- The for-each loop in Java (also called the enhanced for loop) was introduced in Java 5 to simplify iteration over arrays and collections. It is cleaner and more readable than the traditional for loop and is commonly used when the exact index of an element is not required.
- Below is a basic example of using the for-each loop to iterate through an array and print each element. A for-each loop directly accesses elements without needing index variables.

```
// Using for-each loop
import java.io.*;
class ForEach
{
    public static void main(String[] args)
    {
        // Array declaration
        int arr[] = { 1, 2, 3, 4, 5 };

        // Using for-each loop to
        // print each element
        for (int e : arr)
        {
            System.out.print(e + " ");
        }//End of foreach
    }//End of main
}//End of ForEach class
```

Sum of Digits of a Number

```
public class SumOfDigits {
    public static void main(String[] args) {
        int number = 1234;
        int sum = 0;
        while (number > 0) {
            sum += number % 10; // Add last digit
            number /= 10;       // Remove last digit
        }
        System.out.println("Sum of digits: " + sum);
    }
}
```

Reverse a Number

```
public class ReverseNumber {
    public static void main(String[] args) {
        int number = 1234, reversed = 0;
        while (number > 0) {
            int digit = number % 10; // Extract the last digit
            reversed = reversed * 10 + digit; // Add it to reversed number
            number /= 10; // Remove the last digit
        }
        System.out.println("Reversed Number: " + reversed);
    }
}
```

Print a Multiplication Table

```
public class MultiplicationTable {
    public static void main(String[] args) {
        int number = 5;
        for (int i = 1; i <= 10; i++) {
            System.out.println(number + " x " + i + " = " + (number *i));
        }
    }
}
```

Check if a Number is Prime

```
public class PrimeCheck {
  public static void main(String[] args) {
        int number = 29;
        boolean isPrime = true;
        for (int i = 2; i <= number / 2; i++) {
            if (number % i == 0) {
                isPrime = false;
                break;
            }
        }
        if (isPrime) {
            System.out.println(number + " is a prime number.");
        } else {
            System.out.println(number + " is not a prime number.");
        }
    }
}
```

Right-Angle Triangle Pattern

```
import java.util.Scanner;

public class RightAngleTriangle {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows (n): ");
        int n = sc.nextInt();

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Pyramid Pattern

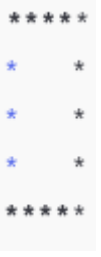
```
import java.util.Scanner;
public class PyramidPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows (n): ");
        int n = sc.nextInt();

        for (int i = 1; i <= n; i++) {
            // Print spaces
            for (int j = 1; j <= n - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int j = 1; j <= 2 * i - 1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Diamond Pattern

```
import java.util.Scanner;
public class DiamondPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows (n):
");
        int n = sc.nextInt();
        // Upper part of the diamond
        for (int i = 1; i <= n; i++) {
            // Print spaces
            for (int j = 1; j <= n - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int j = 1; j <= 2 * i - 1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
```

```
/ Lower part of the diamond
        for (int i = n - 1; i >= 1; i--) {
            // Print spaces
            for (int j = 1; j <= n - i; j++) {
                System.out.print(" ");
            }
            // Print stars
            for (int j = 1; j <= 2 * i - 1; j+
+) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Hollow Square of Stars



Zig-Zag Pattern