Loops In JAVA

Day 17
27/07/2025

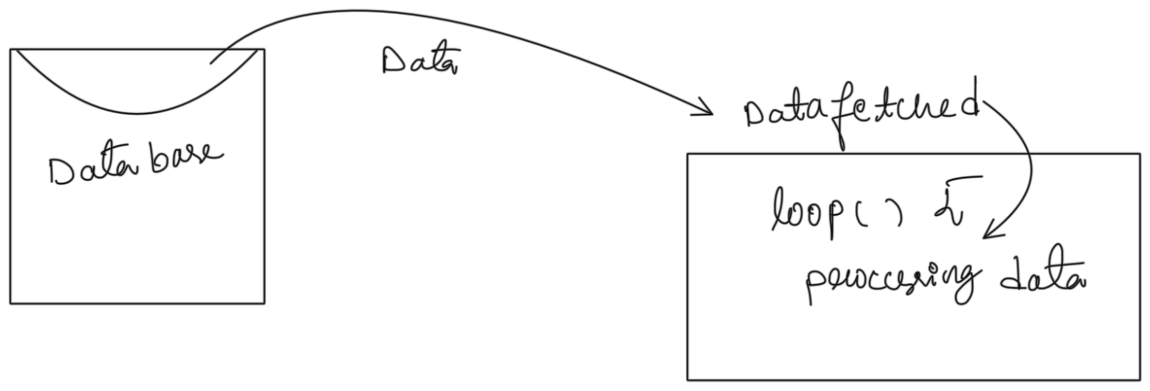
Why We Need Loops ?
Where they are used in Industry ?
Examples of Loops ?
Types of Loops and difference between them ?
- for
- while
- Do While

Nested loops
Trade-offs between for & while loop
Pattern printing
Multiplication table demo

Logical statements, Loops, Functions, Expressions are the Most Important Topics In the programming

combinations of Loops & arrays Can form thousands of Questions.

In the software Industry Loops are used more for proccessing the Data which is fetched.



Most of the sorting techniques are using Loops

The Loops plays an important Role in Analyzing Time Complexity, code optimization

Loops

1. Loops have Starting point
2. stopping point
3. Change Increment or decrement

$$\begin{tikzcd} & \text{Loops} \arrow[dl] \arrow[d] \arrow[dr] & \\ \text{start} & \text{End Stop} & \text{Change} \end{tikzcd}$$

$\longrightarrow \quad \downarrow \; \uparrow$

$\longleftarrow$

Increment or decrement

styles of Writting Loops

i. Small $\longrightarrow$ Big

$1 \xrightarrow{\quad + \quad} 100$

2. Big $\longrightarrow$ Small

$100 \xrightarrow{\quad - \quad} 1$

i. # set or collection of items

# processing

$$\begin{tikzcd} & & \text{processing} \arrow[dll, leftrightarrow] \arrow[dl, leftrightarrow] \arrow[d] \arrow[dr] & \\ \text{Searching} & \text{Math} & \text{sorting} & \text{filter} \end{tikzcd}$$

Whenever there is a collection of items there is a Loop

② # generation

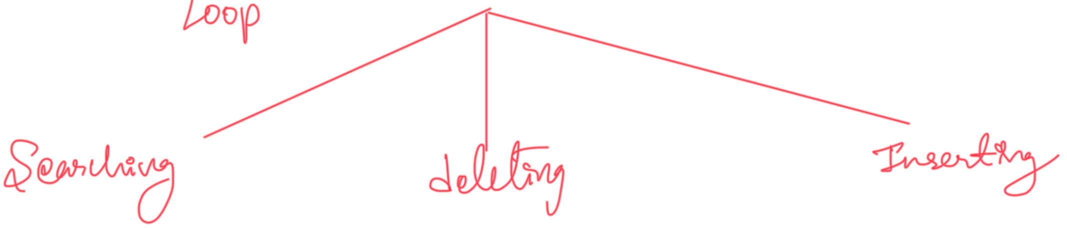
Whenever there is a generation we use loops

generating any Number Sequence.

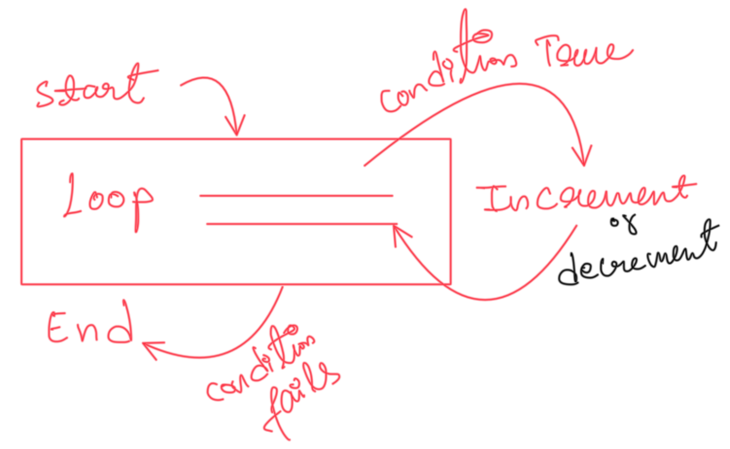
generating any type of Mathematic or Logical Sequence.

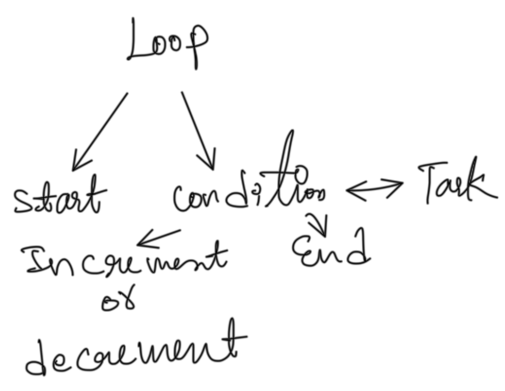
③ String (string is a collection of characters)

So some operation on string requires a Loop



Homogenous collections (same elements) can be traversed & accessed. (using loops)





for Loop

Syntax for (start ; condition ; updation)

Note key of loop updation

Example :

$\text{for (}\underbrace{\text{int index = 0}}_{\text{start}}\text{ ; }\underbrace{\text{index <= 10}}_{\text{condition}}\text{ ; i = i+1) \{}$

$\boxed{\text{printf ("\%d", index)}}$ Task or function

}

Task is performed untill condition fails.

We can start Loop at any point & End at any point By adjusting start & condition & updation.

Example

```
for ( int i=10 ; i ==20 ; i+=5) {
        perform task ;
}
```

We can write Any Numbers of Conditions In the Loop So for Loop have All this possibilities.

While Loop

while Loop only check condition & performs updation Inside its Code Block

The starting point decided Before while Loop is written & the Ending is decided By the while Loop's Condition.

start

We use While Loop

```
while (condition) {
        ______
        ______
        ______
      updation
 }
```

When we dont know How many Numbers of times loop should Execute So we use the condition to Break the loop

Example "Binary Search"
"Traversing on Linked List"

```
do {
     [          ]
} while ()
         ↳ condition
```

do while Loop is also called as Exit control loop.

In do while loop first Iteration Last condition check.

do while Loop Executes atleast one time No matter condition is true or false.

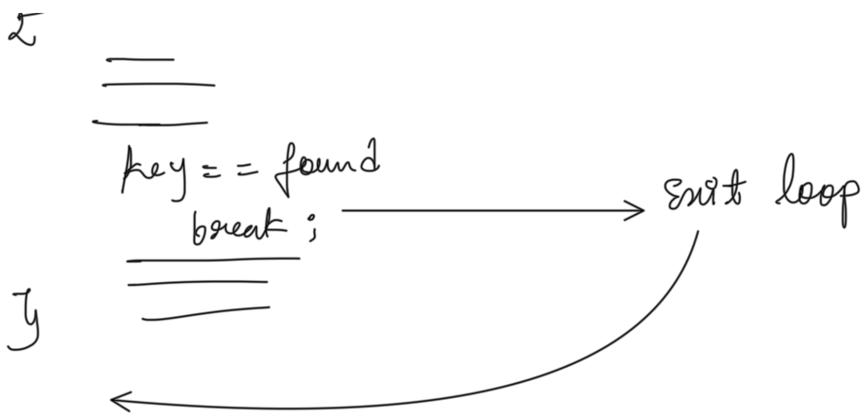
---

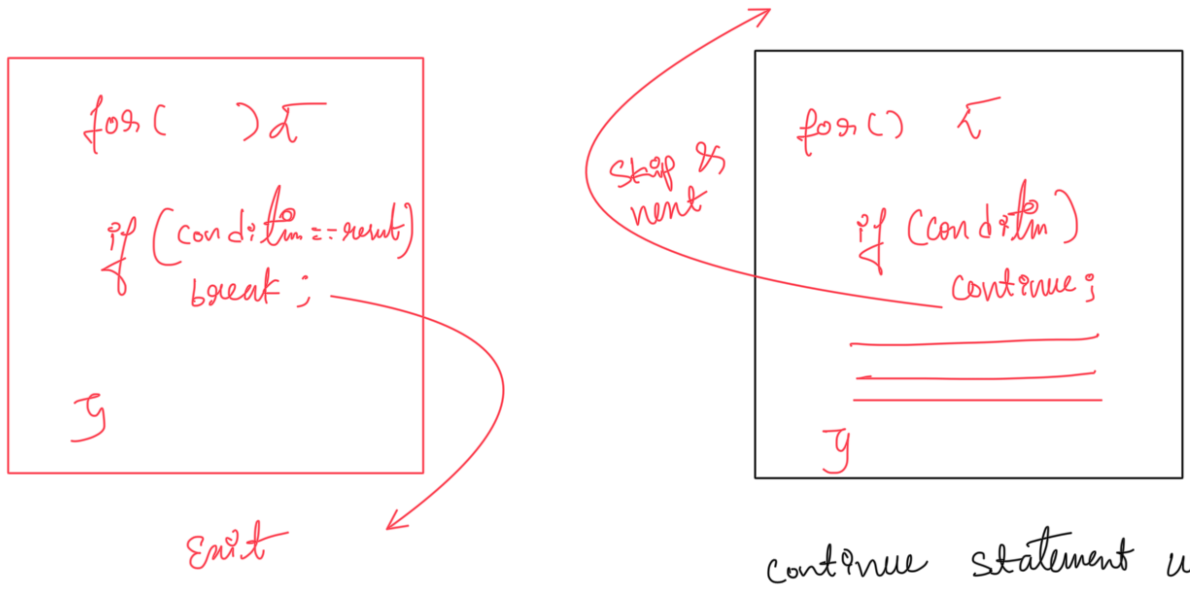
loop Control statements

break statement
continue statement] This statement have control of loop Execution and stopping the execution and skipping current Execution.

break statement
↳ when we get our required result we Just break the loop and get the result for proccessing. No Need of Executing the Loop further.

for (   )





continue statement used to skip the current Execution of loop and moves control to next Iteration.

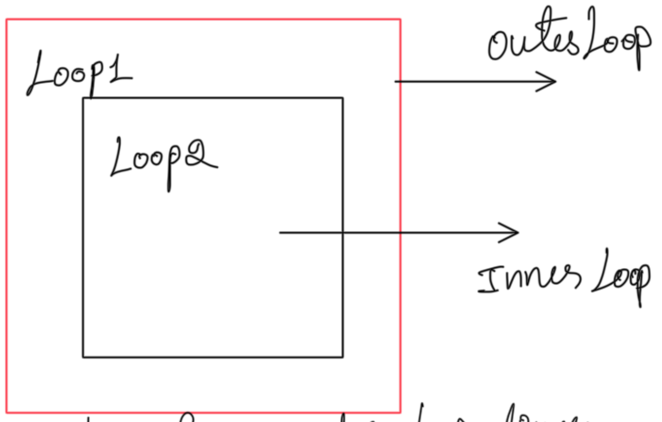
Nested Looping   (usually Nested Loops have $O(n^2)$ Complexity

Running a Loop Inside a Loop or

Loping a Loop

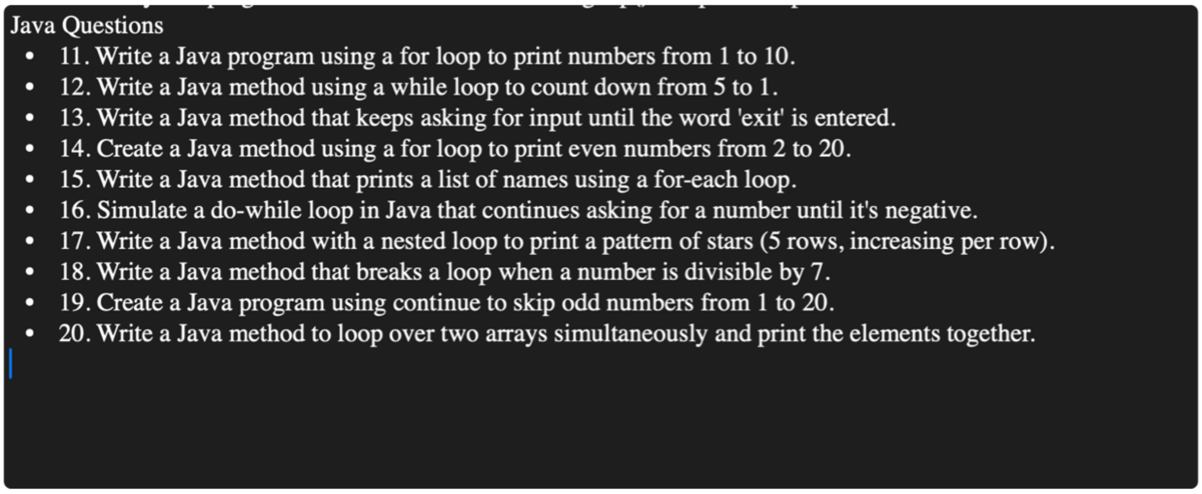
Time Complexity
$O(M \times N)$

$\Rightarrow O(N^2)$



Inner Loop Runs outer Loop times

Basic Looping Questions



Java Questions

- 11. Write a Java program using a for loop to print numbers from 1 to 10.
- 12. Write a Java method using a while loop to count down from 5 to 1.
- 13. Write a Java method that keeps asking for input until the word 'exit' is entered.
- 14. Create a Java method using a for loop to print even numbers from 2 to 20.
- 15. Write a Java method that prints a list of names using a for-each loop.
- 16. Simulate a do-while loop in Java that continues asking for a number until it's negative.
- 17. Write a Java method with a nested loop to print a pattern of stars (5 rows, increasing per row).
- 18. Write a Java method that breaks a loop when a number is divisible by 7.
- 19. Create a Java program using continue to skip odd numbers from 1 to 20.
- 20. Write a Java method to loop over two arrays simultaneously and print the elements together.

Understanding Loops in Java

1 Why Do We Need Loops?

Loops allow us to repeat a set of instructions multiple times without manually duplicating code.

Without a loop:

```java
System.out.println(1);
System.out.println(2);
System.out.println(3);
```

With a loop:

```java
for (int i = 1; i <= 3; i++) {
    System.out.println(i);
}
```

2 Where Are Loops Used in Industry?

Industry Area	**Usage of Loops**
Web Development	Rendering dynamic lists (e.g. user profiles, posts)
Game Development	Running game frames and user events
Finance	Calculating interest, EMIs, processing transactions
Data Processing	Iterating over rows in files, databases
Automation Testing	Repeating test cases across inputs
Machine Learning	Running iterations during model training

3 Types of Loops in Java

- **For Loop** – Used when the number of iterations is known.
- **While Loop** – Runs based on a condition.

- **Do-While Loop** – Executes once before checking condition.
- **For-Each Loop** – Iterates over arrays and collections.

4 Difference Between Loop Types

Feature	**For Loop**	**While Loop**	**Do-While Loop**
Entry Condition	Yes	Yes	No
Initialization	In loop header	Outside loop	Outside loop
Best Use Case	Counted loops	Condition-based loops	Run at least once

5 Nested Loops

Loops within loops. Useful in grids, matrix operations, and patterns.

```
for(int i = 1; i <= 3; i++) {
    for(int j = 1; j <= 2; j++) {
        System.out.println(i + " " + j);
    }
}
```

6 For vs While – Tradeoffs

Factor	**For Loop**	**While Loop**
Readability	Best for counted loops	Best for indefinite loops
Control Location	All-in-one header	Spread across lines
Infinite Loop Risk	Lower	Higher

7 Java Loop Examples

11. Print 1 to 10 using for loop

```
public void printOneToTen() {
    for (int i = 1; i <= 10; i++) {
        System.out.println(i);
    }
}
```

12. Countdown from 5 to 1 using while loop

```java
public void countDown() {
    int i = 5;
    while (i >= 1) {
        System.out.println(i);
        i--;
    }
}
```

13. Ask for input until 'exit'

```java
public void keepAsking() {
    Scanner sc = new Scanner(System.in);
    String input;
    while (true) {
        System.out.print("Enter input: ");
        input = sc.nextLine();
        if (input.equalsIgnoreCase("exit")) break;
    }
}
```

14. Print even numbers from 2 to 20

```java
public void printEvens() {
    for (int i = 2; i <= 20; i += 2) {
        System.out.println(i);
    }
}
```

15. Print names using for-each loop

```java
public void printNames() {
    String[] names = {"Alice", "Bob", "Charlie"};
    for (String name : names) {
        System.out.println(name);
    }
}
```

16. Simulate do-while until number is negative

```java
public void askUntilNegative() {
    Scanner sc = new Scanner(System.in);
    int num;
    do {
        System.out.print("Enter number: ");
        num = sc.nextInt();
    } while (num >= 0);
}
```

17. Star pattern using nested loops

```
public void printStars() {
    for (int i = 1; i <= 5; i++) {
        for (int j = 1; j <= i; j++) {
            System.out.print("* ");
        }
        System.out.println();
    }
}
```

18. Break when number divisible by 7

```
public void breakOnSeven() {
    for (int i = 1; i <= 20; i++) {
        if (i % 7 == 0) {
            System.out.println("Divisible by 7: " + i);
            break;
        }
    }
}
```

19. Skip odd numbers using continue

```
public void skipOdds() {
    for (int i = 1; i <= 20; i++) {
        if (i % 2 != 0) continue;
        System.out.println(i);
    }
}
```

20. Loop over two arrays simultaneously

```
public void loopTwoArrays() {
    String[] names = {"Alice", "Bob", "Charlie"};
    int[] scores = {90, 80, 85};
    for (int i = 0; i < names.length; i++) {
        System.out.println(names[i] + " scored " + scores[i]);
    }
}
```

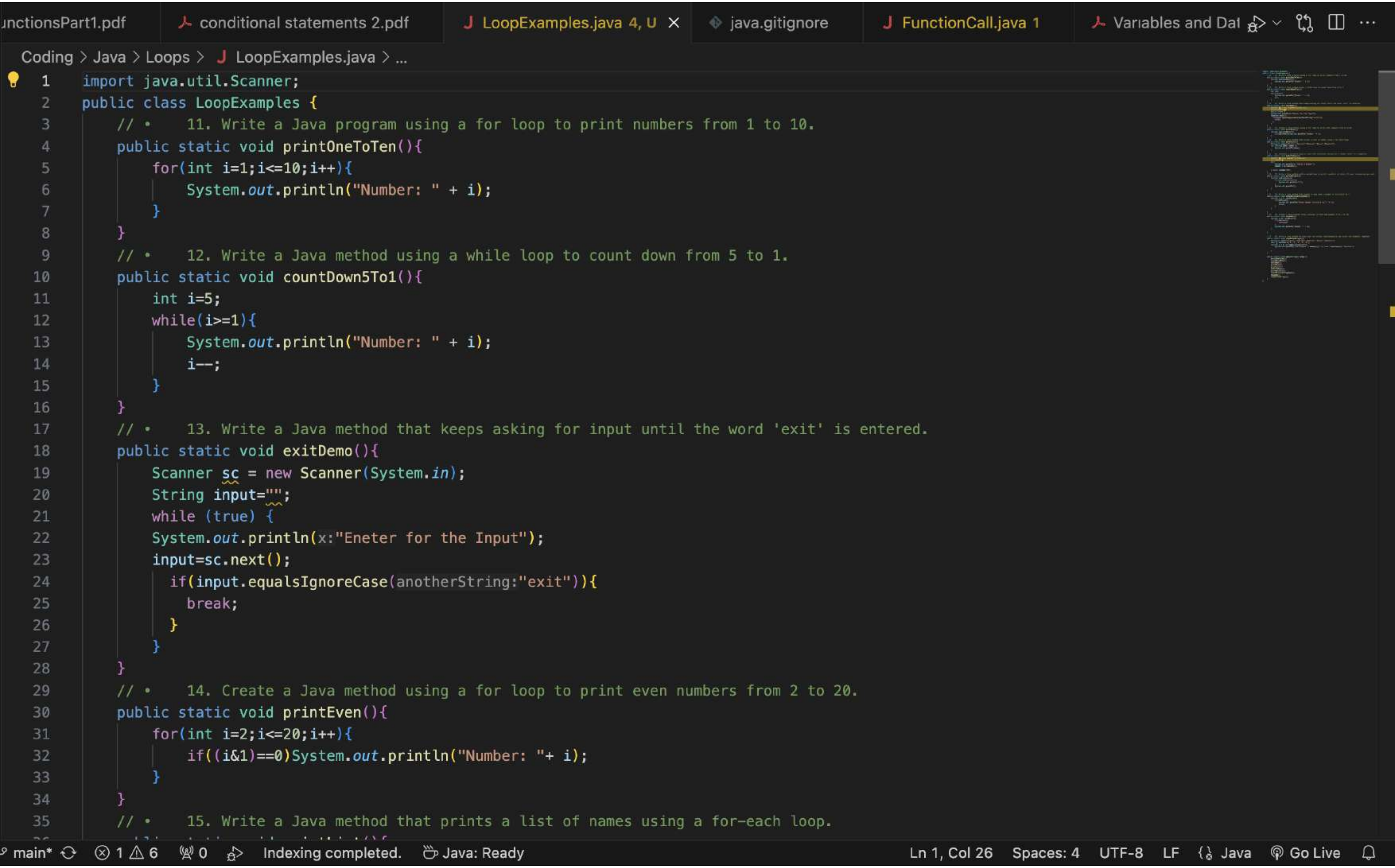
8 Bonus Examples

Multiplication Table Using Nested Loops

```
public void multiplicationTable() {
    for (int i = 1; i <= 10; i++) {
        for (int j = 1; j <= 10; j++) {
            System.out.print((i * j) + "\t");
        }
        System.out.println();
    }
}
```

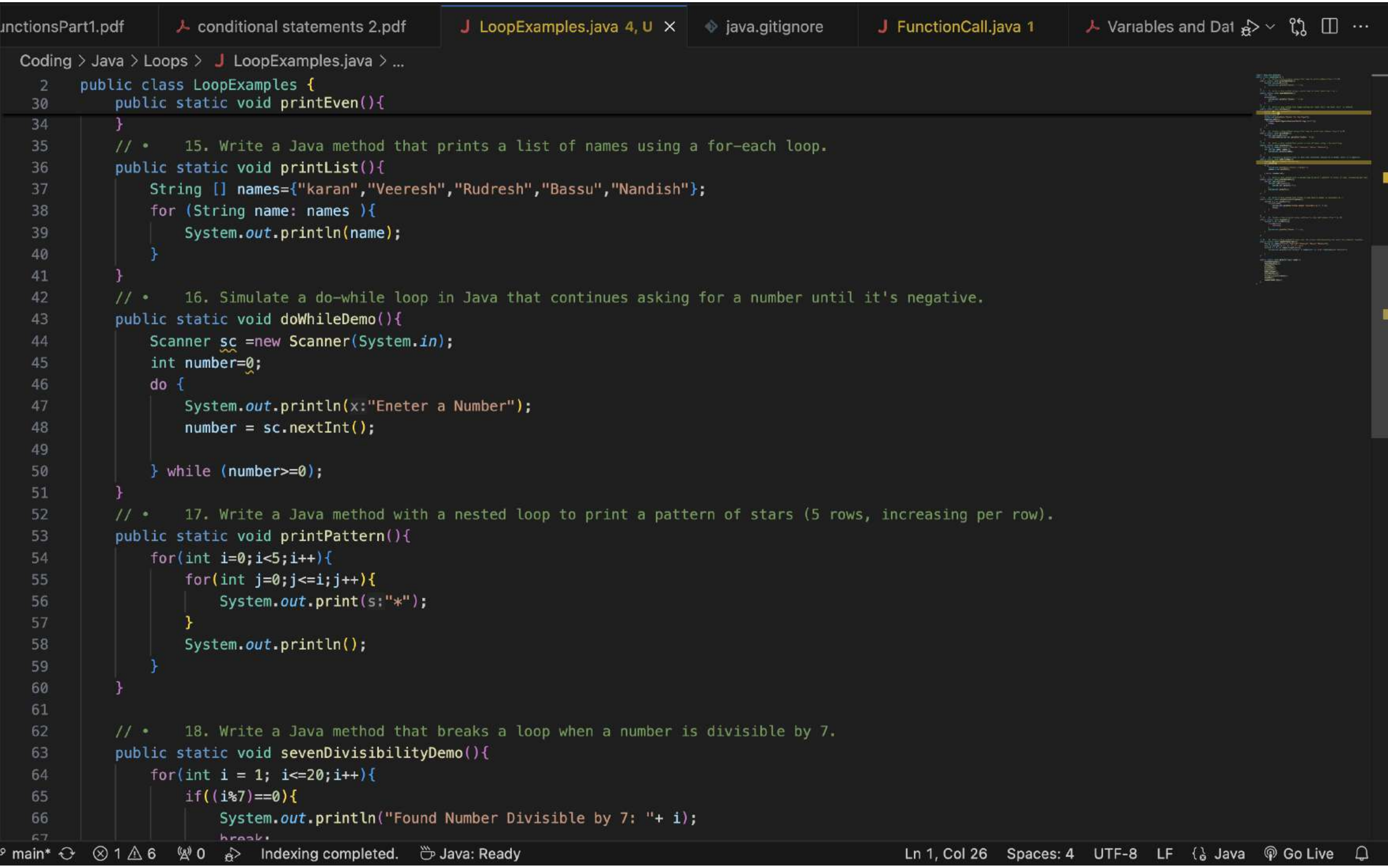
Alternate Printer Example

```
String[] printer1 = {"Doc1", "Doc3", "Doc5"};
String[] printer2 = {"Doc2", "Doc4", "Doc6"};

for (int i = 0; i < printer1.length; i++) {
    System.out.println("Printer 1: " + printer1[i]);
    System.out.println("Printer 2: " + printer2[i]);
}
```

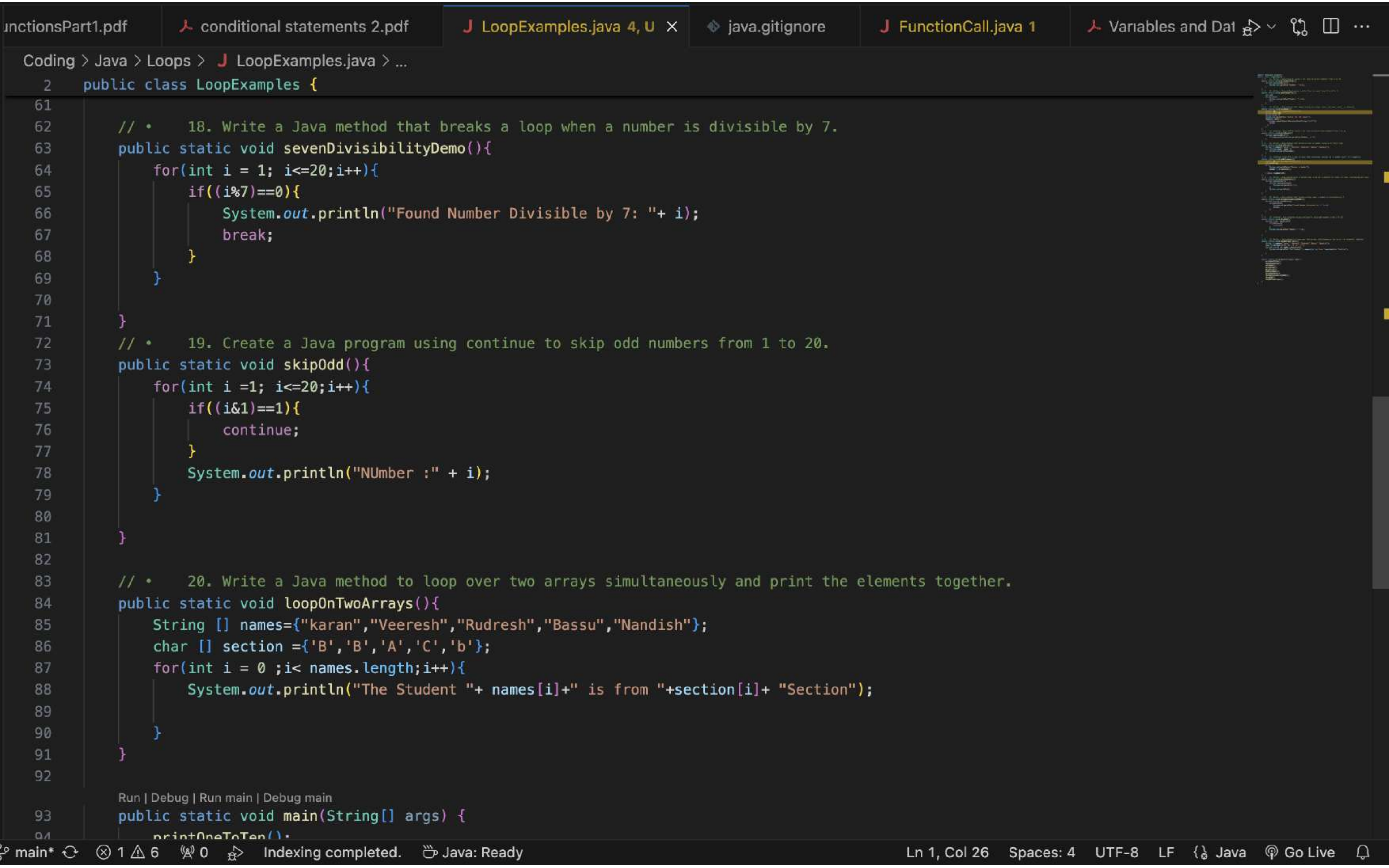


unctionsPart1.pdf
conditional statements 2.pdf
LoopExamples.java 4, U
java.gitignore
FunctionCall.java 1
Variables and Dat
Coding > Java > Loops > LoopExamples.java > ...
main* ⊗ 1 ⚠ 6 0 Indexing completed. Java: Ready Ln 1, Col 26 Spaces: 4 UTF-8 LF Java Go Live

```java
import java.util.Scanner;
public class LoopExamples {
    // •    11. Write a Java program using a for loop to print numbers from 1 to 10.
    public static void printOneToTen(){
        for(int i=1;i<=10;i++){
            System.out.println("Number: " + i);
        }
    }
    // •    12. Write a Java method using a while loop to count down from 5 to 1.
    public static void countDown5To1(){
        int i=5;
        while(i>=1){
            System.out.println("Number: " + i);
            i--;
        }
    }
    // •    13. Write a Java method that keeps asking for input until the word 'exit' is entered.
    public static void exitDemo(){
        Scanner sc = new Scanner(System.in);
        String input="";
        while (true) {
        System.out.println(x:"Eneter for the Input");
        input=sc.next();
          if(input.equalsIgnoreCase(anotherString:"exit")){
            break;
          }
        }
    }
    // •    14. Create a Java method using a for loop to print even numbers from 2 to 20.
    public static void printEven(){
        for(int i=2;i<=20;i++){
            if((i&1)==0)System.out.println("Number: "+ i);
        }
    }
    // •    15. Write a Java method that prints a list of names using a for-each loop.
```

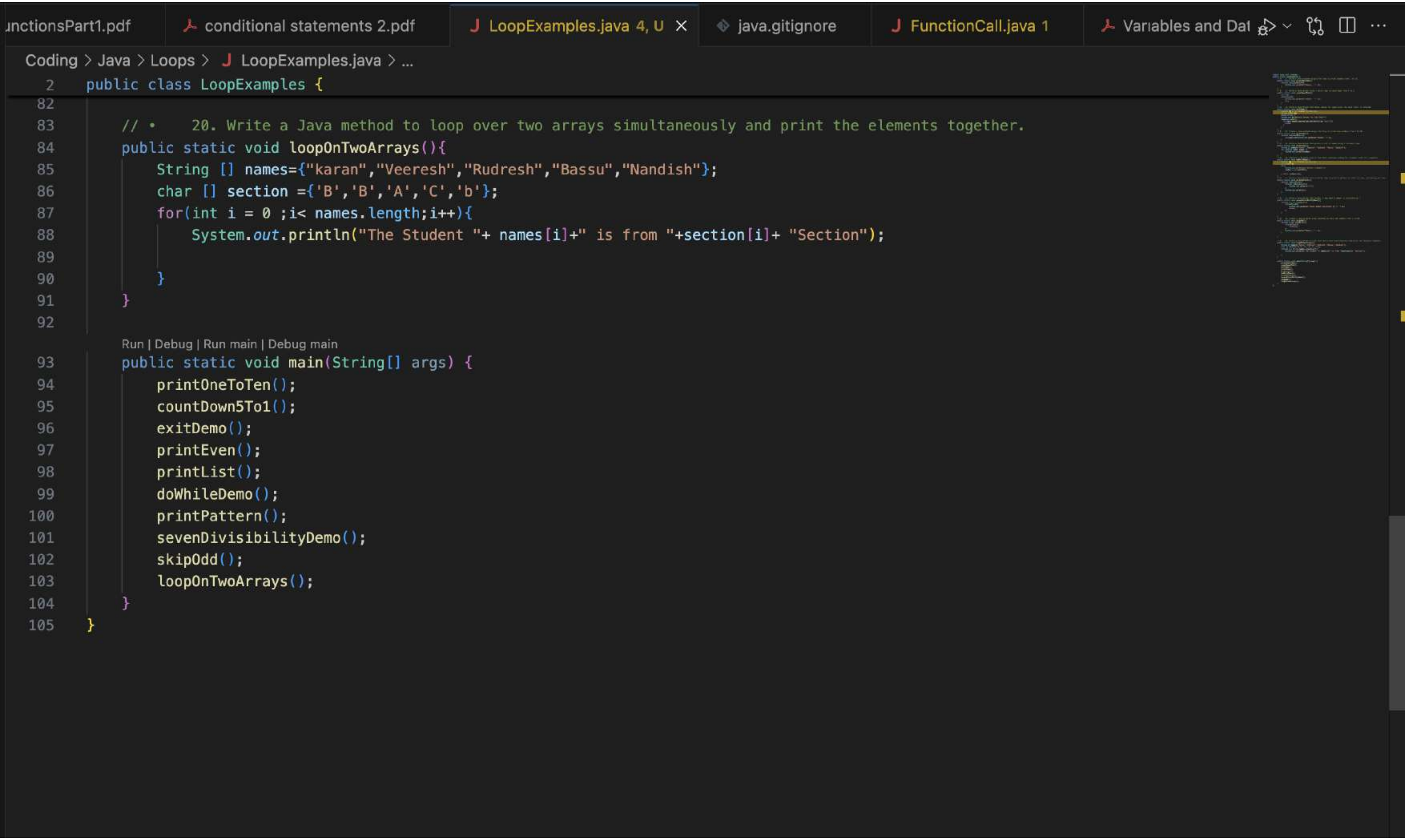


```java
public class LoopExamples {
    public static void printEven(){
    }
    // •    15. Write a Java method that prints a list of names using a for-each loop.
    public static void printList(){
        String [] names={"karan","Veeresh","Rudresh","Bassu","Nandish"};
        for (String name: names ){
            System.out.println(name);
        }
    }
    // •    16. Simulate a do-while loop in Java that continues asking for a number until it's negative.
    public static void doWhileDemo(){
        Scanner sc =new Scanner(System.in);
        int number=0;
        do {
            System.out.println(x:"Eneter a Number");
            number = sc.nextInt();

        } while (number>=0);
    }
    // •    17. Write a Java method with a nested loop to print a pattern of stars (5 rows, increasing per row).
    public static void printPattern(){
        for(int i=0;i<5;i++){
            for(int j=0;j<=i;j++){
                System.out.print(s:"*");
            }
            System.out.println();
        }
    }

    // •    18. Write a Java method that breaks a loop when a number is divisible by 7.
    public static void sevenDivisibilityDemo(){
        for(int i = 1; i<=20;i++){
            if((i%7)==0){
                System.out.println("Found Number Divisible by 7: "+ i);
                break;
```



```java
public class LoopExamples {

    // •    18. Write a Java method that breaks a loop when a number is divisible by 7.
    public static void sevenDivisibilityDemo(){
        for(int i = 1; i<=20;i++){
            if((i%7)==0){
                System.out.println("Found Number Divisible by 7: "+ i);
                break;
            }
        }

    }
    // •    19. Create a Java program using continue to skip odd numbers from 1 to 20.
    public static void skipOdd(){
        for(int i =1; i<=20;i++){
            if((i&1)==1){
                continue;
            }
            System.out.println("NUmber :" + i);
        }

    }

    // •    20. Write a Java method to loop over two arrays simultaneously and print the elements together.
    public static void loopOnTwoArrays(){
        String [] names={"karan","Veeresh","Rudresh","Bassu","Nandish"};
        char [] section ={'B','B','A','C','b'};
        for(int i = 0 ;i< names.length;i++){
            System.out.println("The Student "+ names[i]+" is from "+section[i]+ "Section");

        }
    }

    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
```

main* ⊗ 1 ⚠ 6 0 Indexing completed. Java: Ready Ln 1, Col 26 Spaces: 4 UTF-8 LF Java Go Live



unctionsPart1.pdf
conditional statements 2.pdf
LoopExamples.java 4, U
java.gitignore
FunctionCall.java 1
Variables and Dat
Coding > Java > Loops > LoopExamples.java > ...
Run | Debug | Run main | Debug main

```java
public class LoopExamples {

    // •    20. Write a Java method to loop over two arrays simultaneously and print the elements together.
    public static void loopOnTwoArrays(){
        String [] names={"karan","Veeresh","Rudresh","Bassu","Nandish"};
        char [] section ={'B','B','A','C','b'};
        for(int i = 0 ;i< names.length;i++){
            System.out.println("The Student "+ names[i]+" is from "+section[i]+ "Section");

        }
    }

    public static void main(String[] args) {
        printOneToTen();
        countDown5To1();
        exitDemo();
        printEven();
        printList();
        doWhileDemo();
        printPattern();
        sevenDivisibilityDemo();
        skipOdd();
        loopOnTwoArrays();
    }
}
```