Explore | Expand | Enrich

<Codemithra />™

Test time on basic input,output,operators

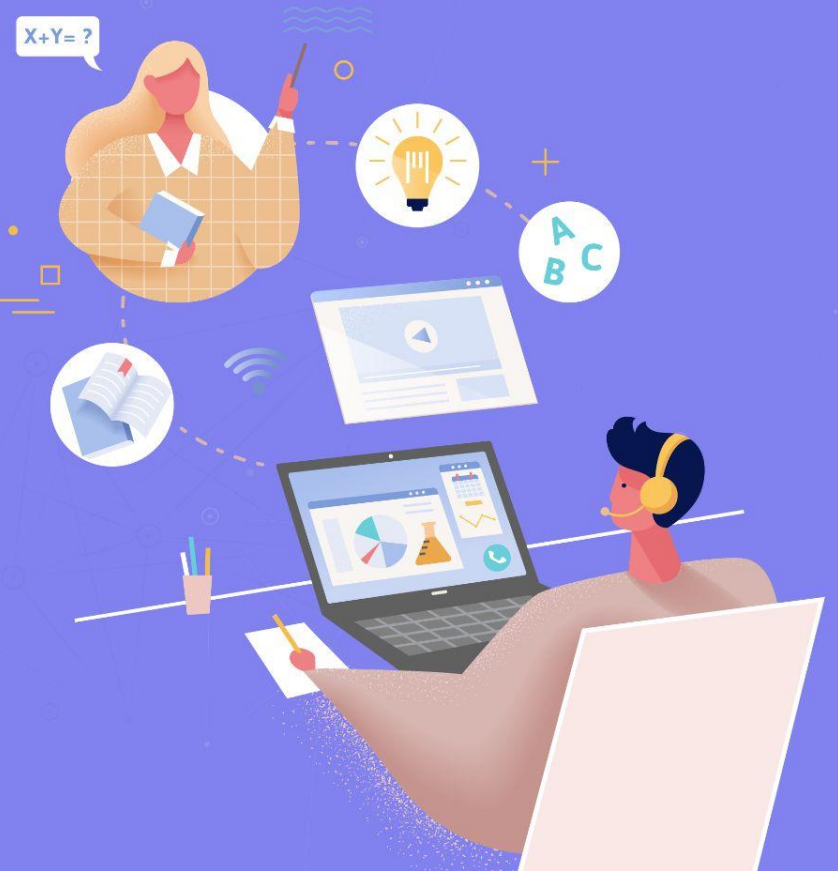
URL:https://forms.gle/vLep53Dn1cNGn6ks9

QR CODE:

<Codemithra />™

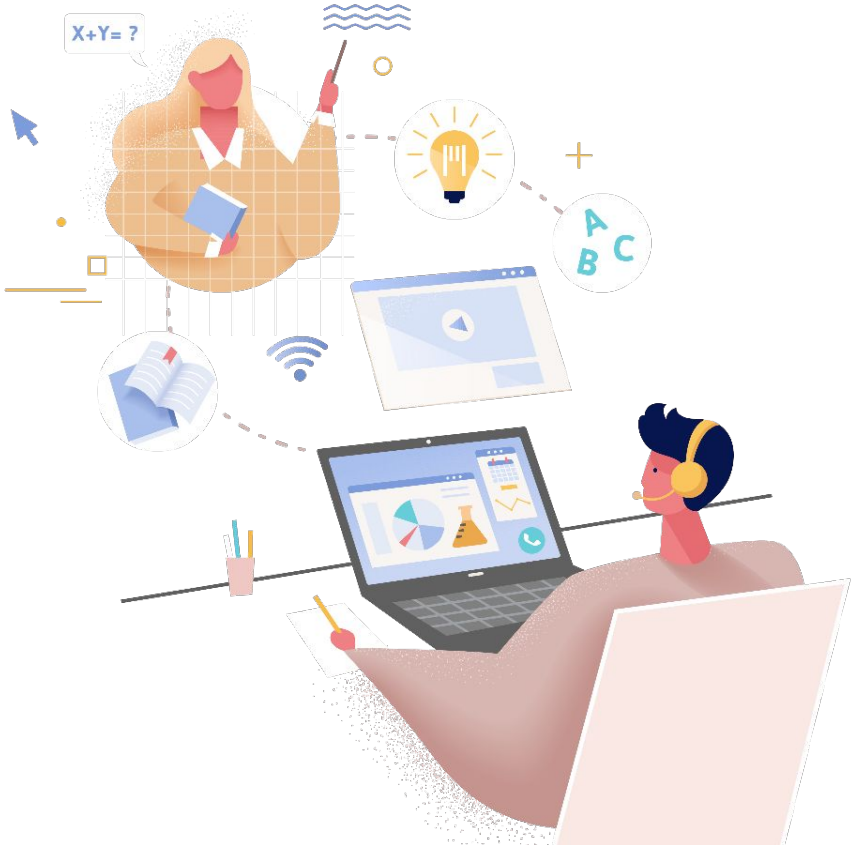
ETHNUS
Explore | Expand | Enrich

DECISION MAKING AND CONTROL STATEMENT



TOPICS

- What is decision making?
- If
- If-else
- If-else If-else statement
- Switch
- Ternary operator
- Jump statements



What is decision-making?

Decision-making statements are used to control the flow of a program based on certain conditions. These statements allow you to make decisions and execute different blocks of code depending on whether a given condition is true or false.

1. if statement: The "if" statement is the fundamental decision-making statement. It evaluates a boolean expression inside the parentheses and executes the block of code within the curly braces if the condition is true.

```
if (condition) {
   // Code to be executed if the condition is true
}
```

Example

```
public class Main {
    public static void main(String[] args) {
        int number = 10;
        if (number > 5) {
            System.out.println("The number is greater than 5.");
        }
    }
}
```

2.if-else statement: The "if-else" statement provides an alternative block of code to be executed when the condition is false.

```
if (condition) {
    // Code to be executed if the condition is true
} else {
    // Code to be executed if the condition is false
}
```

```
public class Main {
    public static void main(String[] args) {
        int number = 3;
        if (number % 2 == 0) {
            System.out.println("The number is even.");
        } else {
            System.out.println("The number is odd.");
        }
    }
}
```

3. if-else if-else statement: The "if-else if-else" statement allows you to chain multiple conditions together and execute different blocks of code based on the first condition that evaluates to true.

```
if (condition1) {
    // Code to be executed if condition1 is true
} else if (condition2) {
    // Code to be executed if condition 2 is true
} else {
    // Code to be executed if all previous conditions are false
}
```

```
public class Main {
    public static void main(String[] args) {
        int score = 85;
        if (score >= 90) {
            System.out.println("A");
        } else if (score >= 80) {
            System.out.println("B");
        } else if (score >= 70) {
            System.out.println("C");
        } else {
            System.out.println("D");
        }}}
```

4. switch statement: The "switch" statement provides an alternative way to handle multiple conditions based on the value of an expression. It allows you to choose a specific block of code to execute based on different possible values.

```
switch (expression) {
   case value1:
      // Code to be executed if the expression equals value1
      break;
   case value2:
      // Code to be executed if the expression equals value2
      break;
   // More cases can be added here
   default:
      // Code to be executed if none of the cases match the expression
}
```

```
public class Main {
    public static void main(String[] args)
{
        int dayOfWeek = 2;
        String dayName;
        switch (dayOfWeek) {
            case 1:
                dayName = "Sunday";
                break;
            case 2:
                dayName = "Monday";
                break;
            case 3:
                dayName = "Tuesday";
                break;
            case 4:
                dayName = "Wednesday";
                break;
            case 5:
                dayName = "Thursday";
                break;
            case 6:
                dayName = "Friday";
                break;
            case 7:
                dayName = "Saturday";
                break;
            default:
                dayName = "Invalid day";
                break;
        }
        System.out.println("The day is: "
+ dayName);
    }
}
```

decision making statements

Ternary operator (?:):

It is a shorthand way to write simple if-else statements.

```
int num = 10;
String result = (num > 0) ? "Positive" : "Non-positive";
System.out.println(result);
```

decision making statements-Jump statements

There are three jump statements:

- Break
- Continue
- Return

These statements are used to control the flow of a program and are typically used in loops and conditional blocks.

decision making statements-Jump statements

1. break: The `break` statement is used to exit a loop prematurely, even if the loop condition is not met. When the `break` statement is encountered, the control flow exits the loop, and the program continues with the statement after the loop.

Example

```
public class Main {
   public static void main(String[] args) {
      for (int i = 1; i <= 10; i++) {
         if (i == 5) {
            break; // Exit the loop when i becomes 5
         }
         System.out.println("Value: " + i);
      }
   }
}
```

decision making statements-Jump statements

2. continue: The `continue` statement is used to skip the rest of the current iteration and continue with the next iteration of a loop. When the `continue` statement is encountered, the control flow jumps back to the loop's beginning to evaluate the loop condition again.

Example

```
public class Main {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            if (i == 3) {
                    continue; // Skip iteration when i is 3 and continue with the next
iteration
            }
            System.out.println("Value: " + i);
        }
    }
}
```

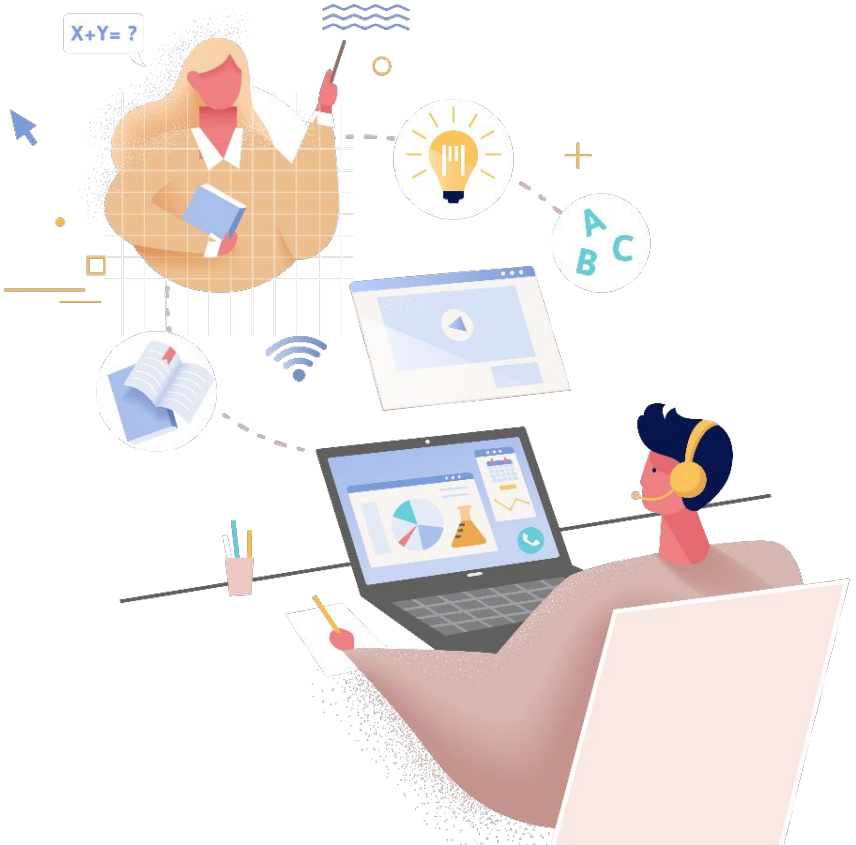
decision making statements-Jump statements

3.return: The `return` statement is used inside a method to terminate the method's execution and optionally return a value to the caller. When a `return` statement is encountered, the method's execution stops, and the control flow returns to the calling method.

Example

```
public class Main {
    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;
        int result = add(num1, num2);
        System.out.println("Sum: " + result);
    }
    public static int add(int a, int b) {
        int sum = a + b;
        return sum; // Return the sum to the caller
    }
}
```

INTERVIEW QUESTIONS



Interview questions

1. What is control flow in Java, and why is it important in programming?

Answer: Control flow refers to the order in which statements in a program are executed. It’s essential for defining the logic and sequence of operations in a program.

Interview questions

2. What are the primary categories of control flow in Java?

Answer: The primary categories of control flow in Java are conditional statements, loops, and method calls.

Interview questions

3. Explain the ‘if’ statement in Java and its syntax.

Answer: The ‘if’ statement is used for conditional execution. Syntax:

```
if (condition) {
    // Code to execute if condition is true
}
```

Interview questions

4. What is the purpose of the ‘else’ clause in an ‘if’ statement?

Answer: The ‘else’ clause is used to provide an alternative code block that executes when the ‘if’ condition is false.

Interview questions

5. What is the ‘else if’ statement in Java, and how is it used for multiple conditions?

Answer: The ‘else if’ statement allows you to specify multiple conditions in a chain, with each condition tested in sequence until one is true.

Interview questions

6. Explain the ‘switch’ statement in Java and its syntax.

Answer: The ‘switch’ statement is used to evaluate a variable against a list of values.

Syntax:

```
switch (variable) {
    case value1:
        // Code for value1
        break;
    case value2:
        // Code for value2
        break;
    default:
```

Practice questions

Question 1. Leap Year Checker:

Write a program to determine whether a given year is a leap year or not. A leap year is divisible by 4 but not by 100 unless it is also divisible by 400.

Sample Test Case:

Input: 2024

Output: Leap year

Input: 2100

Output: Not a leap year

Practice questions

Question 2. Grade Calculator:

Create a program that takes a student's score as input and outputs their corresponding grade based on the following criteria:

A: 90-100
B: 80-89
C: 70-79
D: 60-69
F: Below 60

Sample Test Case:
Input: 85
Output: B

Practice questions

Question 3. Factorial Calculator:

Write a program to calculate the factorial of a non-negative integer input by the user.

Sample Test Case:

Input: 5

Output: 120 (5! = 5 x 4 x 3 x 2 x 1 = 120)

Practice questions

Question 4. Number Guessing Game:

Implement a number guessing game where the computer generates a random number between 1 and 100, and the user has to guess it. Provide feedback to the user if their guess is too high or too low.

Sample Test Case:

Random number: 50

User's Guess: 30

Output: Too low. Try again.

User's Guess: 70

Output: Too high. Try again.

User's Guess: 50

Output: Congratulations! You guessed it right.

Practice questions

Question 5. Fibonacci Sequence Generator:

Write a program to generate the Fibonacci sequence up to a specified number of terms provided by the user.

Sample Test Case:

Input: 8

Output: 0 1 1 2 3 5 8 13

<Codemithra />™

eTHNUS™
Explore | Expand | Enrich

THANK YOU

+91 78150 95095 | codemithra@ethnus.com | www.codemithra.com