

Using Print Format

- Using **format** to print at the console
- Examples:

```
String name = "George";  
int age = 5;  
System.out.printf("Name: %s, Age: %d", name, age);  
// Name: George, Age: 5
```

Placeholder **%s** stands for string and corresponds to **name**

Placeholder **%d** stands for integer number and corresponds to **age**

Formatting Numbers in Placeholders

- **D** – format number to certain digits with leading zeros
- **F** – format floating point number with certain digits after the decimal point
- Examples:

```
int percentage = 55;  
double grade = 5.5334;  
System.out.printf("%03d", percentage);    // 055  
System.out.printf("%.2f", grade);          // 5.53
```

Using String.format

- Using **String.format** to create a string by pattern
- Examples:

```
String name = "George";  
int age = 5;  
String result = String.format("Name: %s,  
                             Age: %d", name, age);  
System.out.println(result);  
//Name: George, Age 5
```

Comparison Operators

Operator	Notation in Java
Equals	<code>==</code>
Not Equals	<code>!=</code>
Greater Than	<code>></code>
Greater Than or Equals	<code>>=</code>
Less Than	<code><</code>
Less Than or Equals	<code><=</code>

Comparing Numbers

- Values can be compared:

```
int a = 5;  
int b = 10;  
  
System.out.println(a < b);           // true  
System.out.println(a > 0);           // true  
System.out.println(a > 100);         // false  
System.out.println(a < a);           // false  
System.out.println(a <= 5);          // true  
System.out.println(b == 2 * a);      // true
```

Logical Operators

- Logical operators give us the ability to write multiple conditions in one **if** statement
- They return a boolean value and compare boolean values

Operator	Notation in Java	Example
Logical NOT	!	!false -> true
Logical AND	&&	true && false -> false
Logical OR		true false -> true

Example: Divisible by 3

- Print the numbers from 1 to 100, that are divisible by 3

```
for (int i = 3; i <= 100; i += 3) {  
    System.out.println(i);  
}
```

3

- You can use "fori" live template in IntelliJ



```
for (int i = 0; i < ; i++) {  
    }  
}
```

Do ... While Loop

- Similar to the **while** loop, but always executes at least once:

```
int i = 1;  
do {  
    System.out.println(i);  
    i++;  
} while (i <= 10);
```

Initial value

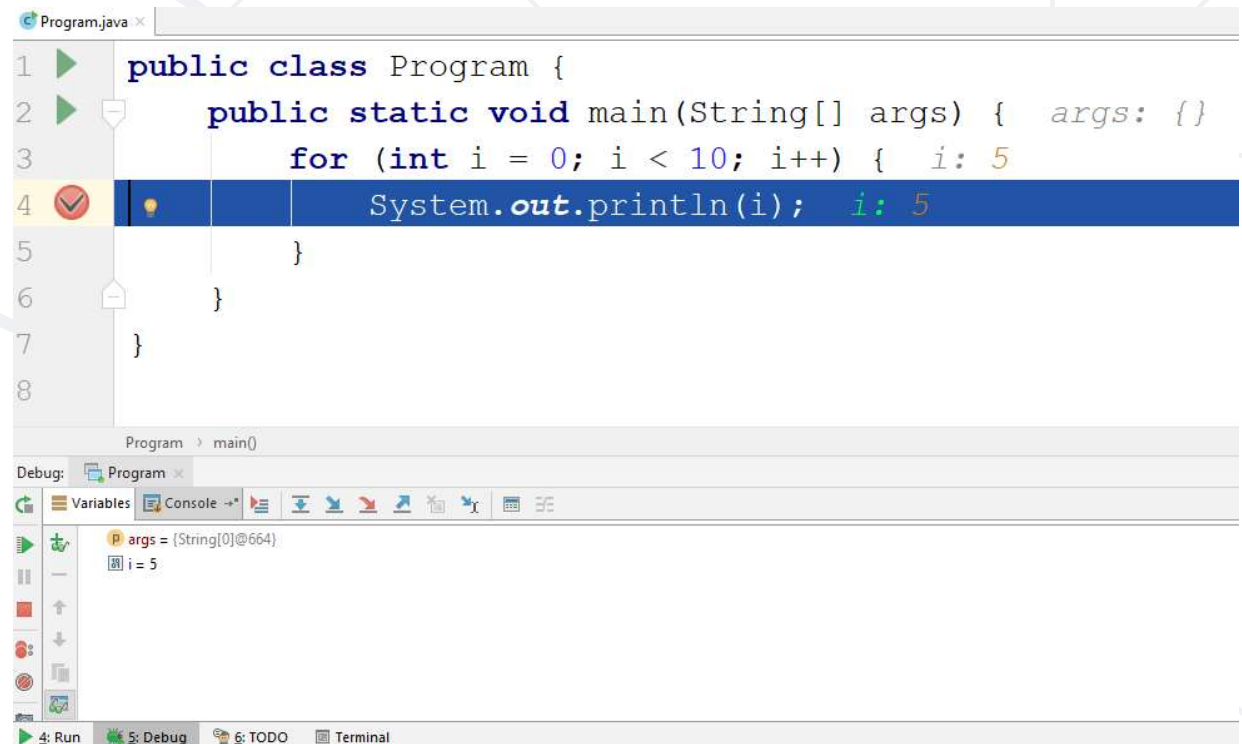
Loop body

Increment
the counter

Condition

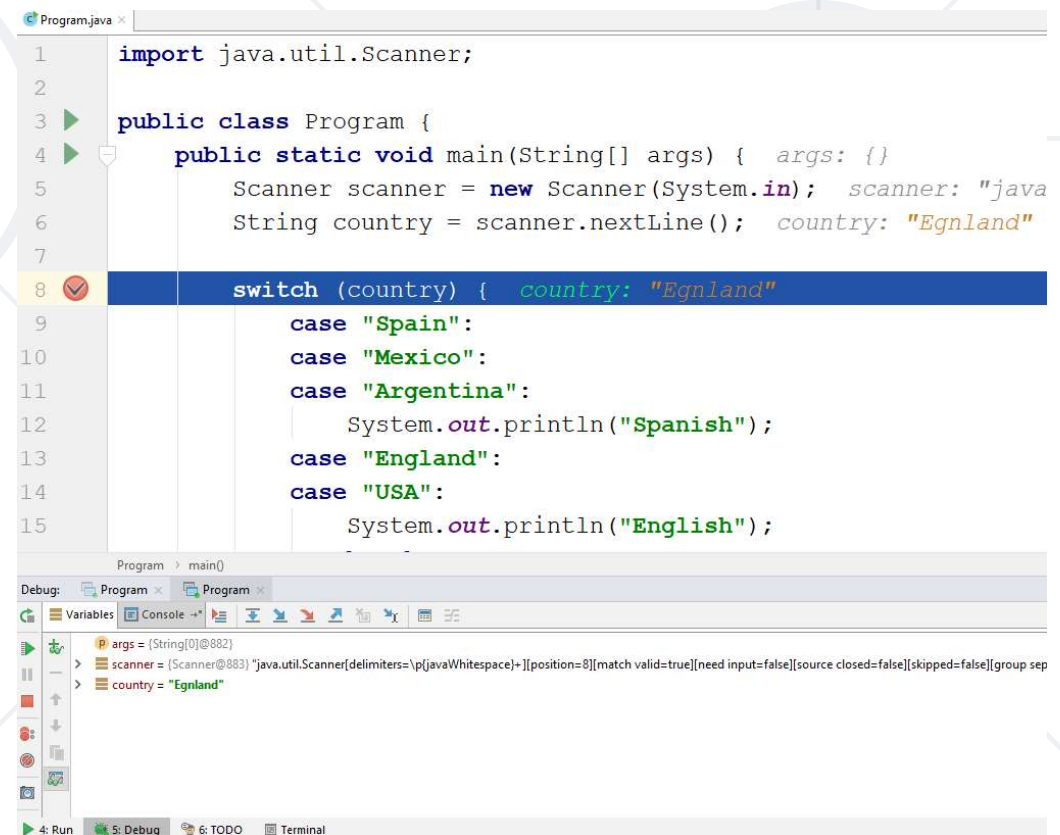
Debugging in IntelliJ

- IntelliJ has a built-in **debugger**
- It provides:
 - **Breakpoints**
 - Ability to **trace** the code execution
 - Ability to **inspect** variables at runtime



Using the Debugger in IntelliJ

- Start without Debugger: **[Ctrl+Shift+F10]**
- Toggle a breakpoint: **[Ctrl+F8]**
- Start with the Debugger: **[Alt+Shift+F9]**
- Trace the program: **[F8]**
- Conditional breakpoints



The screenshot displays the IntelliJ IDEA IDE. The top pane shows a Java file named `Program.java` with the following code:

```
1 import java.util.Scanner;
2
3 public class Program {
4     public static void main(String[] args) { args: {}
5         Scanner scanner = new Scanner(System.in); scanner: "java
6         String country = scanner.nextLine(); country: "Egnland"
7
8         switch (country) { country: "Egnland"
9             case "Spain":
10             case "Mexico":
11             case "Argentina":
12                 System.out.println("Spanish");
13             case "England":
14             case "USA":
15                 System.out.println("English");
```

A breakpoint is set on line 8, indicated by a red circle with a white checkmark. The bottom pane shows the `Debug` console with the following variables:

```
args = [String[]@882]
scanner = (Scanner@883) "java.util.Scanner[delimiters=\p(javaWhitespace)+][position=8][match valid=true][need input=false][source closed=false][skipped=false][group sep]
country = "Egnland"
```