

IN THE NAME OF ALLAH

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

THE GRACIOUS, THE MERCIFUL.



GETS - GETTING EDUCATION WITH TECHNOLOGICAL SYSTEM

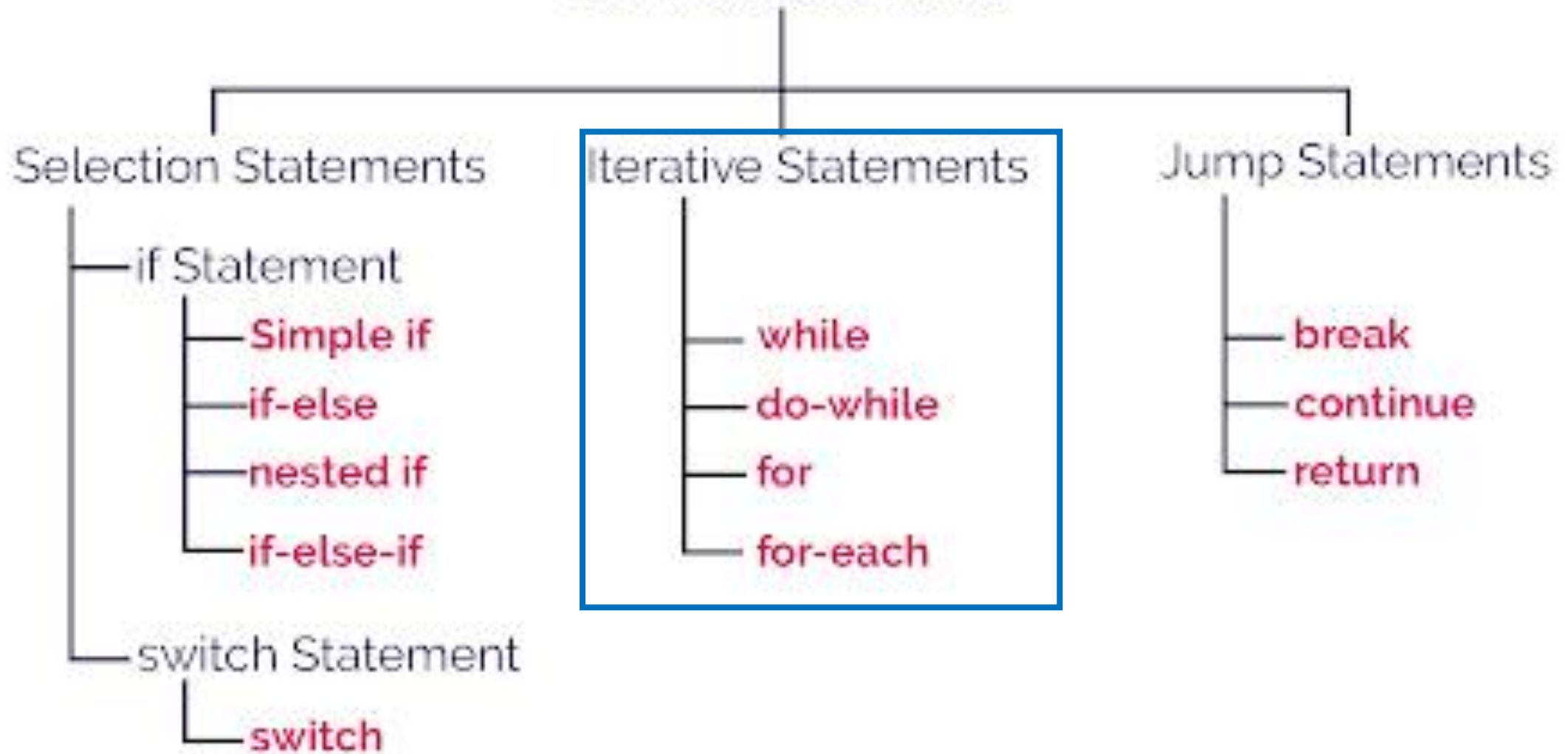
Instructor: Sir A.Rehman Ali Brohi

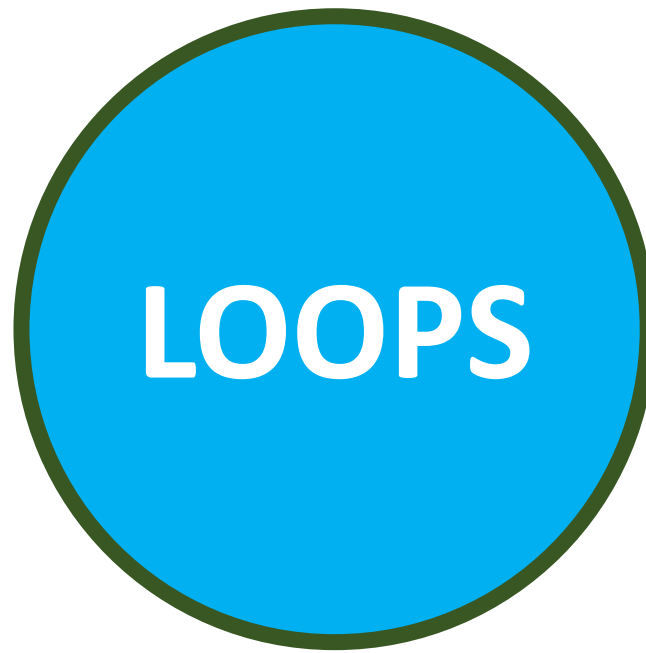
LECTURE: 6

LOOPS



Control Statements





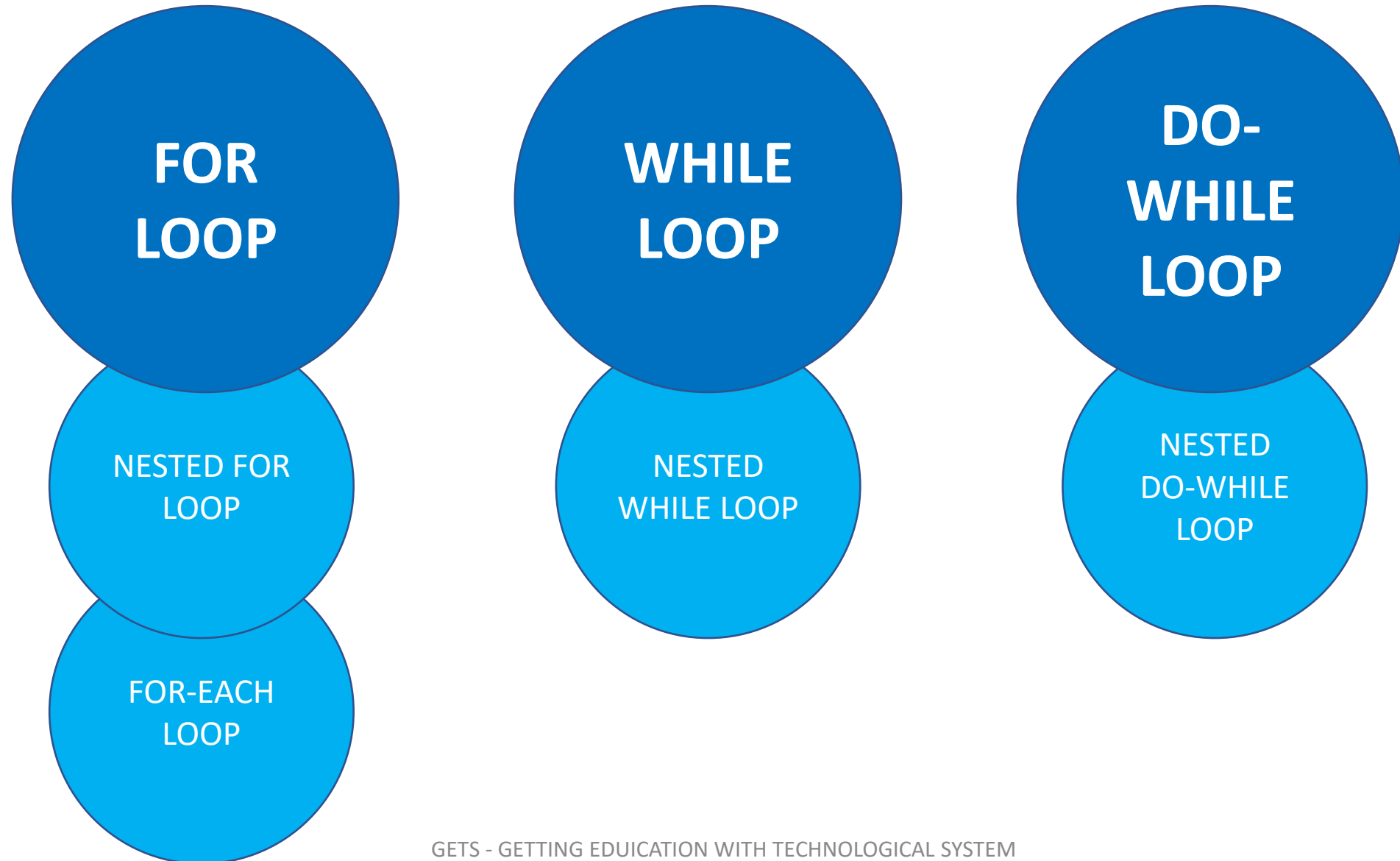
loops are used to repeat a block of code.

For example, if you want to show a message 100 times, then rather than typing the same code 100 times, you can use a loop.

Loops means REPETITION

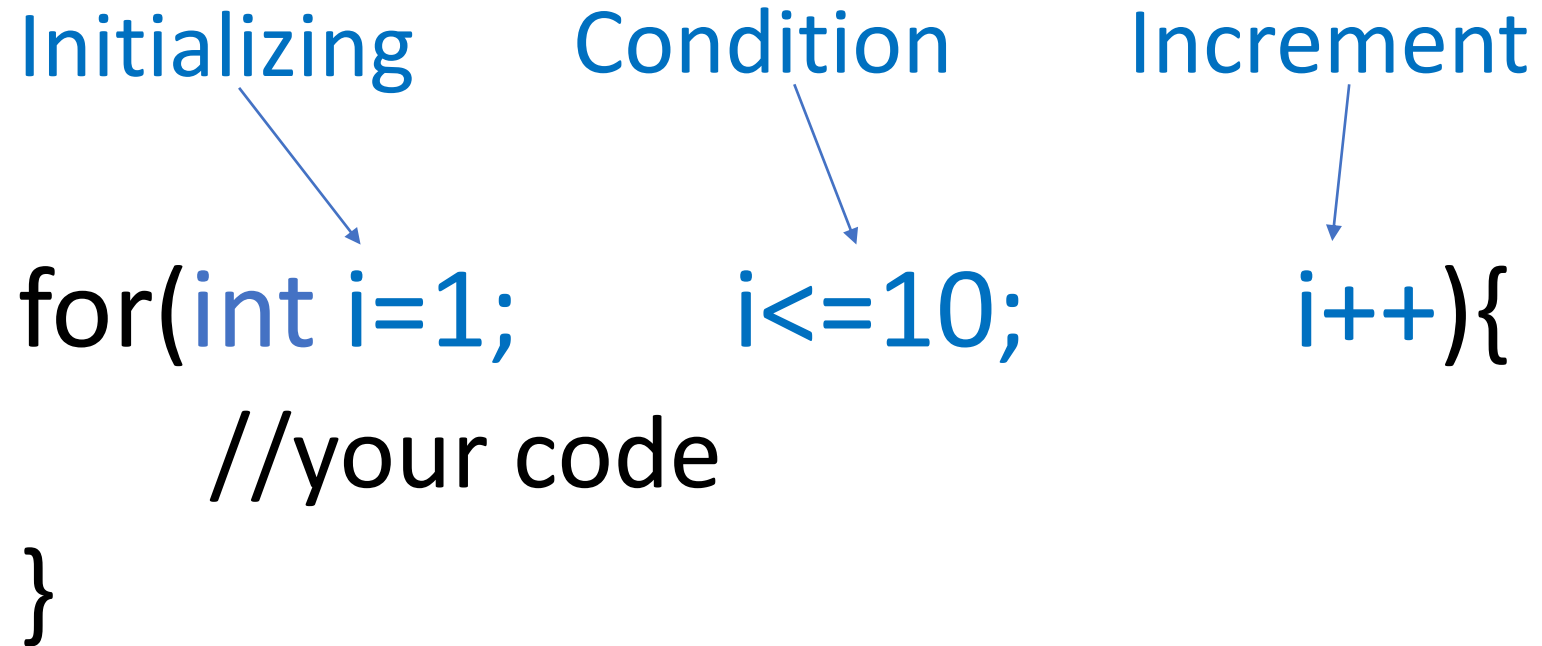
REPETITION
REPETITION
REPETITION
REPETITION
REPETITION
REPETITION
REPETITION

3 TYPES OF ITERATION/LOOPS IN JAVA



FOR LOOP

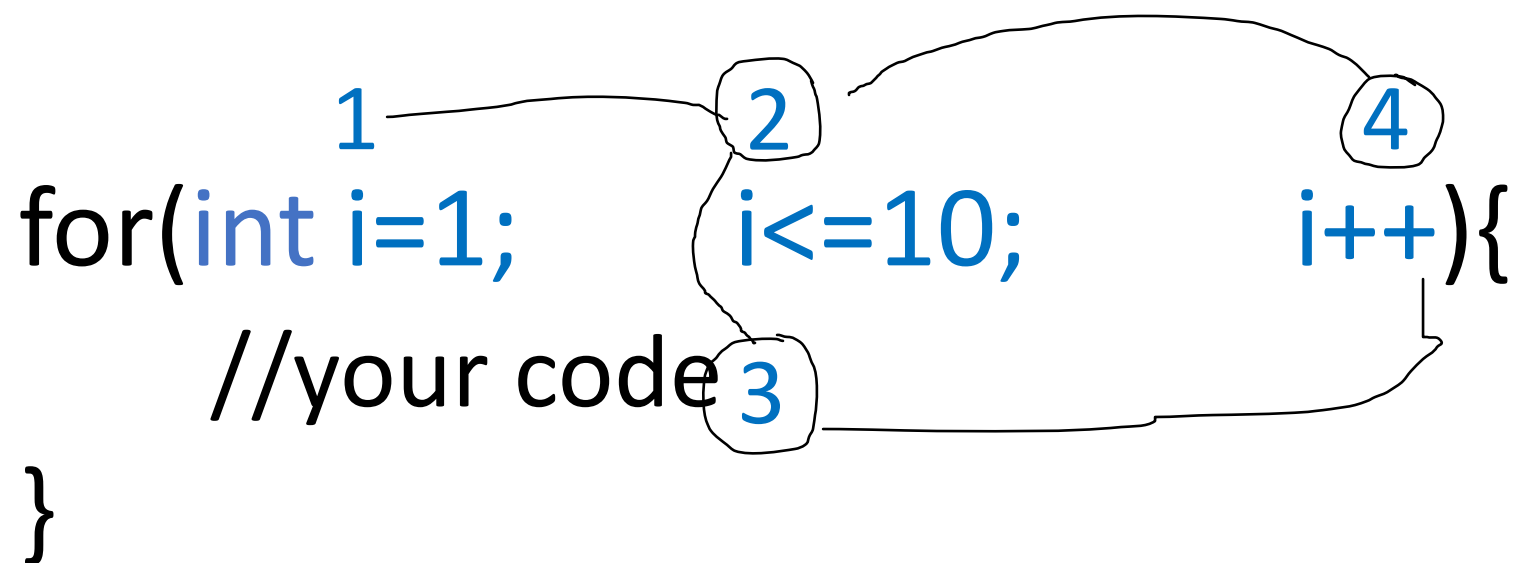
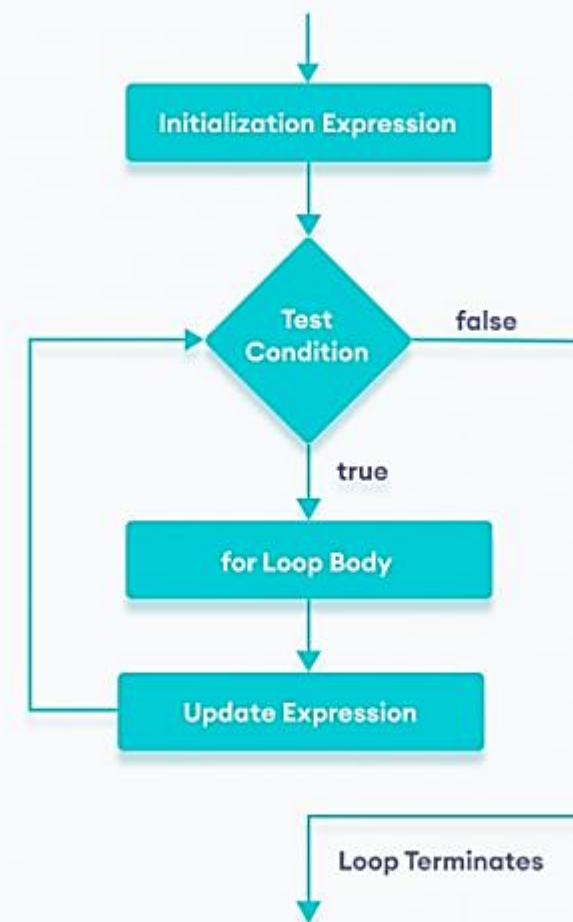
Initializing Condition Increment



```
for(int i=1;      i<=10;      i++){  
    //your code  
}
```

FOR LOOP (HOW CODE WORKS)

1
for(int i=1; 2 i<=10; 4 i++){
//your code 3
}

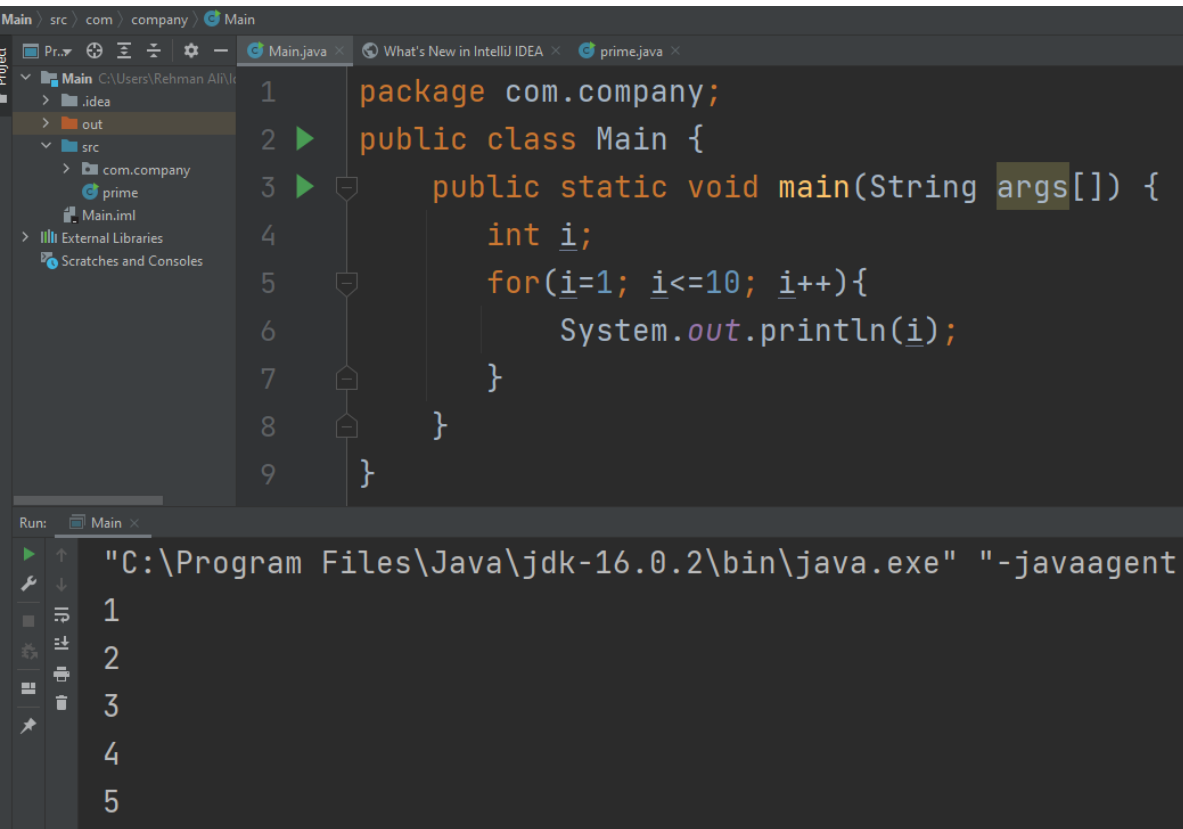
A diagram illustrating the components of a Java for loop. The code snippet is: `for(int i=1; i<=10; i++){
//your code
}`. Handwritten annotations include: a '1' pointing to the opening curly brace, a '2' in a circle pointing to the initialization 'i=1', a '3' in a circle pointing to the body '//your code', and a '4' in a circle pointing to the increment 'i++'. A curved line connects the '2' and '4' annotations, indicating the loop's execution cycle.

Flowchart of Java for loop

Here is how this program works.

Iteration	Variable	Condition: $i \leq n$	Action
1st	<div>i = 1</div> <div>n = 5</div>	true	<div>Java is fun</div> is printed. <div>i</div> is increased to 2 .
2nd	<div>i = 2</div> <div>n = 5</div>	true	<div>Java is fun</div> is printed. <div>i</div> is increased to 3 .
3rd	<div>i = 3</div> <div>n = 5</div>	true	<div>Java is fun</div> is printed. <div>i</div> is increased to 4 .
4th	<div>i = 4</div> <div>n = 5</div>	true	<div>Java is fun</div> is printed. <div>i</div> is increased to 5 .
5th	<div>i = 5</div> <div>n = 5</div>	true	<div>Java is fun</div> is printed. <div>i</div> is increased to 6 .
6th	<div>i = 6</div> <div>n = 5</div>	false	The loop is terminated.

JAVA FOR LOOP PROGRAMS

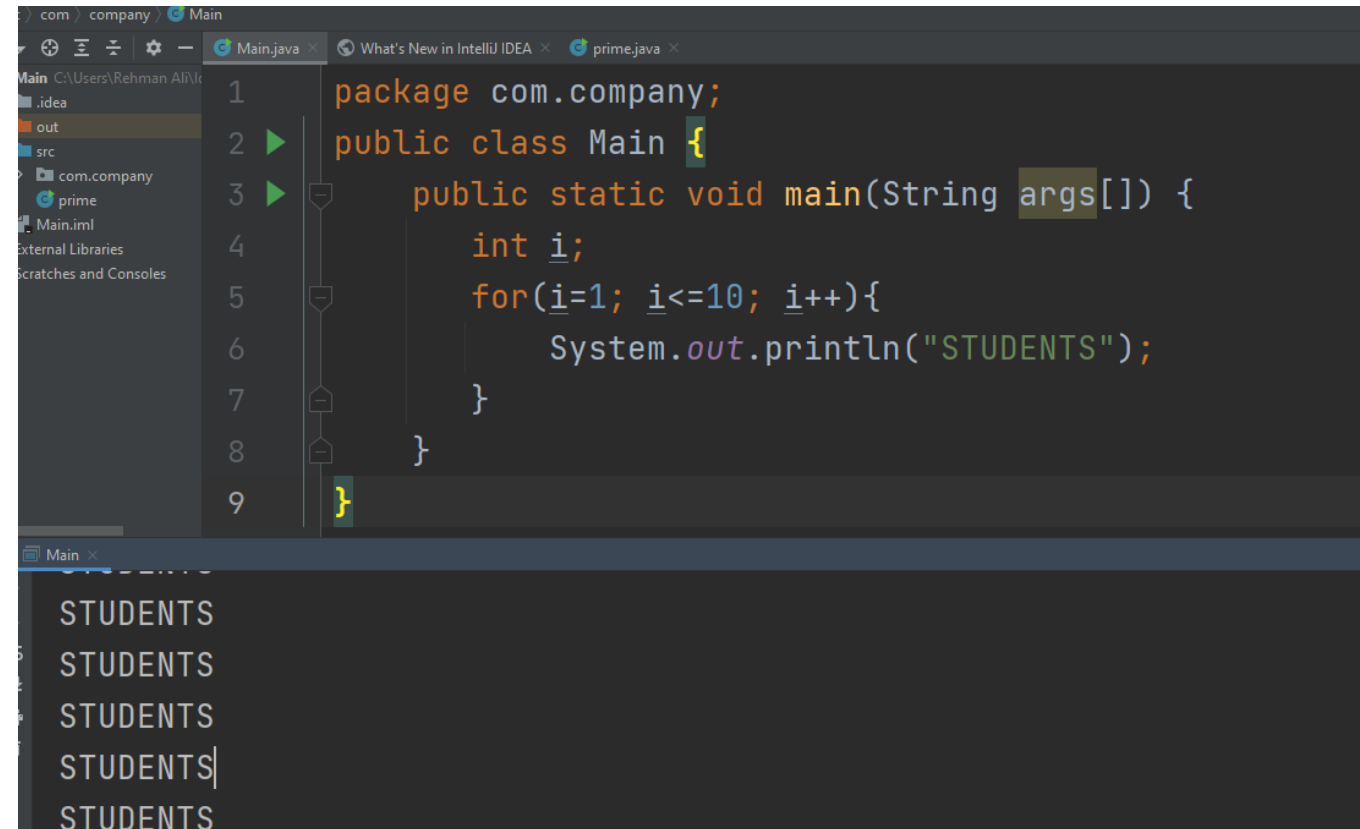


```
1 package com.company;
2 public class Main {
3     public static void main(String args[]) {
4         int i;
5         for(i=1; i<=10; i++){
6             System.out.println(i);
7         }
8     }
9 }
```

Run: Main x

"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:
1
2
3
4
5

Display Natural Numbers 1 to 10



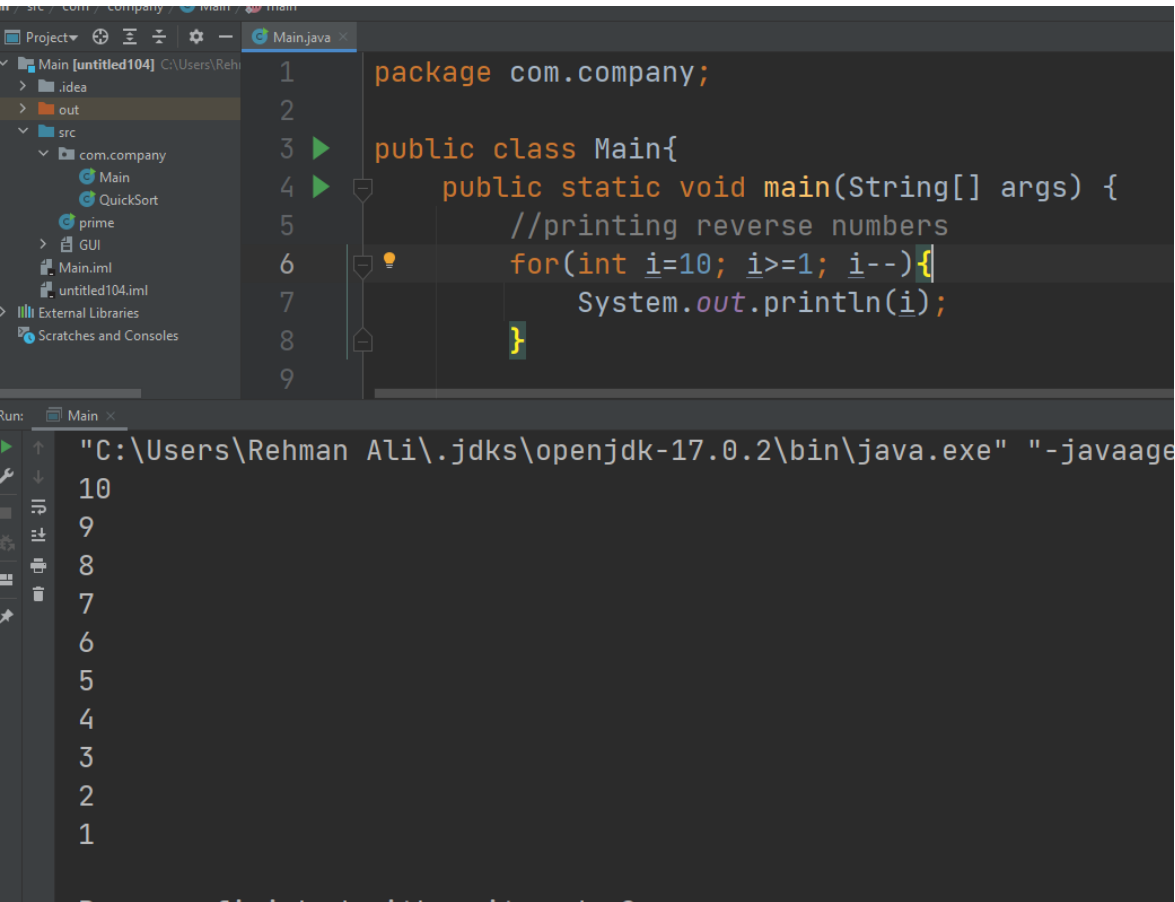
```
1 package com.company;
2 public class Main {
3     public static void main(String args[]) {
4         int i;
5         for(i=1; i<=10; i++){
6             System.out.println("STUDENTS");
7         }
8     }
9 }
```

Main x

STUDENTS
STUDENTS
STUDENTS
STUDENTS
STUDENTS

Display text repetition

JAVA FOR LOOP PROGRAMS

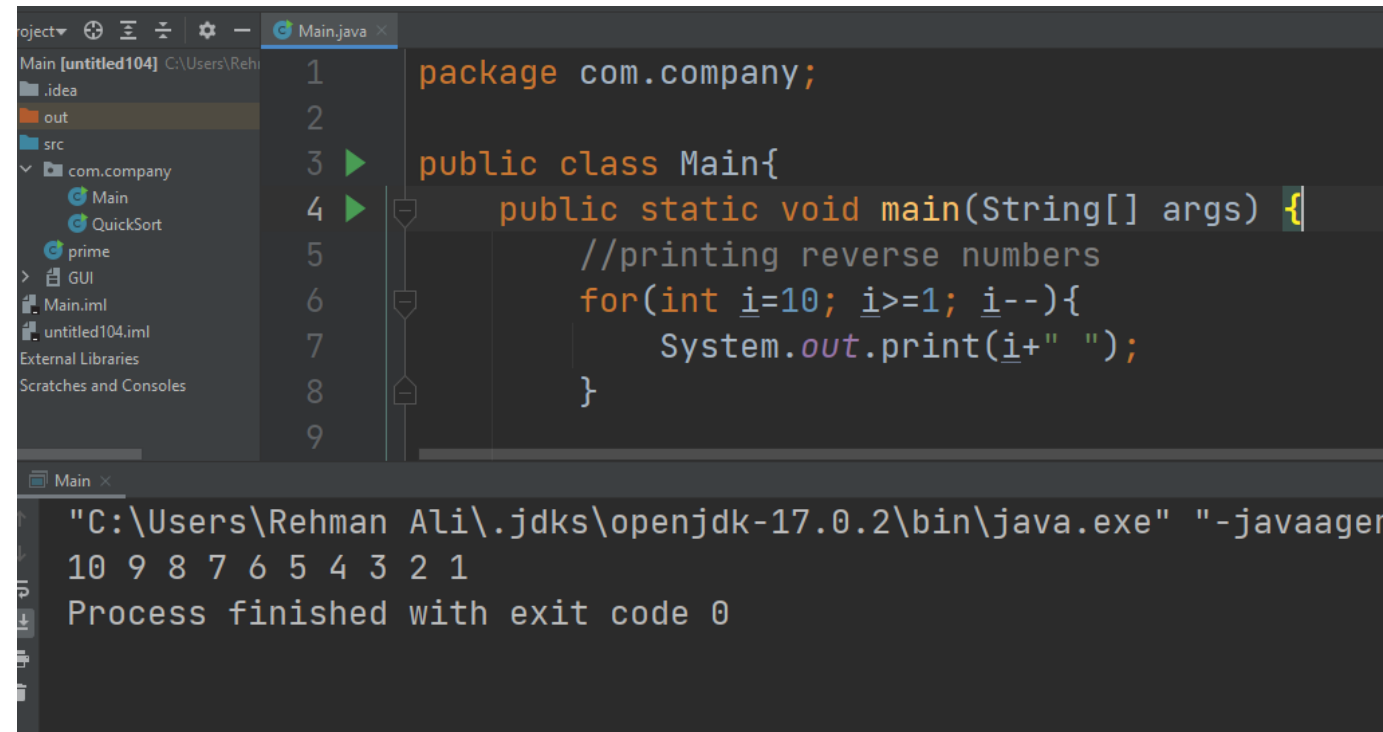


```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         //printing reverse numbers
6         for(int i=10; i>=1; i--){
7             System.out.println(i);
8         }
9     }
}
```

Run: Main

```
"C:\Users\Rehman Ali\.jdk\openjdk-17.0.2\bin\java.exe" "-javaagent..."
10
9
8
7
6
5
4
3
2
1
```

Display reverse numbers



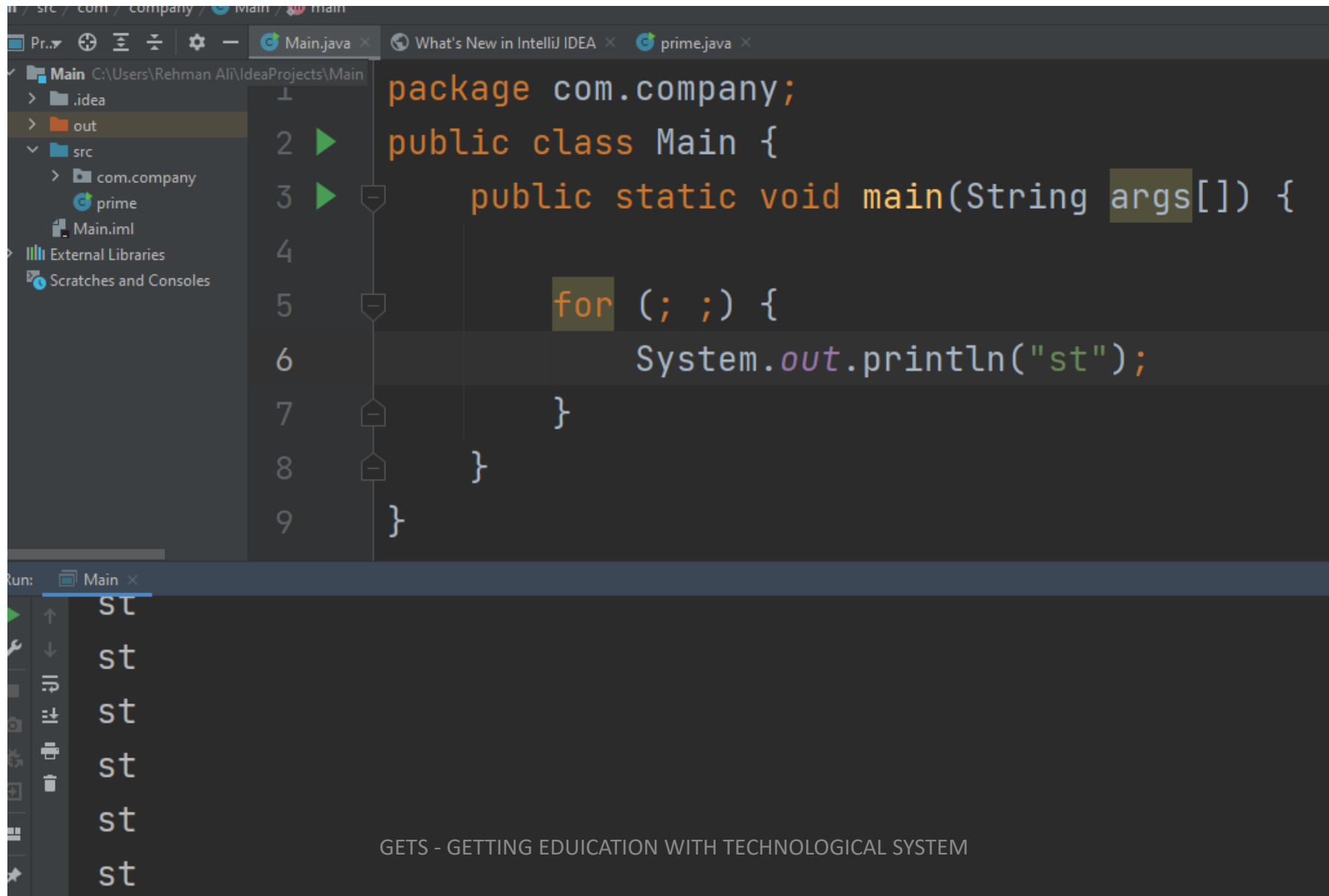
```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         //printing reverse numbers
6         for(int i=10; i>=1; i--){
7             System.out.print(i+" ");
8         }
9     }
}
```

Main

```
"C:\Users\Rehman Ali\.jdk\openjdk-17.0.2\bin\java.exe" "-javaagent..."
10 9 8 7 6 5 4 3 2 1
Process finished with exit code 0
```

Can also Display as horizontal/rows

INFINITIVE FOR LOOP (non-stop)



The screenshot displays the IntelliJ IDEA IDE. The main editor window shows the file `Main.java` with the following Java code:

```
1 package com.company;  
2 public class Main {  
3     public static void main(String args[]) {  
4  
5         for (; ; ) {  
6             System.out.println("st");  
7         }  
8     }  
9 }
```

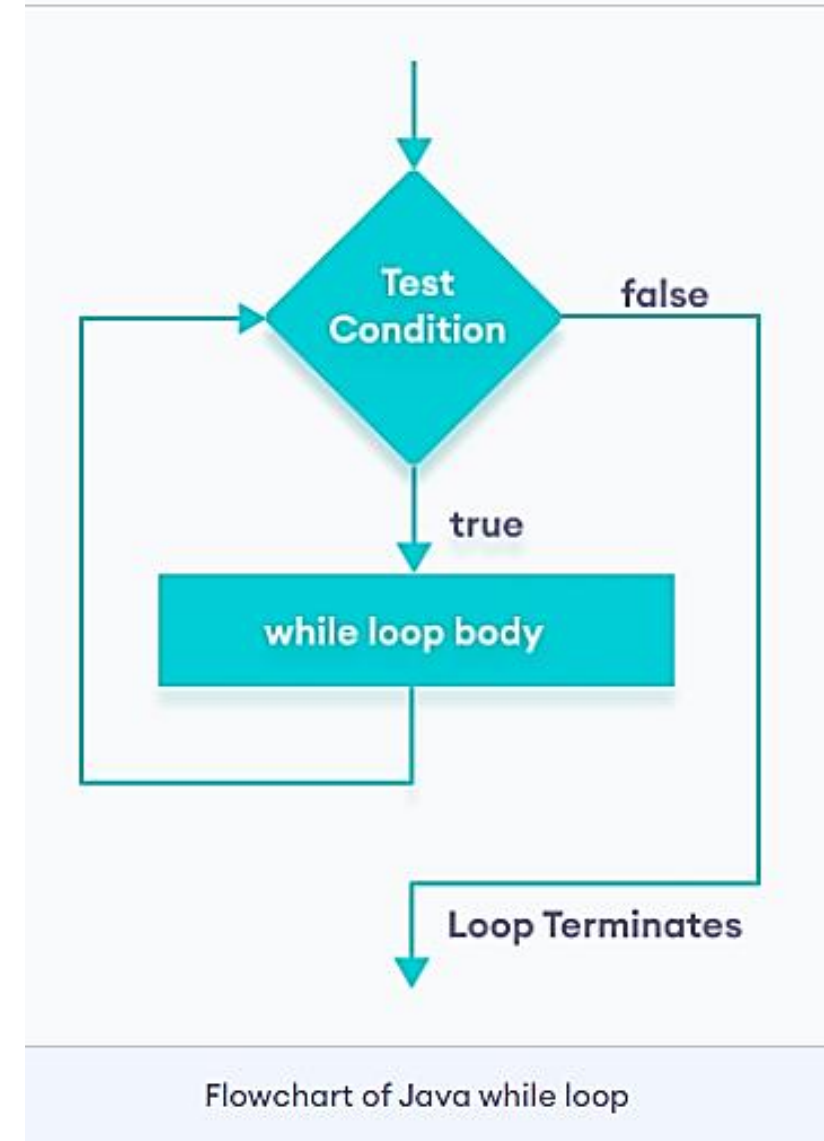
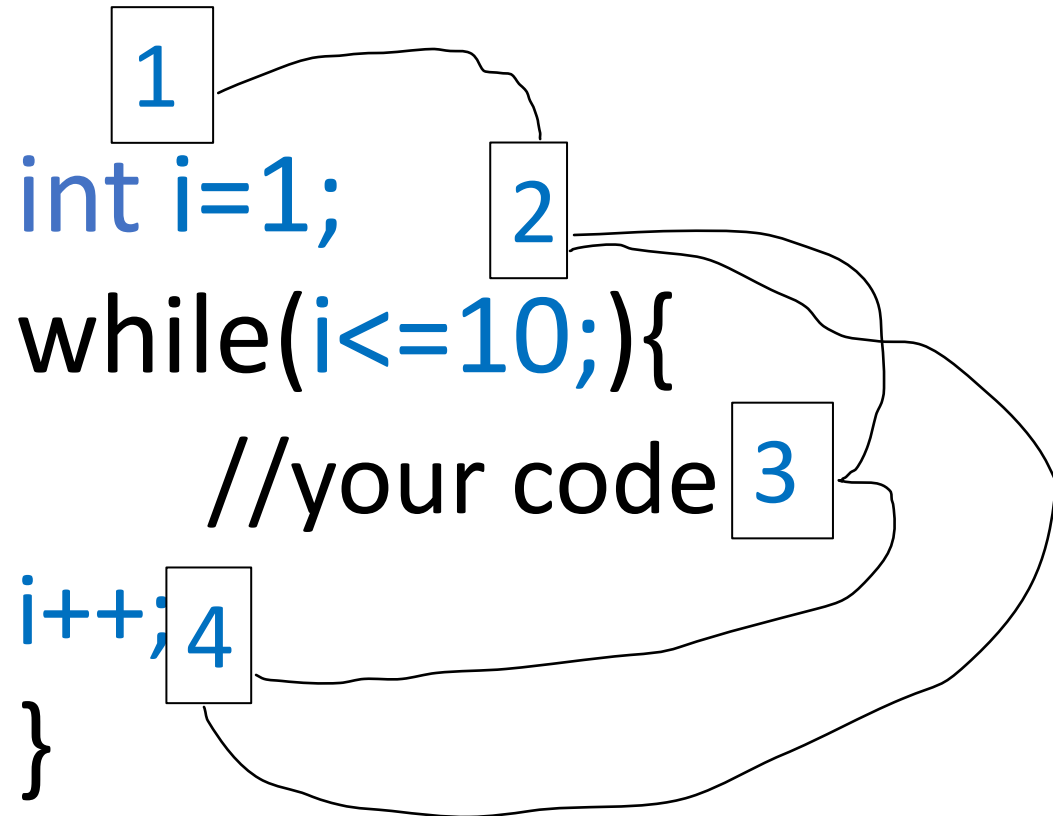
The code implements an infinite loop using a `for` loop with empty semicolons (`for (; ;)`). The loop body contains a single statement: `System.out.println("st");`. The IDE's Run console at the bottom shows the output of the program, which is the string `st` printed multiple times, demonstrating the non-stop nature of the loop.

JAVA WHILE LOOP

- The Java while loop is used to iterate a part of the program several times.

If the number of iteration is not fixed, it is recommended to use while loop.

WHILE LOOP (HOW CODE WORKS)



Here is how this program works.

Iteration	Variable	Condition: $i \leq n$	Action
1st	<div>i = 1</div> <div>n = 5</div>	true	<div>1 is printed.</div> <div>i is increased to 2.</div>
2nd	<div>i = 2</div> <div>n = 5</div>	true	<div>2 is printed.</div> <div>i is increased to 3.</div>
3rd	<div>i = 3</div> <div>n = 5</div>	true	<div>3 is printed.</div> <div>i is increased to 4.</div>
4th	<div>i = 4</div> <div>n = 5</div>	true	<div>4 is printed.</div> <div>i is increased to 5.</div>
5th	<div>i = 5</div> <div>n = 5</div>	true	<div>5 is printed.</div> <div>i is increased to 6.</div>
6th	<div>i = 6</div> <div>n = 5</div>	false	The loop is terminated

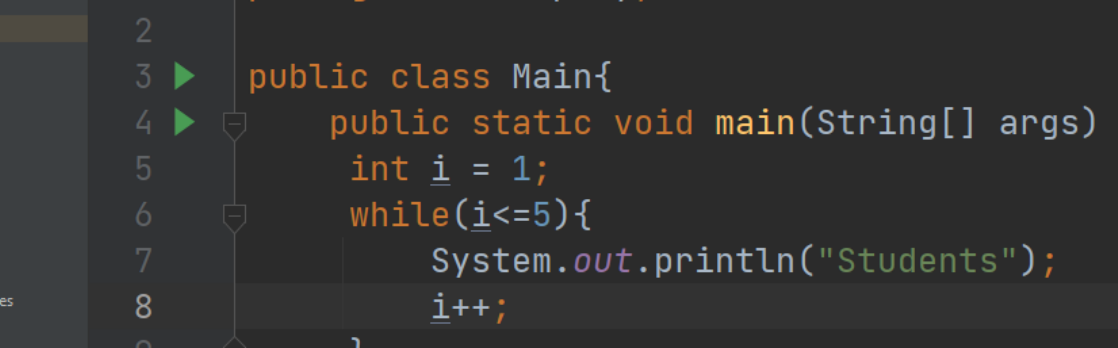
JAVA WHILE LOOP PROGRAMS

The screenshot shows the IntelliJ IDEA IDE interface. The top toolbar includes icons for running (a green play button) and debugging (a bug icon). Below the toolbar, there are two tabs: "Main.java" and "What's New in IntelliJ IDEA". The "Main.java" tab is active, displaying the following code:

```
1 package com.company;  
2 public class Main {  
3     public static void main(String args[]) {  
4         int i=1;  
5         while(i<=10){  
6             System.out.println(i);  
7             i++;  
8         }  
9     }
```

The code is syntactically highlighted. To the left of the code editor is a project explorer showing the file structure: "Main" (C:\Users\Rehman Ali\IdeaProjects\Main), ".idea", "out", "src", "com.company", "prime", "Main.iml", "External Libraries", and "Scratches and Consoles". At the bottom of the screen, a terminal window titled "Run: Main" displays the command used to execute the program: "C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\Java\jdk-16.0.2\lib\jrt-fs.jar" "-Djava.class.path=C:\Users\Rehman Ali\AppData\Local\Temp\intellij-run-agent-1602\agent.jar" "com.company.Main". The terminal output shows the numbers 1 through 10, each on a new line, indicating the successful execution of the loop.

Display Natural Numbers 1 to 10



The screenshot shows an IDE with a Java file named `Main.java` open. The code defines a package `com.company` and a public class `Main` with a `main` method. The `main` method initializes `i` to 1 and enters a `while` loop that prints "Students" and increments `i` until it reaches 5. The left sidebar shows the project structure with `src` containing `com.company` and `Main`. The bottom console shows the command used to run the program and the output, which is "Students" printed five times.

```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         int i = 1;
6         while(i<=5){
7             System.out.println("Students");
8             i++;
9         }
10    }
```

Execution Command:

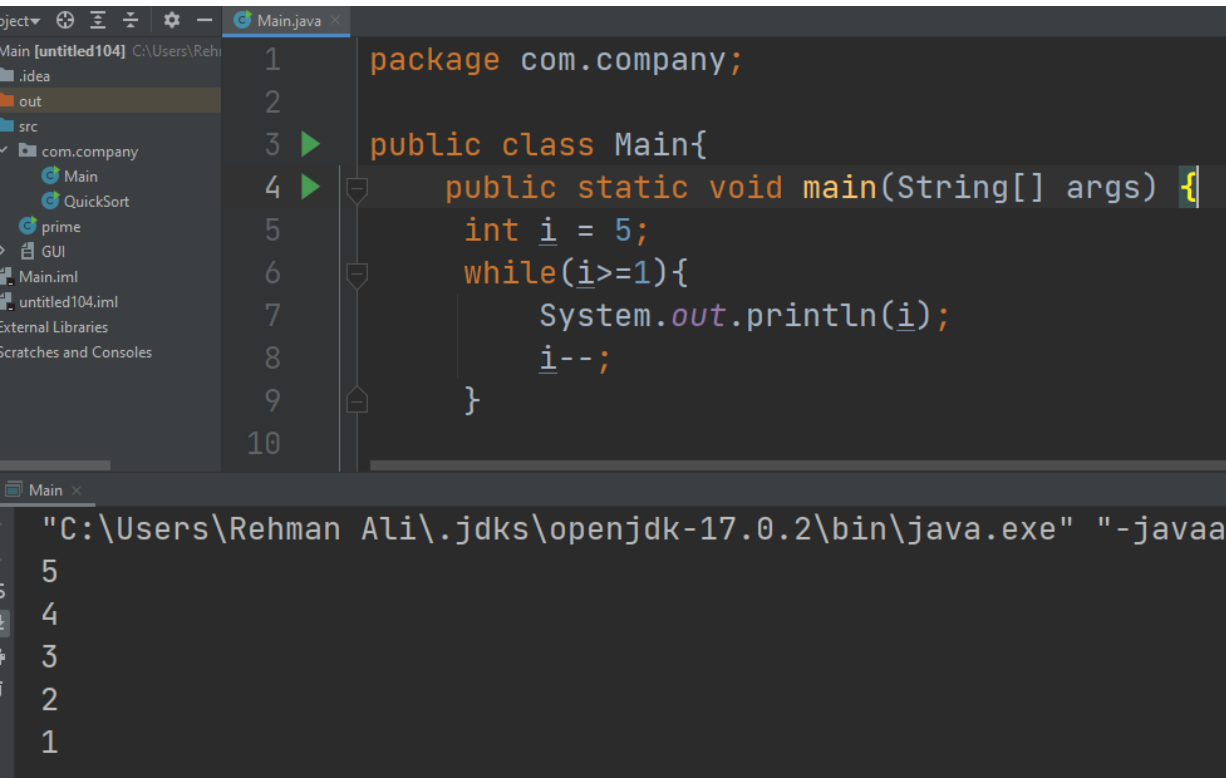
```
"C:\Users\Rehman Ali\.jdk\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\Java\jdk-17.0.2\lib\jconsole.jar" -classpath .;C:\Users\Rehman Ali\IdeaProjects\untitled104\out\production\untitled104 Main
```

Output:

```
Students
Students
Students
Students
Students
```

Display text repetition

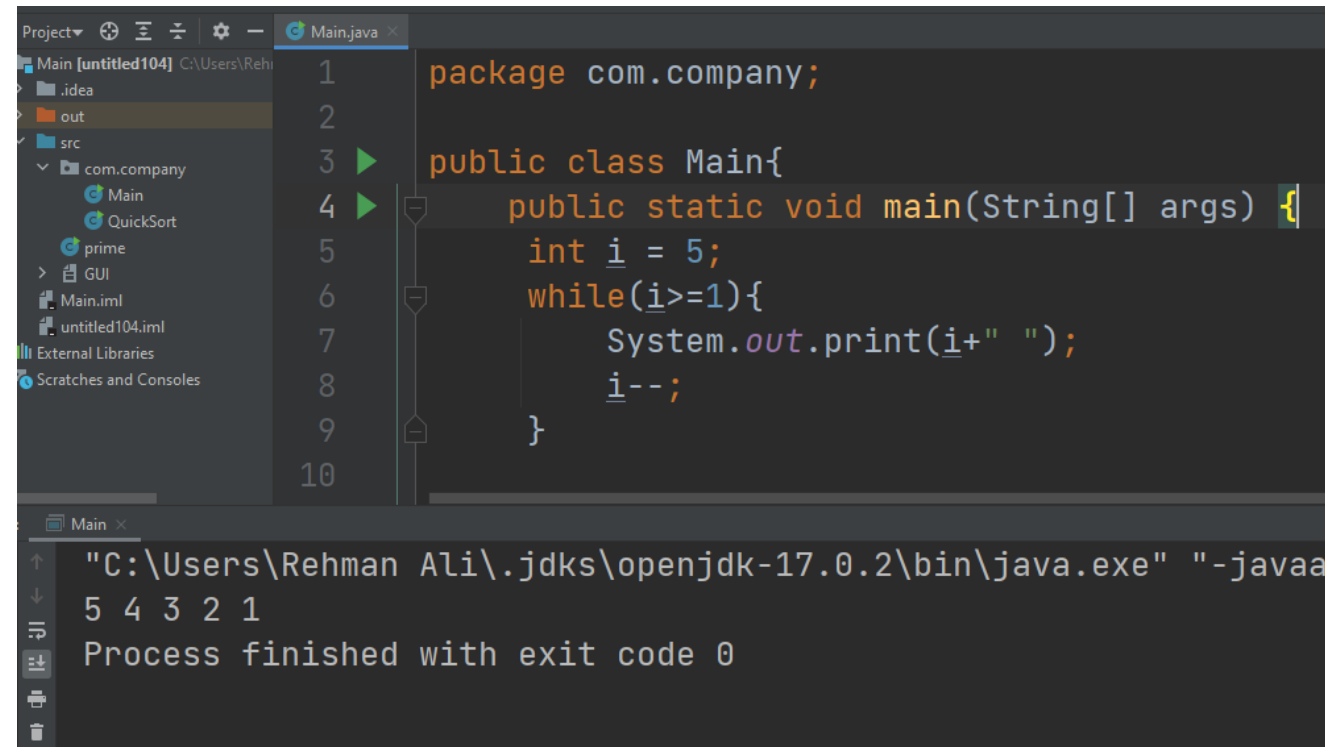
JAVA WHILE LOOP PROGRAMS



```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         int i = 5;
6         while(i>=1){
7             System.out.println(i);
8             i--;
9         }
10    }
```

The console output shows the numbers 5, 4, 3, 2, 1 printed on separate lines.

Display Natural Numbers in reverse

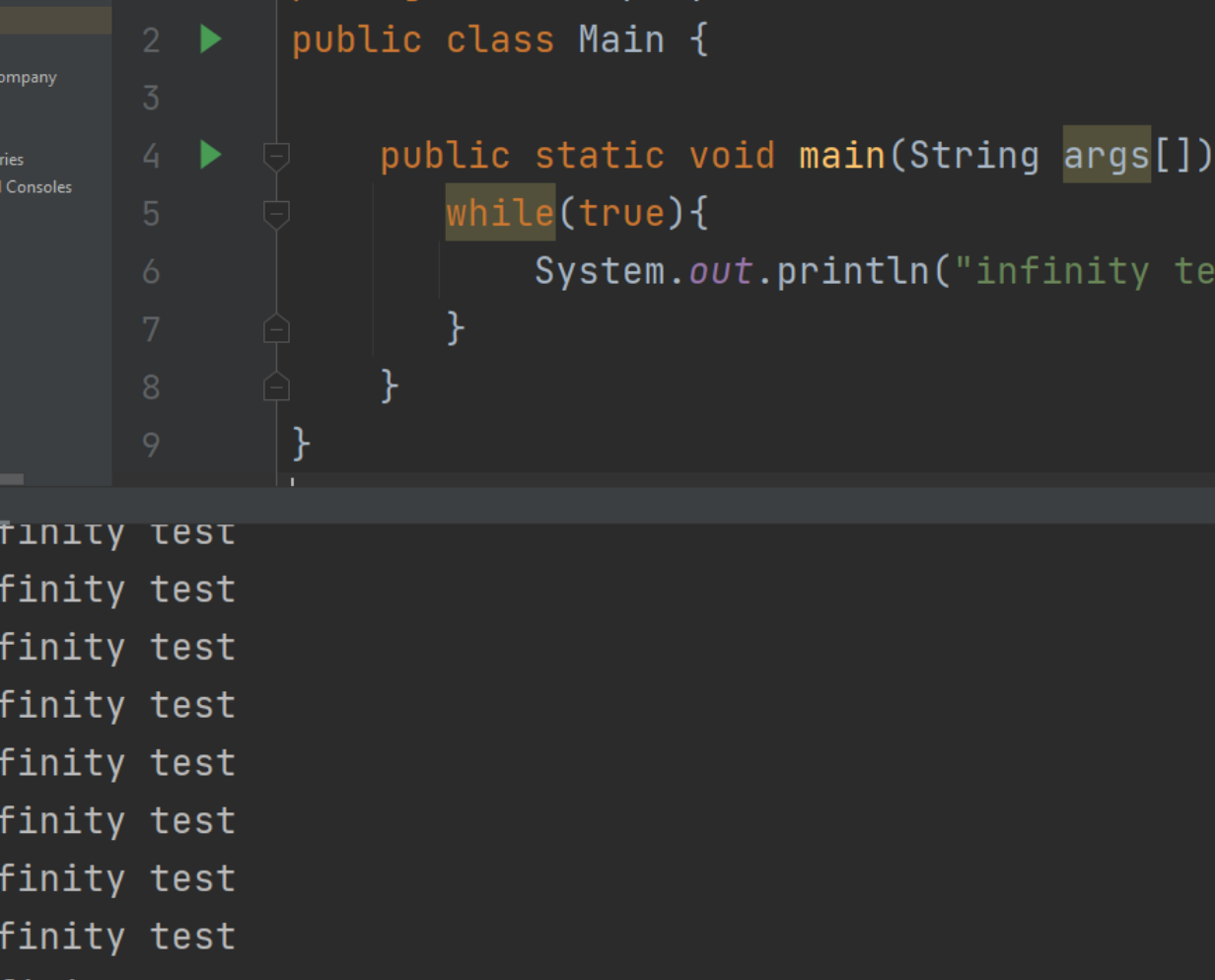


```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         int i = 5;
6         while(i>=1){
7             System.out.print(i+" ");
8             i--;
9         }
10    }
```

The console output shows the numbers 5 4 3 2 1 printed on a single line, followed by the message "Process finished with exit code 0".

Can also Display as horizontal/rows

JAVA WHILE INFINITIVE LOOP



The screenshot displays the IntelliJ IDEA IDE. The top toolbar shows icons for file operations (New, Open, Save, etc.), a settings gear, and tabs for 'Main.java', 'What's New in IntelliJ IDEA', and 'prime.java'. The left sidebar shows a project tree with 'Main' at the root, containing 'src' and 'out' folders. The 'src' folder is expanded, showing 'com.company' and 'prime' packages. The 'Main' class is selected. The main editor window shows the following Java code:

```
1 package com.company;  
2 public class Main {  
3  
4     public static void main(String args[]) {  
5         while(true){  
6             System.out.println("infinity test");  
7         }  
8     }  
9 }
```

The code is syntactically correct. The 'while(true)' loop is highlighted with a yellow background. The 'System.out.println' statement is highlighted with a green background. The output window at the bottom shows the following text:

```
infinity test  
infinity test  
infinity test  
infinity test  
infinity test  
infinity test  
infinity test  
infinity test  
infinity test
```

The output window also shows the message 'Process finished with exit code -1'.

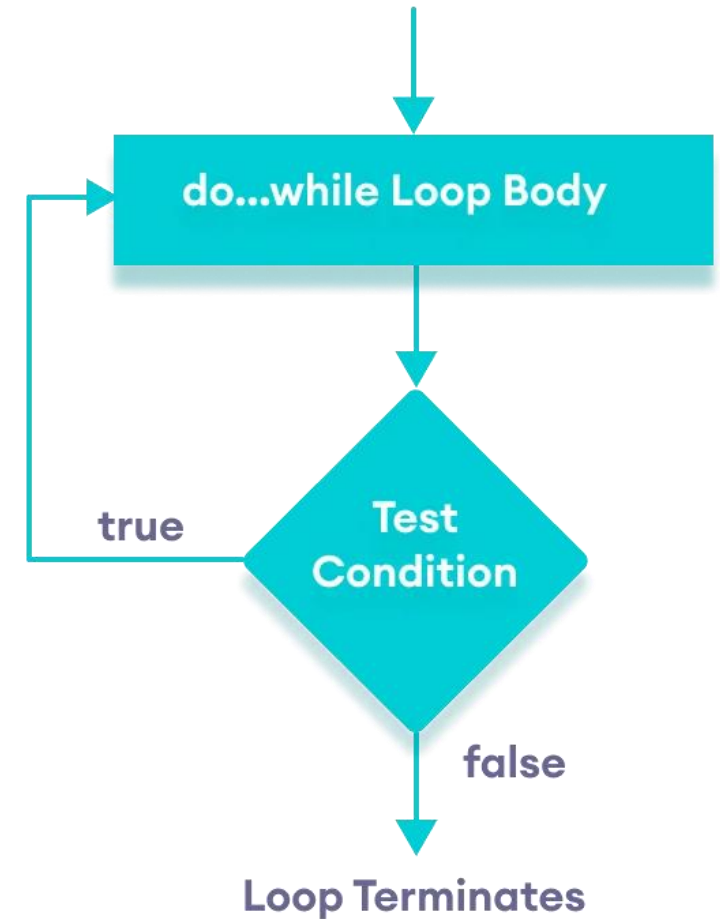
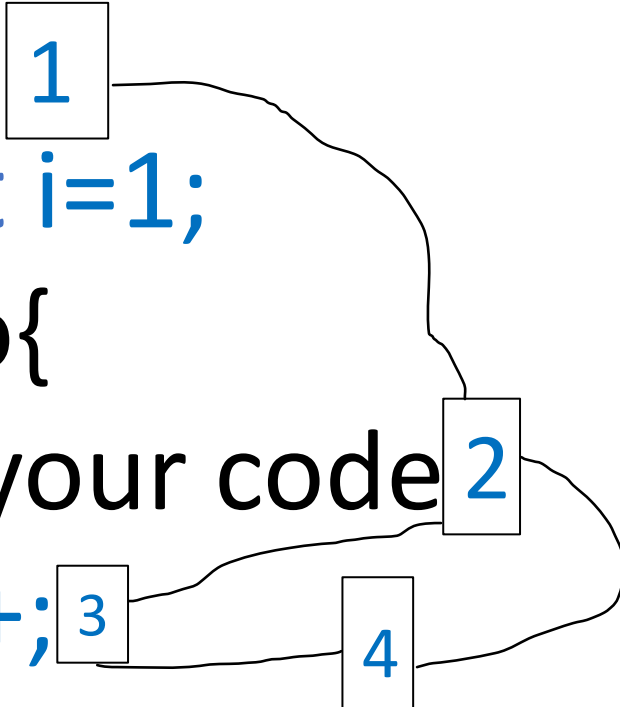
JAVA DO-WHILE LOOP

- The Java do-while loop is used to iterate a part of the program several times.

If the number of iteration is not fixed and you must have to execute the loop at least once.

DO-WHILE LOOP (HOW CODE WORKS)

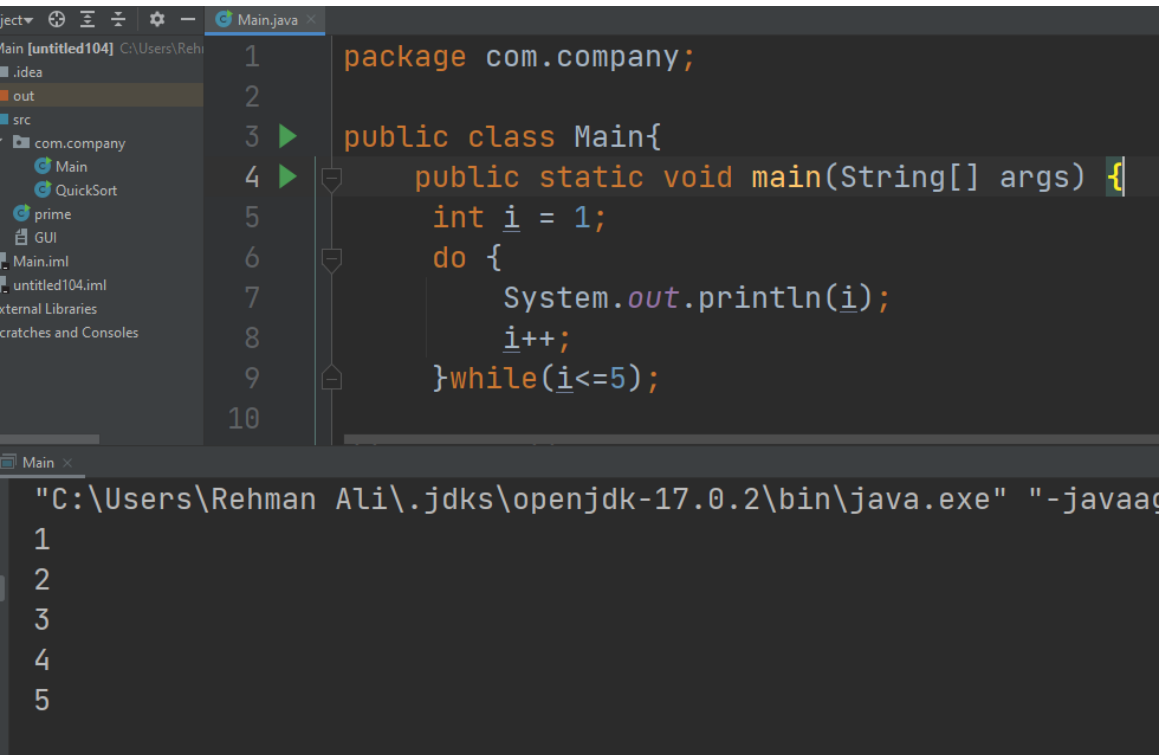
```
1  
int i=1;  
Do{  
  //your code 2  
  i++; 3  
}while(i<=10;); 4
```



Here is how this program works.

Iteration	Variable	Condition: $i \leq n$	Action
	$i = 1$ $n = 5$	not checked	1 is printed. i is increased to 2 .
1st	$i = 2$ $n = 5$	true	2 is printed. i is increased to 3 .
2nd	$i = 3$ $n = 5$	true	3 is printed. i is increased to 4 .
3rd	$i = 4$ $n = 5$	true	4 is printed. i is increased to 5 .
4th	$i = 5$ $n = 5$	true	5 is printed. i is increased to 6 .
5th	$i = 6$ $n = 5$	false	The loop is terminated

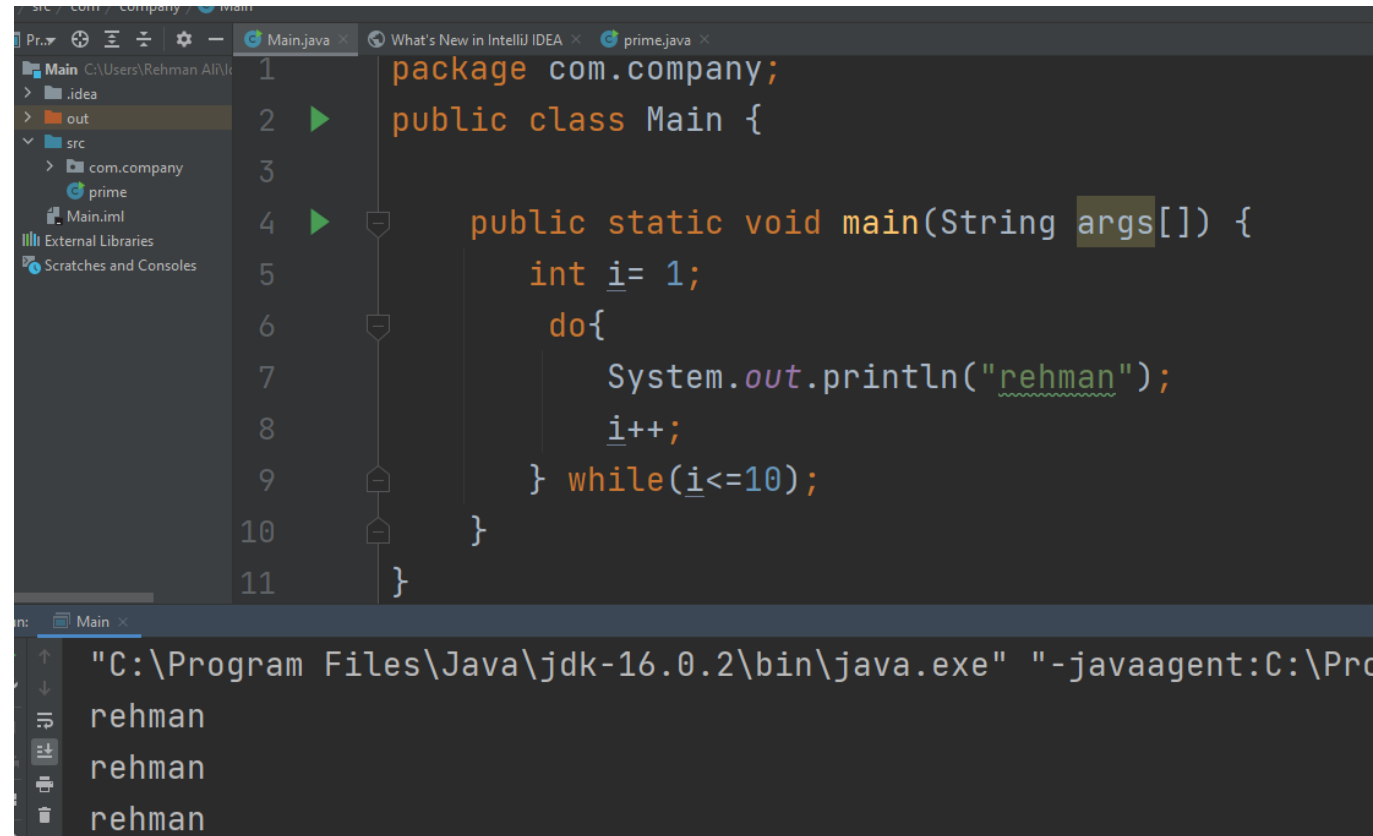
JAVA DO-WHILE LOOP PROGRAMS



```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         int i = 1;
6         do {
7             System.out.println(i);
8             i++;
9         }while(i<=5);
10    }
```

The screenshot shows the IntelliJ IDEA IDE with a Java file named Main.java. The code is a do-while loop that prints the numbers 1 through 5. The console output at the bottom shows the command to run the program and the resulting output: 1, 2, 3, 4, 5.

Display Natural Numbers 1 to 5

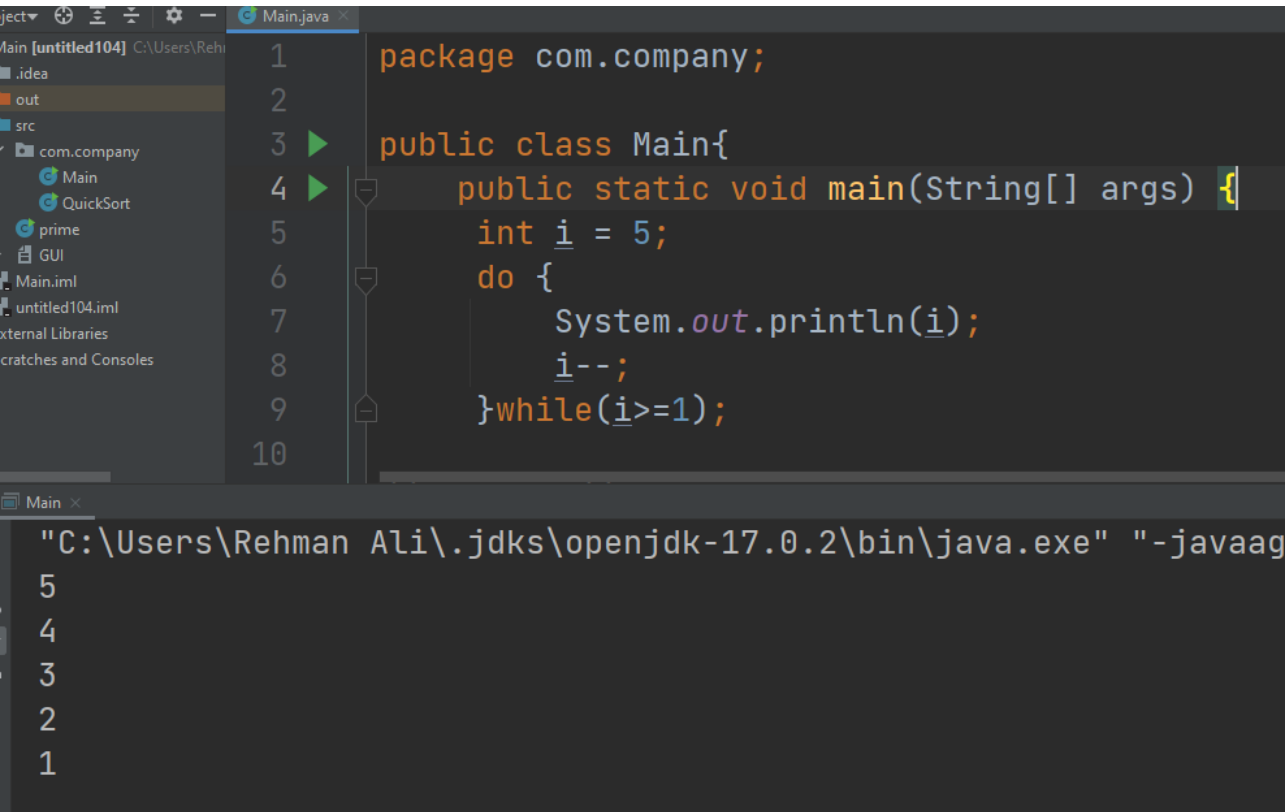


```
1 package com.company;
2 public class Main {
3
4     public static void main(String args[]) {
5         int i= 1;
6         do{
7             System.out.println("rehman");
8             i++;
9         } while(i<=10);
10    }
11 }
```

The screenshot shows the IntelliJ IDEA IDE with a Java file named Main.java. The code is a do-while loop that prints the text "rehman" 10 times. The console output at the bottom shows the command to run the program and the resulting output: rehman, rehman, rehman.

Display text repetition

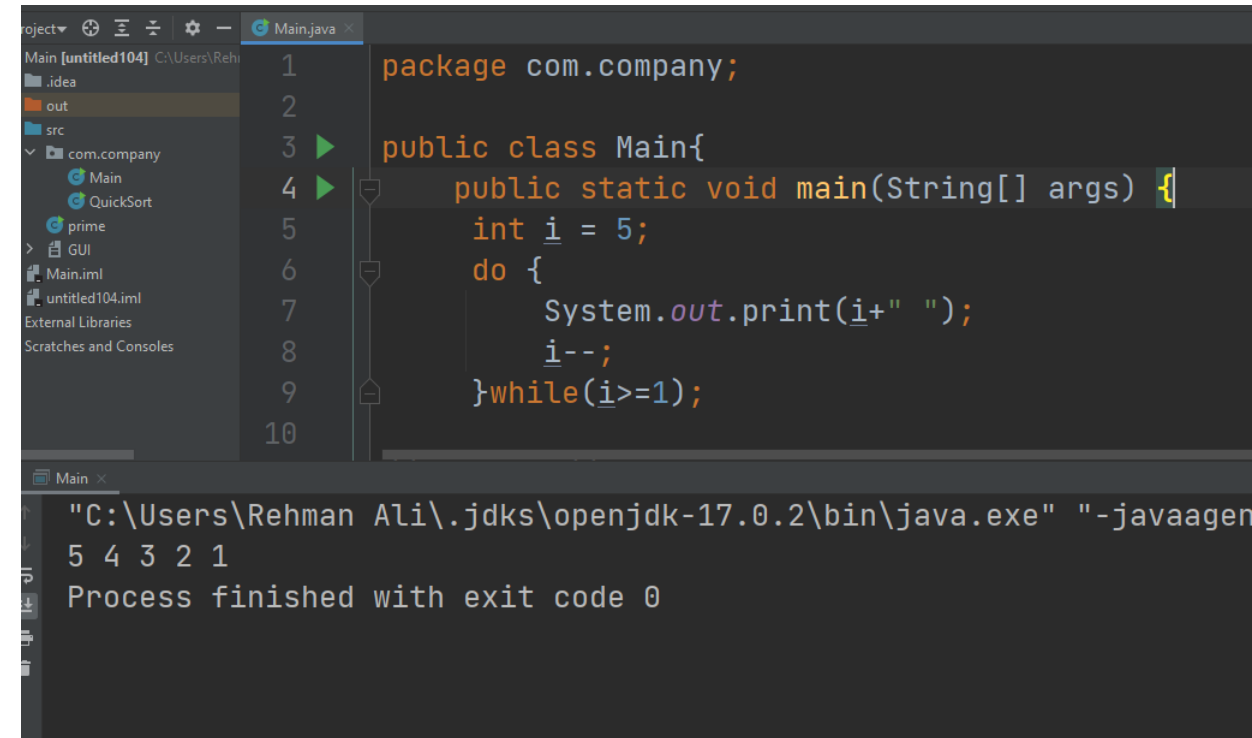
JAVA DO-WHILE LOOP PROGRAMS



```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         int i = 5;
6         do {
7             System.out.println(i);
8             i--;
9         }while(i>=1);
10    }
```

The console output shows the numbers 5, 4, 3, 2, 1 printed vertically.

Display Natural Numbers 1 to 5

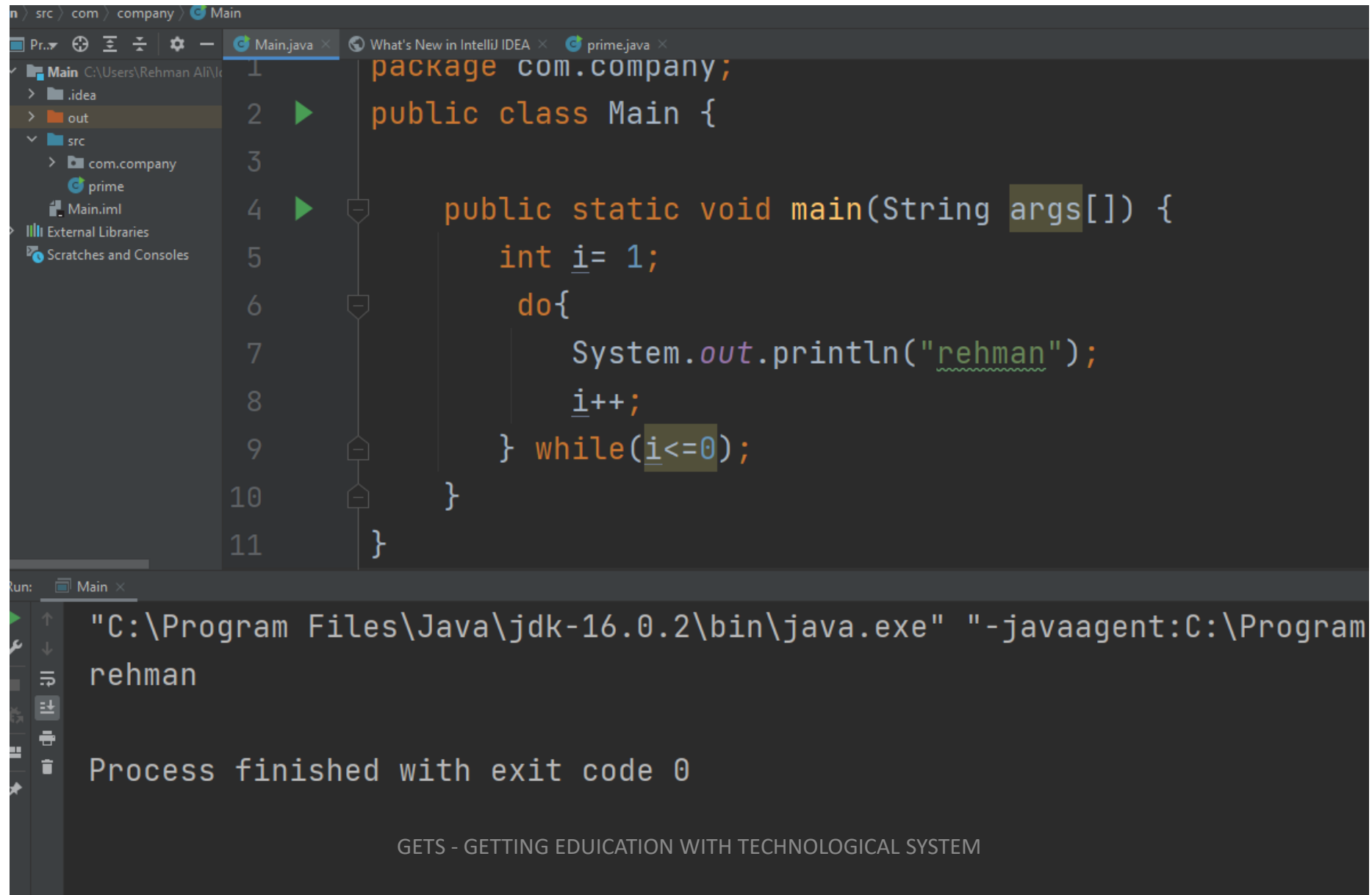


```
1 package com.company;
2
3 public class Main{
4     public static void main(String[] args) {
5         int i = 5;
6         do {
7             System.out.print(i+" ");
8             i--;
9         }while(i>=1);
10    }
```

The console output shows the numbers 5 4 3 2 1 printed horizontally, followed by the message "Process finished with exit code 0".

Can also Display as horizontal/rows

DO-WHILE LOOP AT LEAST PRINT ONE TIME



The screenshot displays the IntelliJ IDEA IDE. The main editor window shows a Java file named `Main.java` with the following code:

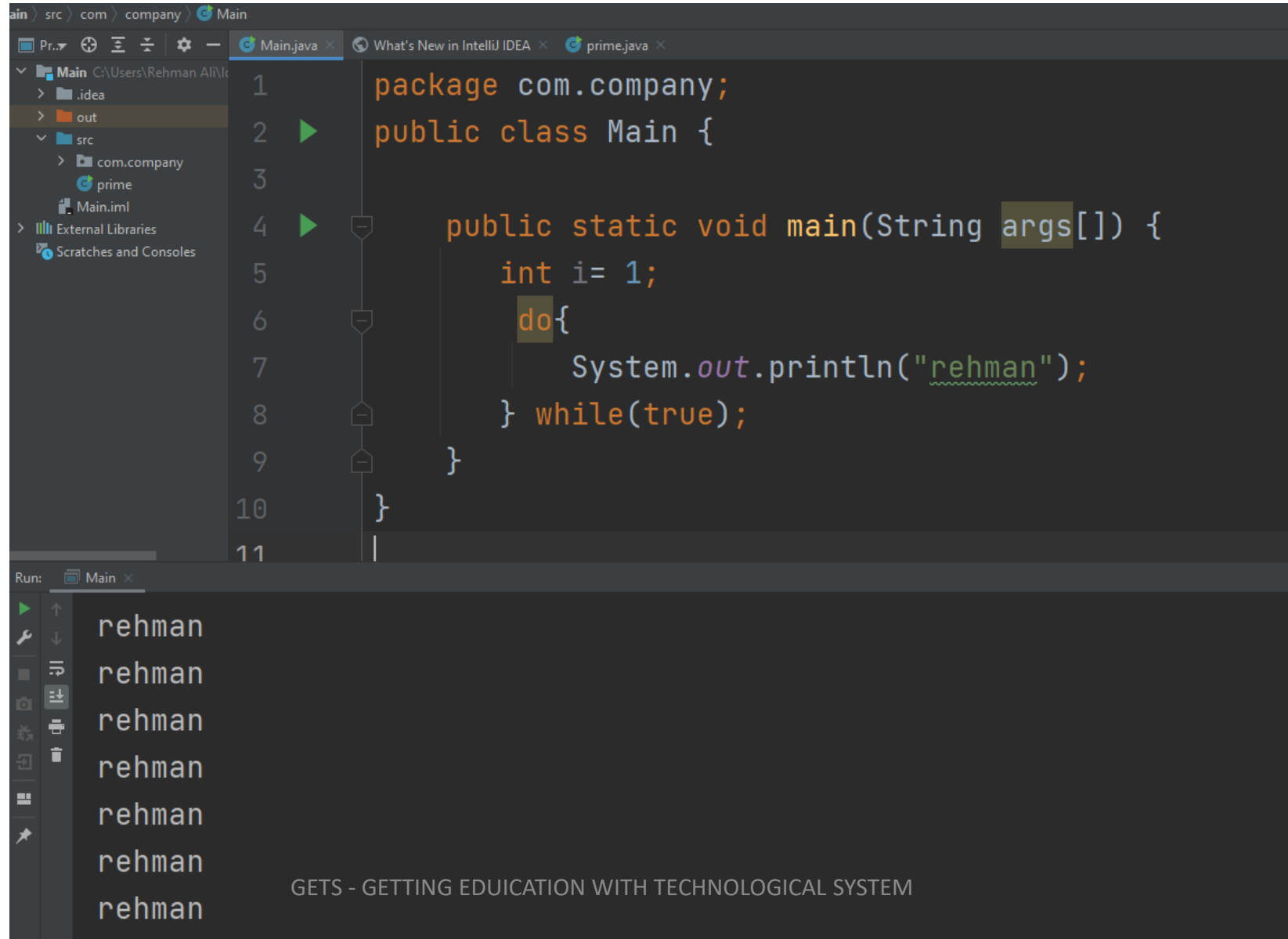
```
1 package com.company;  
2 public class Main {  
3  
4     public static void main(String args[]) {  
5         int i= 1;  
6         do{  
7             System.out.println("rehman");  
8             i++;  
9         } while(i<=0);  
10    }  
11 }
```

The code is syntactically incorrect as the loop condition `i <= 0` is never true, yet the program runs successfully. The Run window at the bottom shows the command executed and the output:

```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program  
rehman  
Process finished with exit code 0
```

The IDE interface includes a Project View on the left showing the project structure, a Run toolbar on the bottom left, and tabs for the code editor and the Run window.

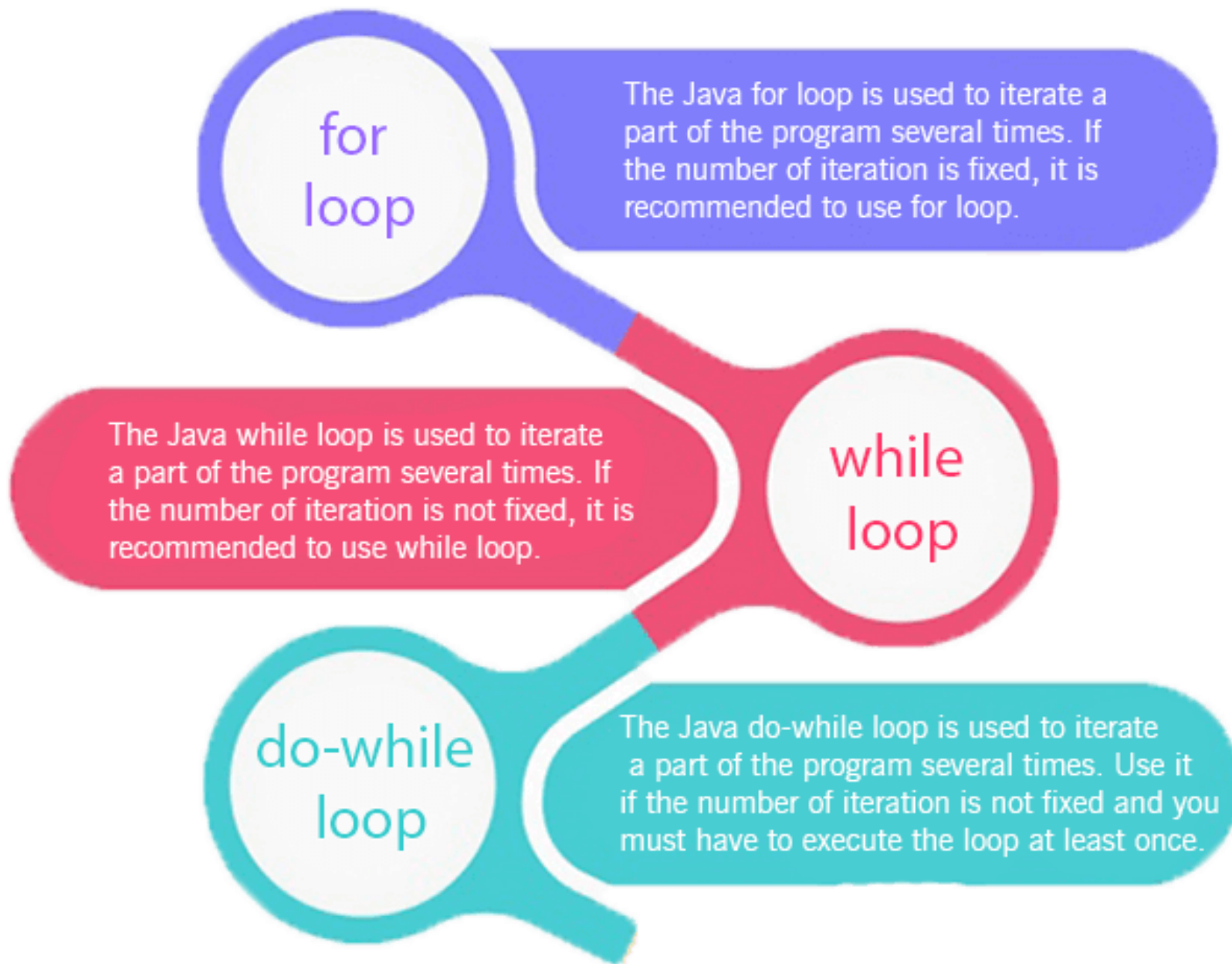
JAVA DO-WHILE INFINITIVE LOOP



The screenshot displays the IntelliJ IDEA IDE with a Java project named 'Main'. The source code in 'Main.java' is as follows:

```
1 package com.company;  
2 public class Main {  
3  
4     public static void main(String args[]) {  
5         int i= 1;  
6         do{  
7             System.out.println("rehman");  
8         } while(true);  
9     }  
10 }  
11
```

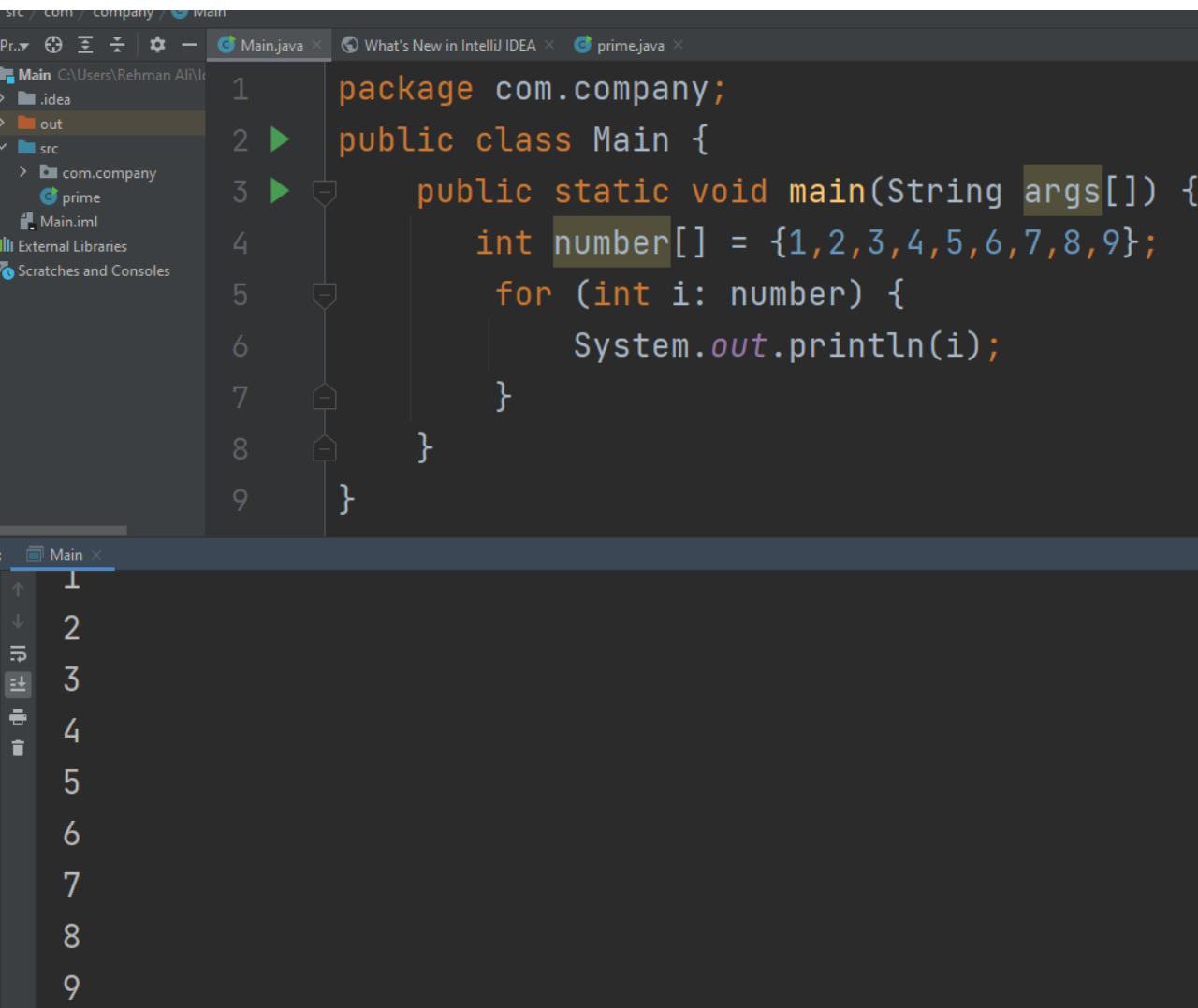
The 'Run' window at the bottom shows the output of the program, which is the word 'rehman' printed six times, demonstrating the infinite loop.



JAVA FOR-EACH LOOP

1. The for-each loop is used **to iterate through elements of arrays and collections.**
2. It is a newer way with lesser code to iterate over an array. It makes the code more easier.

JAVA FOR-EACH LOOP PROGRAMS

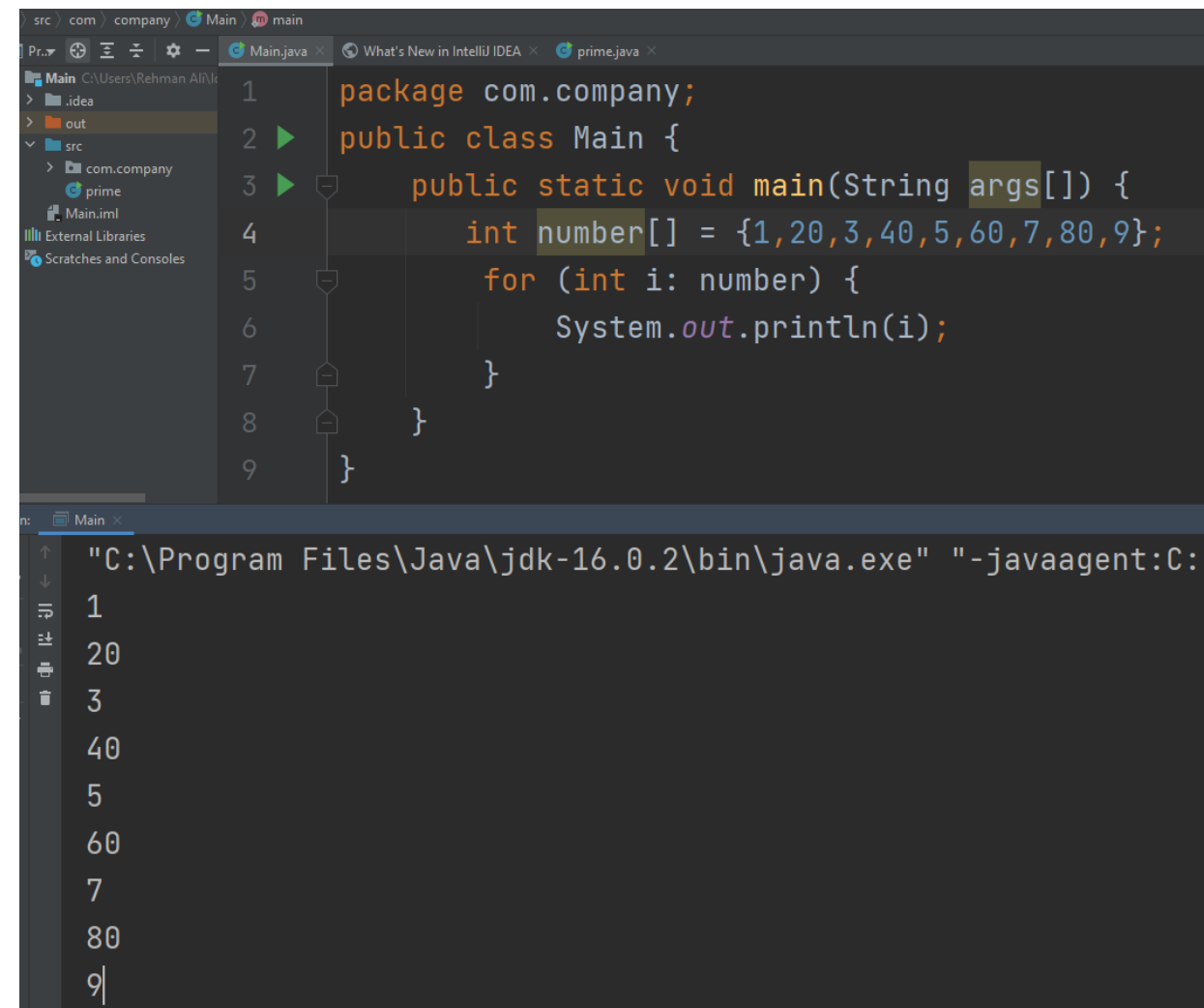


The screenshot shows the IntelliJ IDEA IDE with a project named 'Main'. The 'Main.java' file is open, displaying the following code:

```
1 package com.company;
2 public class Main {
3     public static void main(String args[]) {
4         int number[] = {1,2,3,4,5,6,7,8,9};
5         for (int i: number) {
6             System.out.println(i);
7         }
8     }
9 }
```

The 'Run' button (a green play icon) is visible next to line 3. Below the code editor, the 'Run' console is open, showing the output of the program:

```
1
2
3
4
5
6
7
8
9
```



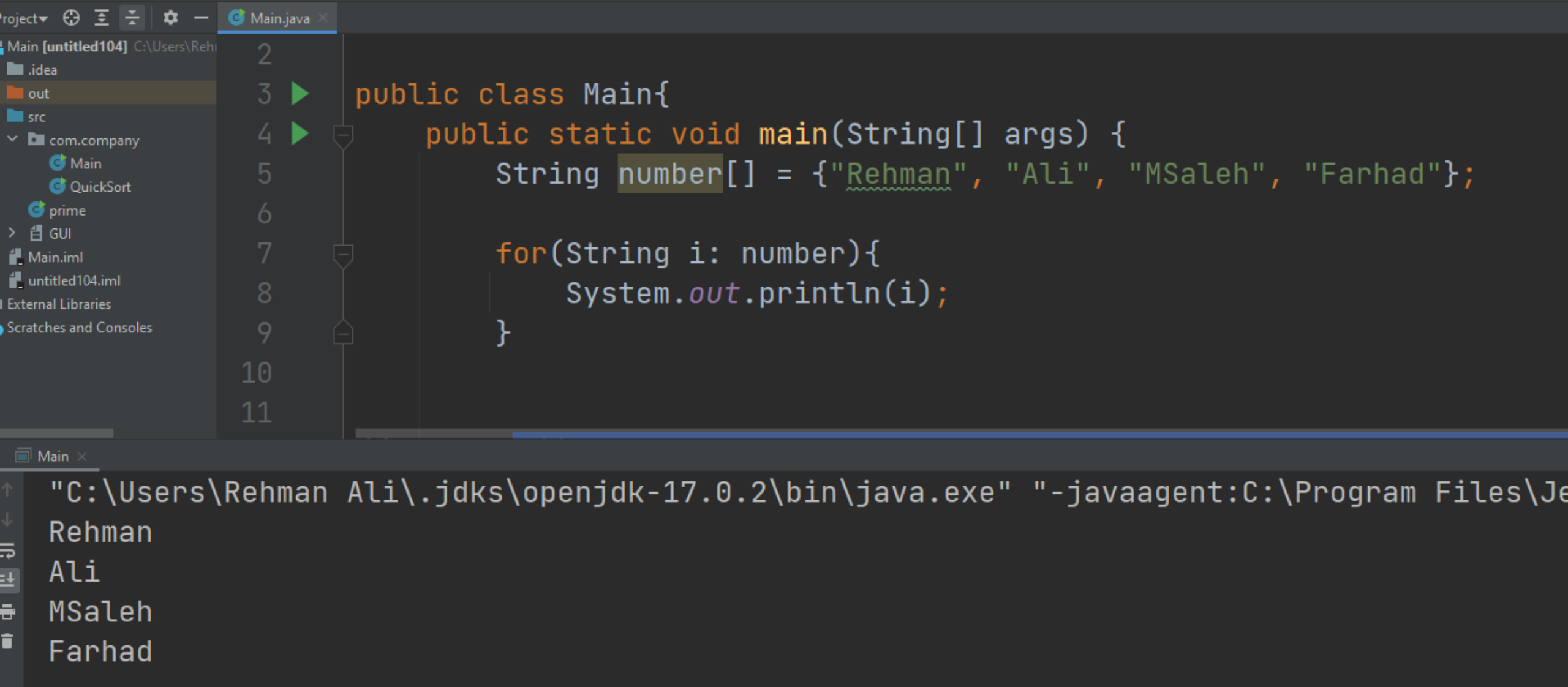
The screenshot shows the IntelliJ IDEA IDE with a project named 'Main'. The 'Main.java' file is open, displaying the following code:

```
1 package com.company;
2 public class Main {
3     public static void main(String args[]) {
4         int number[] = {1,20,3,40,5,60,7,80,9};
5         for (int i: number) {
6             System.out.println(i);
7         }
8     }
9 }
```

The 'Run' button (a green play icon) is visible next to line 3. Below the code editor, the 'Run' console is open, showing the output of the program:

```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:
1
20
3
40
5
60
7
80
9
```

JAVA FOR-EACH LOOP PROGRAMS



The screenshot shows an IDE with a project named 'Main [untitled104]' located at 'C:\Users\Rehman'. The project structure on the left includes a 'src' folder with a 'com.company' package containing 'Main' and 'QuickSort' classes, and a 'GUI' folder. The 'Main.java' file is open, showing the following code:

```
2  
3 public class Main{  
4     public static void main(String[] args) {  
5         String number[] = {"Rehman", "Ali", "MSaleh", "Farhad"};  
6  
7         for(String i: number){  
8             System.out.println(i);  
9         }  
10  
11
```

The output console at the bottom shows the command executed: `"C:\Users\Rehman Ali\.jdk\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\J...` and the resulting output: `Rehman`, `Ali`, `MSaleh`, and `Farhad`.

