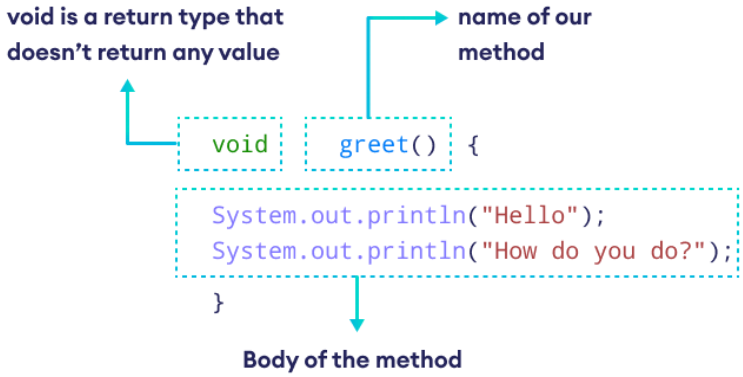
CodInClub

Powered By BridgeLabz

Programming Concept Practiced

1. **Java Methods -** Methods are used to organize our code. Methods help us divide our program into smaller chunks to make it easier to use and understand.
   a. Methods help to reduce and reorganize the complexity of the program. As a result, we can focus on only a small part of the problem at one time.
   b. A method is a verb-like function that performs specific actions so takes in some input and gives a response



   Here are the different parts of the Java Methods
   a. **Method Name -** Every method has a Name. In this case ***greet***. Note the name indicates the work the method is going to perform like greet with *"Hello, How do you do"*
   b. **methodName() with Paraenthesis *( )*** - A method name ends with parentheses. An optional parameters can be added as input to the method. In the case of ***greet***, its a method with empty parameters which means no input is needed to perform action.
   c. **Method Body *{ }*** - Its the body of the method which has code to perform actions indicated with the curly brackets ***{ }***. In the case of ***greet***, the body outputs Hello, How do you do"
   d. **Return Type -** The method represents the type of data returned by the method. In the case of ***greet***, its return type is ***void*** which means method doesn't return any value.

2. **Running *greet()* Method -** The following code when run sees no output. Because the ***greet*** method is not called or invoked

```
class Main {
    void greet() {
        System.out.println("Hello, How do you do?");
    }
    public static void main(String[] args) {
    }
}
```

3. **Calling Method in Java -** In Java, we use an object of the class to call the method, so first, we need to create an object of the class.

```
Main obj = new Main();
```

   a. NOTE: We haven't discussed Objects or Classes, our objective is method calling alone.
   b. So we create the object and call its method as in

```
Main obj = new Main();
obj.greet();
```

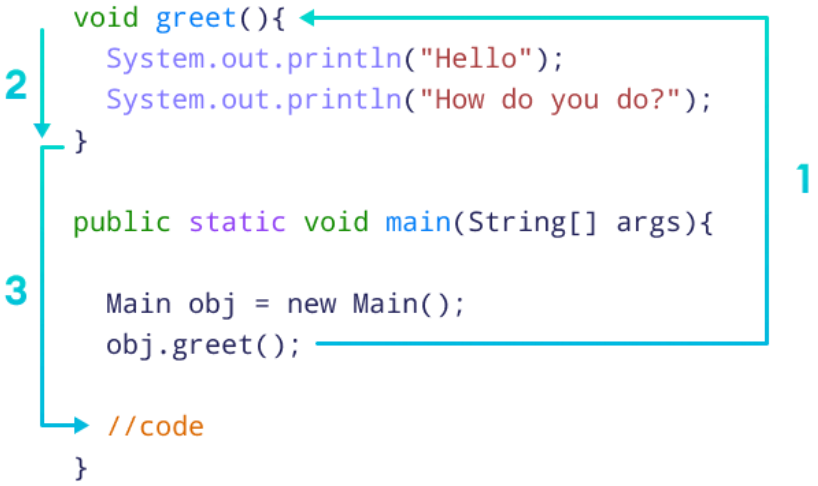
```java
class Main {

    void greet() {
        System.out.println("Hello");
        System.out.println("How do you do?");
    }

    public static void main(String[] args) {
        // create object of Main
        Main obj = new Main();

        // call the method
        obj.greet();
    }
}
```

   c. **Working of Code -** Here, we are using the object (obj) along with the dot operator (.) to call the method



a. **Step 1:** When a method is called, the program's control jumps to the method definition. In this case, the ***greet()*** method is called
b. **Step 2:** Then, the statements inside the ***greet()*** method are executed.
c. **Step 3:** After all the statements are executed, the program's control jumps back to the ***main()*** method and moves to the next line of code after the ***greet()*** method call.

```java
class Main {

    void greet() {
        System.out.println("Hello");
        System.out.println("How do you do?");
    }

    public static void main(String[] args) {
        // create object of Main
        Main obj = new Main();

        // call the method
        obj.greet();
    }
}
```

4. **Using a Method Multiple Times -** One of the benefits of writing methods is to write once and use it many times in different parts of the program
5. **Making Methods Dynamic -** It is possible to create methods that take input and return output. In programming terms:
    a. **Method Arguments or Method Parameters:** Adding arguments or parameters to methods means adding inputs to the method. We can add 0, 1, or many parameters to the method as per the input needed to do the work
    b. **Java *return* keyword -** Using the ***return*** keyword method can return the results essentially returning the output of the method
6. **Benefits of using Methods -**
    a. A method is an independent piece of **working code**. Write once and use it everywhere.
    b. The method can be **reused** multiple times in different parts of the code.
    c. Writing Methods make the program more **modular** and less error-prone.

    The Following Program crates a ***UnitConverter*** class with a method ***convertKmToMiles*** which takes in kilometer as a parameter and returns miles

Java

```java
// Write a Program to convert from one unit to another unit and convert
// kilometer to miles

public class UnitConverter {

    // Method To convert kilometers to miles and return the value
    public double convertKmToMiles(double km) {
        // Convert km to miles
        double km2miles = 0.621371;
        double miles = km * km2miles;

        // return the value
        return miles;
    }

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

        // Take input for km
        System.out.print("Enter the distance in kilometers: ");
        double km = sc.nextDouble();

        // Create a UnitConverter object
        UnitConverter unitConverter = new UnitConverter();

        // Call the method to convert km to miles
        double miles = unitConverter.convertKmToMiles(km);

        // Display value in miles
        System.out.println("Distance in miles: " + miles);

        // Calling the Method again to convert km to miles and display it
        System.out.print("Enter the distance in kilometers: ");
        km = sc.nextDouble();
        miles = unitConverter.convertKmToMiles(km);
        System.out.println("Distance in miles: " + miles);

        // Close the Scanner object
        sc.close();
    }
}
```

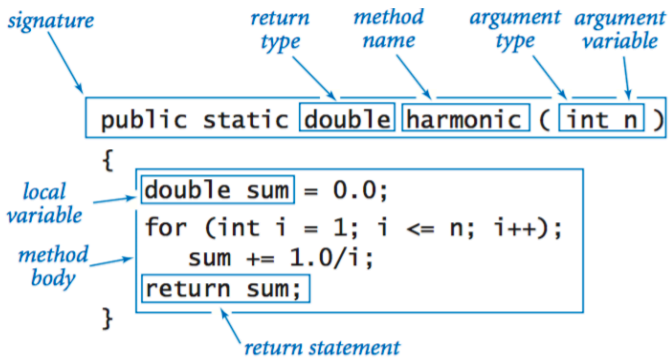
7. **User-Defined Methods -** The methods defined so far are written by us the user and hence are called User Defined Methods. They have a return type, method name, method parameters, and method body.

8. **Recursive Methods -** Recursive Methods are methods that call themselves in order to solve a problem.

```java
public class RecursiveExample {
    // Recursive Method to calculate factorial
    public static int factorial(int n) {
        if (n == 0) { // Base case: factorial of 0 is 1
            return 1;
        } else {
            return n * factorial(n - 1); // Recursive case
        }
    }

    public static void main(String[] args) {
        int number = 5;
        System.out.println("Factorial of " + number + " is: " +
                           factorial(number));
    }
}
```

9. **Java Static Method -** A static method in Java is a method that belongs to the class rather than any specific object. You can call it directly using the class name without creating an object of the class. Key Characteristics of Static Methods
    a. **Belongs to the Class:** Declared with the ***static*** keyword, it can be accessed using the class name.
    b. **Cannot Access Instance Variables:** Since static methods are not tied to a specific object, they can only access other static members (variables or methods) directly.
    c. **Useful for Utility Methods:** Commonly used for utility or helper functions.



Java

```java
// Write a Program to convert from one unit to another unit as in convert
// kilometer to miles

public class UnitConverter {

    // Method To convert kilometers to miles and return the value
    public static double convertKmToMiles(double km) {
        // Convert km to miles
        double km2miles = 0.621371;
        double miles = km * km2miles;

        // return the value
        return miles;
    }

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

        // Take input for km
        System.out.print("Enter the distance in kilometers: ");
        double km = sc.nextDouble();

        // Call the method to convert km to miles
        double miles = UnitConverter.convertKmToMiles(km);

        // Display value in miles
        System.out.println("Distance in miles: " + miles);

        // Close the Scanner object
        sc.close();
    }
}
```

10. **Java Standard Library Methods -** The Java Standard Library provides a vast collection of pre-defined classes and methods organized into packages. These methods simplify common programming tasks like string manipulation, mathematical operations, and working with collections. Commonly Used Java Standard Library Methods
    a. ***java.lang*** Package – This package is automatically imported in every Java program. Commonly used Methods are from ***Math*** and ***String*** class. We have so far used ***System.out.println("")*** which is a ***System*** class defined in java.lang package.

Method	Description
`Math.abs(int a)`	Returns the absolute value of `a`.
`Math.pow(double a, double b)`	Returns `a` raised to the power of `b`.
`Math.sqrt(double a)`	Returns the square root of `a`.
`String.toUpperCase()`	Converts all characters in a string to uppercase.
`String.trim()`	Removes whitespace from both ends of a string.
`Integer.parseInt(String s)`	Converts a string to an integer.

    b. ***java.util*** Package – This package provides utility classes like ***Scanner***, ***ArrayList***, and ***HashMap***. We have so far used the Scanner class to take user input.

11. ***java.lang.Math*** **class –** For all the Mathematical operations are defined inside the Math class. Most commonly used methods of Math class are
    a. ***Math.sqrt()*** – The ***sqrt()*** method is used to find the square root of a number.

```
class Main {
   public static void main(String[] args) {
      // find the square root
      int number = 25;
      double squareRoot = Math.sqrt(number);
      System.out.println("Square root of " + number + " is " + squareRoot);
   }
}
```

    b. ***Math.pow()*** – The ***pow()*** is used to find the power of a number. It takes two arguments (base value and power value) and returns the power raised to the base.

```
class Main {
   public static void main(String[] args) {
      // find the power of a number whose base is 2 and power is 3 i.e. 2^3
      int base = 2, power = 3;
      double result = Math.pow(base, power);
      System.out.println("The result is " + result);
   }
}
```

c. ***Math.random()*** – The ***random()*** method to get a random number from 0.0 to less than 1.0.

d. ***Math.random()*** between 0.0 and 5.0 – We can modify this program to generate random numbers from 0.0 to less than 5.0. by multiplying the random number by 5 as in
`double randomNumber = Math.random() * 5;`

e. Integer ***Math.random()*** between 1.0 and 10.0 – We can get an Integer random number between 0.0 and 10.0 by multiplying the random result by 10 and then adding 1. Finally type casting to ***int*** as in
`int randomNumber = (int) (Math.random() * 10 + 1);`

Java

```
class Main {
   public static void main(String[] args) {
      // random number between 0.0 and 1.0
      double randomNumber1 = Math.random();
      double randomNumber2 = Math.random();
      System.out.println("The random numbers are " + randomNumber1 +
                         " and " + randomNumber2);

      // random number 0.0 to 4.999
      double randomNumber3 = Math.random() * 5;
      System.out.println("The random number between 0 & 5.0 is “ +
                        randomNumber3);

      // integer random number from 1 to 10
      int randomNumber4 = (int) (Math.random() * 10 + 1);
      System.out.println("The integer random number between 0 & 10 is “ +
                        randomNumber4);
   }
}
```

Best Programming Practice

1. All values as variables including Fixed, User Inputs, and Results
2. Proper naming conventions for all variables
3. Proper Program Name and Class Name
4. Proper Method Name which indicates action taking inputs and providing result

Sample Program 1: Create a program to find the sum of all the digits of a number given by a user using an array and display the sum.

a. Use Math.random() and get a 4-digit random integer number
b. Write a method to count digits in the number
c. Write a method to return an array of digits from a given number.
d. Write a method to Find the sum of the digits of the number in the array
e. Finally, display the sum of the digits of the number

Java

```java
// Create SumOfDigit Class to compute the sum of 4 digits random number
class SumOfDigits {
   // Get a 4 digit random number
   public int get4DigitRandomNumber() {
      return (int) (Math.random() * 9000) + 1000;
   }

   // Find the count of digits in the number
   public int countDigits(int number) {
      int count = 0, temp = number;
      while (temp > 0) {
         count++;
         temp /= 10;
      }
      return count;
   }

   // Store the digits of the number in an array
   public int[] getDigits(int number, int count) {
      int[] digits = new int[count];
      int temp = number;
      for (int i = count - 1; i >= 0; i--) {
         digits[i] = temp % 10;
         temp /= 10;
      }
      return digits;
   }
```

```
    // Find the sum of the elements in an array
    public int sumArray(int[] array) {
        int sum = 0;
        for (int i = 0; i < array.length; i++) {
            sum += array[i];
        }
        return sum;
    }

    public static void main(String[] args) {
        // Get 4 digit random integer number
        SumOfDigits sumOfDigits = new SumOfDigits();
        int number = sumOfDigits.get4DigitRandomNumber();
        System.out.println("The Random Mumber is: " + number);

        // Get the count of digits
        int count = sumOfDigits.countDigits(number);
        System.out.println("The count of digit is: " + count);

        // Get the array of digits from the number
        int[] digits = sumOfDigits.getDigits(number, count);

        // Find the sum of the digits of the number
        int sum = sumOfDigits.sumArray(digits);

        // Display the sum of the digits of the number
        System.out.println("\nSum of Digits: " + sum);
    }
}
```