Semaphore.

**Ques13.java**

```

1 package multithreadingprograms;
2
3 import java.util.concurrent.Semaphore;
4
5 // Shared Resource
6 class SharedResource55 {
7     private final Semaphore semaphore;
8
9     public SharedResource55(int permits) {
10         this.semaphore = new Semaphore(permits);
11     }
12
13     public void accessResource(String threadName) {
14         try {
15             System.out.println(threadName + " is waiting for a permit...");
16             semaphore.acquire(); // Acquires a permit (blocks if none available)
17             System.out.println(threadName + " acquired a permit. Accessing resource...");
18             Thread.sleep(1000); // Simulate work
19             System.out.println(threadName + " is done. Releasing permit...");
20         } catch (InterruptedException e) {
21             System.out.println(threadName + " was interrupted.");
22         } finally {
23             semaphore.release(); // Releases the permit
24         }
25     }
26
27     // Worker Thread
28     class Worker1 extends Thread {
29         private final SharedResource resource;
30
31         public Worker1(SharedResource resource, String name) {
32             super(name);
33             this.resource = resource;
34         }
35
36         @Override
37         public void run() {
38             resource.accessResource(getName());
39         }
40     }
41 }
42

```

## 22. Implement a program to demonstrate thread communication using Exchanger.

**Ques22.java**

```

1 package multithreadingprograms;
2
3 import java.util.concurrent.Exchanger;
4
5 // Worker Class
6 class Worker22 extends Thread {
7     private Exchanger<String> exchanger;
8     private String message;
9
10    public Worker22(Exchanger<String> exchanger, String message) {
11        this.exchanger = exchanger;
12        this.message = message;
13    }
14
15    @Override
16    public void run() {
17        try {
18            System.out.println(getName() + " sending: " + message);
19            // Exchange message with the other thread
20            String response = exchanger.exchange(message);
21            System.out.println(getName() + " received: " + response);
22        } catch (InterruptedException e) {
23            System.out.println(getName() + " was interrupted.");
24        }
25    }
26
27    // Main Class
28    public class Ques22 {
29        public static void main(String[] args) {
30            // Create an Exchanger object
31            Exchanger<String> exchanger = new Exchanger<>();
32
33            // Create two threads to exchange messages
34            Thread t1 = new Worker22(exchanger, "Message from Thread-1");
35            Thread t2 = new Worker22(exchanger, "Message from Thread-2");
36
37            t1.start();
38            t2.start();
39        }
40    }
41 }
42

```

## 23. Write a Java program to create multiple threads and use CompletionService.

CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques23.java - Eclipse IDE

File Edit Source Refactor Search Project Run Window Help

Project Explorer X

Ques15java Ques16java Ques18java Ques17java Ques19java Ques20java Ques21java Ques22java Ques23java x4

Console X

<terminated> Ques23 [11] [Java Application] C:\eclipse-ee-2023-03\workspace\multithreadingprograms\Ques23\bin\Ques23

Task 3 is running...  
Task 1 is running...  
Task 2 is running...  
Task 4 is running...  
Task 3 completed  
Task 5 is running...  
Task 1 completed  
Task 2 completed  
Task 4 completed  
Task 5 completed

```
1 package multithreadingprograms;
2 //create multiple threads and use completion service
3 import java.util.concurrent.*;
4
5 // Worker Callable
6 class Worker66 implements Callable<String> {
7     private final int taskId;
8
9     public Worker66(int taskId) {
10         this.taskId = taskId;
11     }
12
13     @Override
14     public String call() throws Exception {
15         String message = "Task " + taskId + " is running...";
16         System.out.println(message);
17         Thread.sleep(1000); // Simulate work
18         return "Task " + taskId + " completed";
19     }
20 }
21
22 // Main Class
23 public class Ques23 {
24     public static void main(String[] args) {
25         int numTasks = 5;
26         ExecutorService executor = Executors.newFixedThreadPool(3);
27         CompletionService<String> completionService = new ExecutorCompletionService<(exec
28
29         // Submit multiple tasks
30         for (int i = 1; i <= numTasks; i++) {
31             completionService.submit(new Worker66(i));
32         }
33
34         // Collect results as they complete
35         for (int i = 0; i < numTasks; i++) {
36             try {
37                 Future<String> result = completionService.take();
38                 System.out.println(result.get());
39             } catch (InterruptedException | ExecutionException e) {
40                 System.out.println("Error: " + e.getMessage());
41             }
42         }
43     }
44 }
```

Writable Smart Insert 1:1:0

24°C Clear

Search

File Icons

211 AM 3/18/2025

**24. Implement a program to demonstrate thread communication using TransferQueue.**

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques24.java - Eclipse IDE
- File Menu:** File Edit Source Refactor Navigate Project Run Window Help
- Project Explorer:** Shows the project structure under COREJAVAPROGRAMS375, including src, JRE System Library [JSE:1], and various packages like assignment1, basicsofjava, etc.
- Code Editor:** Displays Ques24.java containing Java code for multithreading using TransferQueue. The code defines two classes: Producer77 and Consumer77, each implementing Runnable. The Producer class has a constructor taking a TransferQueue and a taskId, and a run method that prints a message and uses the queue's transfer method. The Consumer class also has a constructor taking a TransferQueue and a run method that uses the queue's take method to consume messages.
- Console:** Shows the output of the program, indicating tasks being produced and consumed by different threads.

```
Ques24.java
package multithreadingprograms;
//thread communication using transfer Queue
import java.util.concurrent.*;


// Producer Thread
class Producer77 implements Runnable {
    private final TransferQueue<String> queue;
    private final int taskId;

    public Producer77(TransferQueue<String> queue, int taskId) {
        this.queue = queue;
        this.taskId = taskId;
    }

    @Override
    public void run() {
        try {
            String message = "Task " + taskId + " produced";
            System.out.println(Thread.currentThread().getName() + " producing: " + message);
            queue.transfer(message); // Transfer message to consumer
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

// Consumer Thread
class Consumer77 implements Runnable {
    private final TransferQueue<String> queue;

    public Consumer77(TransferQueue<String> queue) {
        this.queue = queue;
    }

    @Override
    public void run() {
        try {
            while (true) {
                String message = queue.take(); // Wait and consume message
                System.out.println(Thread.currentThread().getName() + " consumed: " + mess
            }
        } catch (InterruptedException e) {
    }
}
```

Console Output:

```
pool-1-thread-3 producing: Task 1 produced
pool-1-thread-5 producing: Task 3 produced
pool-1-thread-3 producing: Task 4 produced
Consuming: Task 1 produced
pool-1-thread-4 producing: Task 2 produced
Consuming: Task 3 produced
pool-1-thread-5 producing: Task 5 produced
Consuming: Task 4 produced
Consuming: Task 2 produced
Consuming: Task 5 produced
```

**25. Write a Java program to create multiple threads and use Scheduled Executor Service.**

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques25.java - Eclipse IDE
- Project Explorer:** Shows the project structure with files like assignment1, basicjava, and COREJAVAPROGRAMS375.
- Code Editor:** Displays the Java code for Ques25.java. The code uses a ScheduledExecutorService to run multiple tasks. Task 1 is executed by pool-1-thread-2 at 1742244123147. Tasks 2 through 10 are also listed, each with a unique timestamp.
- Console:** Shows the output of the application's execution, including the shutdown message "Shutting down scheduler...".
- Bottom Status Bar:** Shows the date and time as 12:1 3:16, and the system temperature as 24°C.

```
package multithreadingprograms;
// create multiple threads and use scheduled executor service
import java.util.concurrent.*;


// Task to be scheduled
class ScheduledTask implements Runnable {
    private final int taskId;

    public ScheduledTask(int taskId) {
        this.taskId = taskId;
    }

    @Override
    public void run() {
        System.out.println("Task " + taskId + " executed by " + Thread.currentThread().get
    }
}

// Main Class
public class Ques25 {
    public static void main(String[] args) {
        ScheduledExecutorService scheduler = Executors.newScheduledThreadPool(3);

        // Schedule multiple tasks
        for (int i = 1; i < 5; i++) {
            scheduler.scheduleAtFixedRate(new ScheduledTask(i), 0, 2, TimeUnit.SECONDS);
        }

        // Let the tasks run for 10 seconds before shutting down
        scheduler.schedule(() -> {
            System.out.println("Shutting down scheduler...");
            scheduler.shutdown();
        }, 10, TimeUnit.SECONDS);
    }
}
```

```
<terminated> Ques25 (Java Application) [eclipse-jee-2022-06]
Task 1 executed by pool-1-thread-2
at 1742244123147
Task 2 executed by pool-1-thread-2
at 1742244123147
Task 3 executed by pool-1-thread-2
at 1742244123147
Task 4 executed by pool-1-thread-2
at 1742244123147
Task 5 executed by pool-1-thread-2
at 1742244123147
Task 2 executed by pool-1-thread-1
at 1742244123147
Task 2 executed by pool-1-thread-2
at 1742244123147
Task 2 executed by pool-1-thread-3
at 1742244123147
Task 3 executed by pool-1-thread-2
at 1742244123147
Task 3 executed by pool-1-thread-3
at 1742244123147
Task 4 executed by pool-1-thread-2
at 1742244123147
Task 4 executed by pool-1-thread-3
at 1742244123147
Task 1 executed by pool-1-thread-1
at 1742244127139
Task 3 executed by pool-1-thread-1
at 1742244127139
Task 5 executed by pool-1-thread-1
at 1742244127139
Task 2 executed by pool-1-thread-1
at 1742244127139
Task 4 executed by pool-1-thread-1
at 1742244127139
Task 1 executed by pool-1-thread-2
at 1742244127139
Task 2 executed by pool-1-thread-2
at 1742244127139
Task 3 executed by pool-1-thread-2
at 1742244127139
Task 4 executed by pool-1-thread-2
at 1742244127139
Task 1 executed by pool-1-thread-3
at 1742244127139
Task 5 executed by pool-1-thread-3
at 1742244127139
Task 2 executed by pool-1-thread-3
at 1742244127139
Shutting down scheduler...
```

**26. Implement a program to demonstrate thread communication using Lock and Condition.**

CORE JAVA - COREJAVAPROGRAMS375/src/multithreadingprograms/Ques26.java - Eclipse IDE

File Edit Source Refactor Navigate Project Run Window Help

```
1 package multithreadingprograms;
2 import java.util.concurrent.locks.*;
3
4 class SharedResource90 {
5     private int data;
6     private boolean hasData = false;
7     private final Lock lock = new ReentrantLock();
8     private final Condition condition = lock.newCondition();
9
10    public void produce() throws InterruptedException {
11        lock.lock();
12        try {
13            data = (int) (Math.random() * 100);
14            System.out.println("Produced: " + data);
15            hasData = true;
16            condition.signal();
17        } finally {
18            lock.unlock();
19        }
20    }
21
22    public void consume() throws InterruptedException {
23        lock.lock();
24        try {
25            while (!hasData) {
26                condition.await();
27            }
28            System.out.println("Consumed: " + data);
29            hasData = false;
30        } finally {
31            lock.unlock();
32        }
33    }
34
35    public class Ques26 {
36        public static void main(String[] args) {
37            SharedResource shareResource = new SharedResource();
38            Thread producer = new Thread(() -> {
39                try {
40                    for (int i = 0; i < 5; i++) {
41                        shareResource.produce();
42                        Thread.sleep(1000);
43                    }
44                } catch (InterruptedException e) {}
45            });
46            Thread consumer = new Thread(() -> {
47                try {
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
368
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
838
839
839
840
841
842
843
844
845
846
847
847
848
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
918
919
919
920
921
922
923
924
925
926
927
927
928
928
929
929
930
931
932
933
934
935
936
937
938
938
939
939
940
941
942
943
944
945
946
947
947
948
948
949
949
950
951
952
953
954
955
956
957
958
958
959
959
960
961
962
963
964
965
966
967
967
968
968
969
969
970
971
972
973
974
975
976
977
977
978
978
979
979
980
981
982
983
984
985
986
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2167
2168
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
```

**Overall Questions :**

**Introduction** - 25

**Operators** - 25

**Arrays** - 25

**Data Types** - 25

**Strings** - 25

**Overloading** - 25

**Encapsulation** - 28

**Static** - 25

**Abstraction** - 34

**Inheritance** - 32

**Exception Handling** - 25

**Interfaces** - 32

**Multi-Threading** - 26

---

**Total** **375**

