DAY-3

How to Take Input From User in Java?

There are mainly two ways to take input from the user.

1. Using BufferedReader

BufferedReader is simple class which is used to read a character or a sequence of character. It has simple functions like read() which reads a character and another read() which reads a array of character and readLine() which reads a line.

The BufferedReader class is used to read text from an input stream efficiently. It can be wrapped around other input streams to buffer the input and improve efficiency.

BufferedReader throws a checked exception. Exceptions are some advanced condition that happen at runtime. You must code to handle that condition.

Example:

```
import java.io.*;

class Test {

    // Main Method
    public static void main(String[] args)
        throws IOException
    {
        // Creating BufferedReader Object
        // InputStreamReader converts bytes to stream of character
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        // String reading internally
        String str = br.readLine();

        // Integer reading internally
        int it = Integer.parseInt(br.readLine());

        // Printing String
        System.out.println("Entered String : " + str);

        // Printing Integer
        System.out.println("Entered Integer : " + it);
    }
}
```

Explanation of Example:

- `import java.io.*;` imports the necessary classes for input/output operations.
- `class Test` defines a class named `Test`.
- `public static void main(String[] args) throws IOException` is the entry point of the program. It can throw an `IOException` if an input/output error occurs.
- `BufferedReader br = new BufferedReader(new InputStreamReader(System.in));` creates a `BufferedReader` object. The `InputStreamReader` converts byte streams from the input (like keyboard input) into character streams.
- `String str = br.readLine();` reads a line of text from the user and stores it in the `str` variable.
- `int it = Integer.parseInt(br.readLine());` reads another line of text, converts it to an integer, and stores it in the `it` variable.
- `System.out.println("Entered String : " + str);` prints the string entered by the user.
- `System.out.println("Entered Integer : " + it);` prints the integer entered by the user.

2. Using the Scanner Class

Scanner is the advanced method which is used to reading input from the users. Scanner can read formatted input. It has different function for different types of data types.

The Scanner class is part of the java.util package and is used to obtain input of primitive types like int, double, etc., and strings. It is simple and convenient for most input-related tasks.

Example:

```
import java.util.Scanner;

public class InputExample {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);  // Create a Scanner object

      System.out.print("Enter your name: ");
      String name = scanner.nextLine();  // Read user input as a string

      System.out.print("Enter your age: ");
      int age = scanner.nextInt();  // Read user input as an integer

      System.out.print("Enter your weight: ");
```

```
        float weight = scanner.nextFloat();  // Read user input as an Floating point value


        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Weight: " + weight);

        scanner.close();  // Close the scanner
    }
}
```

Explanation of Example:

- import java.util.Scanner; - This imports the Scanner class from the java.util package.
- Scanner scanner = new Scanner(System.in); - Creates a new Scanner object to read from standard input (keyboard).
- scanner.nextLine(); - Reads a line of text from the user.
- scanner.nextInt(); - Reads an integer value from the user.
- scanner.close(); - Closes the Scanner object to free up resources.

Decision making statements provide clients/customers to choose from many

Decision-making statements in programming allow clients or customers to choose from many options by guiding the program to execute different code paths based on certain conditions. Here are the common decision-making statements in programming.

1. If Statement

The `if` statement evaluates a condition. If the condition is true, it executes a block of code.

Example:

```
int age = 18;
if (age >= 18) {
    System.out.println("You are eligible to vote.");
}
```

2. Nested If Statement

A nested `if` statement is an `if` statement inside another `if` statement. It allows for more complex decision-making by evaluating multiple conditions in a hierarchical manner.

Example:

```
int age = 20;
boolean hasVoterID = true;
```

```
if (age >= 18) {
    if (hasVoterID) {
        System.out.println("You can vote.");
    }
}
```

3. If-Else Statement

The `if-else` statement provides an alternative block of code to execute if the condition is false.

Example:

```
int age = 16;
if (age >= 18) {
    System.out.println("You are eligible to vote.");
} else {
    System.out.println("You are not eligible to vote.");
}
```

4. Switch Statement

The `switch` statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Example:

```
int day = 3;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    default:
        System.out.println("Invalid day");
}
```

5. If-Else-If Ladder Statement

The `if-else-if` ladder allows multiple conditions to be checked in sequence. As soon as one condition is true, the corresponding block of code is executed, and the rest are skipped.

Example:

```
int marks = 85;
if (marks >= 90) {
```

```
    System.out.println("Grade A");
} else if (marks >= 80) {
    System.out.println("Grade B");
} else if (marks >= 70) {
    System.out.println("Grade C");
} else {
    System.out.println("Grade D");
}
```

6. Nested If-Else Statement

A nested `if-else` statement is an `if-else` statement inside another `if-else` statement. This structure allows for more detailed and complex decision-making.

Example:

```
int age = 20;
boolean hasVoterID = false;

if (age >= 18) {
    if (hasVoterID) {
        System.out.println("You can vote.");
    } else {
        System.out.println("You need a voter ID to vote.");
    }
} else {
    System.out.println("You are not eligible to vote.");
}
```