Integrierte Entwicklungsumgebung:

Verwenden Sie diese Website: <https://www.programiz.com/java-programming/online-compiler/>

Es ist keine Installation erforderlich, öffnen Sie einfach den Link und verwenden Sie die Software.

Schreiben Sie den Code auf der linken Seite und klicken Sie auf die Schaltfläche „Run“, um ihn auszuführen.

Alternative: https://www.w3schools.com/java/tryjava.asp?filename=demo_compiler

Aufgabe 1 - Java Grundlagen

Aufgabe 1: FizzBuzz

Frage. Schreiben Sie ein Java-Programm, das die Zahlen von 1 bis 100 ausgibt. Aber für Vielfache von drei, gebe „Fizz“ anstelle der Zahl aus, und für Vielfache von fünf, gebe „Buzz“ aus. Für Zahlen, die sowohl ein Vielfaches von drei als auch von fünf sind, gebe „FizzBuzz“ aus.

Erwartete Ausgabe.

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
```

...

Lösung.

```
for (int i = 1; i <= 100; i++) {
    if (i % 3 == 0 && i % 5 == 0) {
        System.out.println("FizzBuzz");
```

```

}
else if (i % 3 == 0) {
    System.out.println("Fizz");
}
else if (i % 5 == 0) {
    System.out.println("Buzz");
}
else {
    System.out.println(i);
}
}
}

```

Aufgabe 2: Approximiere π

Frage. Schreiben Sie ein Java-Programm, das eine Näherung von π mit Hilfe der Leibniz-Reihe berechnet:

$$\pi \approx 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots \right)$$

Ihre Aufgabe besteht darin:

1. Definieren Sie eine feste ganzzahlige Variable n , die die Anzahl der in die Annäherung einzubeziehenden Terme angibt (z. B. $n = 1000$).
2. Verwenden Sie eine Schleife, um die Summe der ersten n Terme der Reihe zu berechnen.
3. Geben Sie den ungefähren Wert von π nach der Summierung von n Termen aus.

Lösung.

```

public class Main {
    public static void main(String[] args) {
        int n = 10000;
        double piApproximation = 0.0;
        boolean addTerm = true; // To alternate between + and -

        // Loop to calculate the approximation of pi
        for (int i = 0; i < n; i++) {
            double term = 1.0 / (2 * i + 1); // Calculate the current term (1/1, 1/3, 1/5, ...)

            // Alternate between adding and subtracting terms
            if (addTerm) {
                piApproximation += term;
            } else {
                piApproximation -= term;
            }
        }
    }
}

```

```
    addTerm = !addTerm;
  }

  piApproximation *= 4;

  // Print the result
  System.out.println("Approximation of pi: " + piApproximation);
}
}
```