

Java Introduction

Basic Syntax , I/O, Conditions, Loops and Debugging



Software University
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

Table of Contents

1. Introduction and Basic Syntax
2. Comparison Operators
3. The if-else / switch-case Statement
4. Logical Operators
5. Loops
6. Debugging and Troubleshooting



sli.do

#fund-java



Introduction and Basic Syntax

Java – Introduction

- **Java** is modern, flexible, general-purpose programming language
- **Object-oriented** by nature, statically-typed, compiled



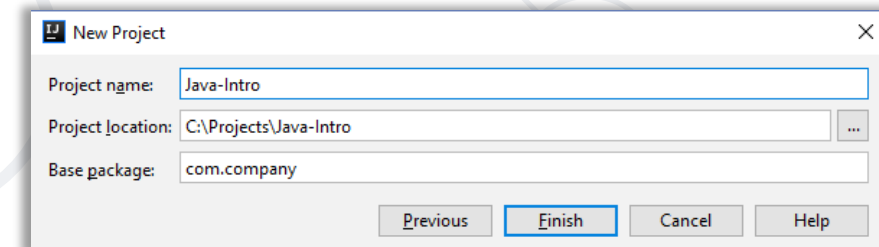
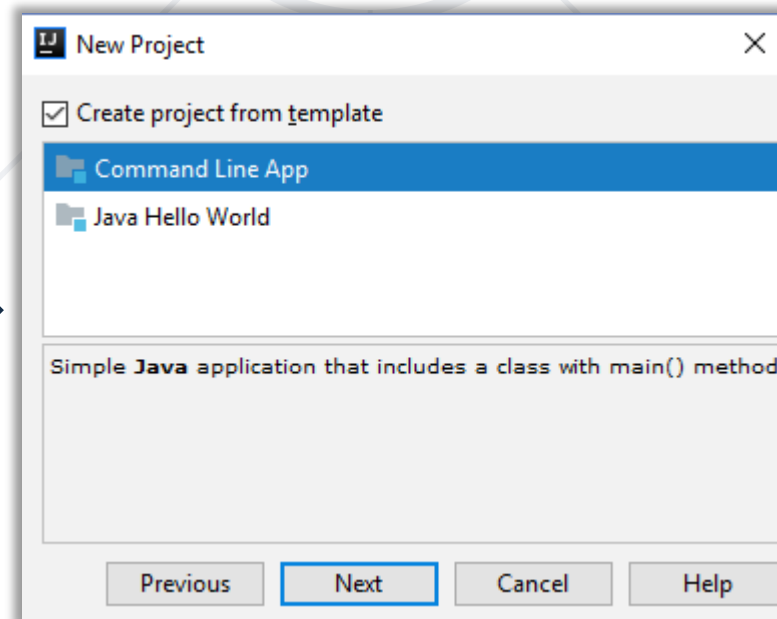
```
static void main(String[] args) {  
    //Source Code  
}
```

Program
starting
point

- In this course will use Java Development Kit (JDK) 13

Using IntelliJ Idea

- **IntelliJ Idea** is powerful IDE for Java and other languages
- Create a project



Declaring Variables

- Defining and Initializing variables

```
{data type / var} {variable name} = {value};
```

- Example:

```
int number = 5;
```

Variable name

Data type

Variable value





Console I/O

Reading from and Writing to the Console

Reading from the Console

- We can **read/write** to the console, using the **Scanner** class
- Import the **java.util.Scanner** class

```
import java.util.Scanner;  
...  
Scanner sc = new Scanner(System.in);
```

- Reading input from the console using


```
String name = sc.nextLine();
```

Returns **string**



Converting Input from the Console

- `scanner.nextLine()` returns a **String**
- Convert the string to number by **parsing**:



```
import java.util.Scanner;

...

Scanner sc = new Scanner(System.in);
String name = sc.nextLine();
int age = Integer.parseInt(sc.nextLine());
double salary = Double.parseDouble(sc.nextLine());
```

- We can **print** to the console, using the **System** class
- Writing output to the console:
 - **System.out.print()**
 - **System.out.println()**

```
System.out.print("Name: ");  
String name = scanner.nextLine();  
System.out.println("Hi, " + name);  
// Name: George  
// Hi, George
```

- 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** 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
```

Problem: Student Information

- You will be given 3 input lines:
 - Student Name, Age and Average Grade
- Print the input in the following format:
 - "Name: {name}, Age: {age}, Grade {grade}"
 - Format the grade to 2 decimal places

John
15
5.40



Name: John, Age: 15, Grade: 5.40

Check your solution here: <https://judge.softuni.bg/Contests/1190/>

Solution: Student Information

```
import java.util.Scanner;

...

Scanner sc = new Scanner(System.in);
String name = sc.nextLine();
int age = Integer.parseInt(sc.nextLine());
double grade = Double.parseDouble(sc.nextLine());

System.out.printf("Name: %s, Age: %d, Grade: %.2f",
                  name, age, grade);
```




Comparison Operators

Comparison Operators

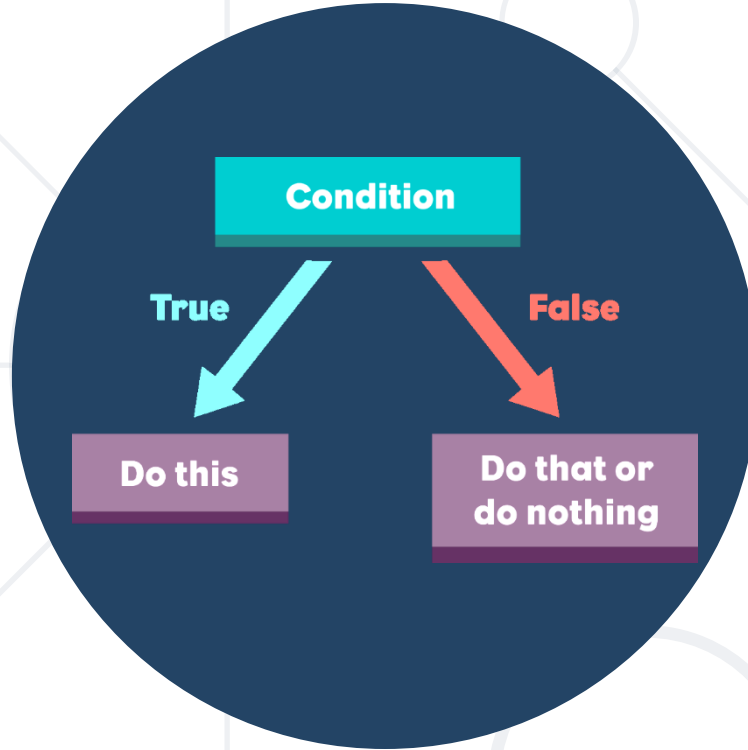
Operator	Notation in Java
Equals	<code>==</code>
Not Equals	<code>!=</code>
Greater Than	<code>></code>
Greater Than or Equals	<code>>=</code>
Less Than	<code><</code>
Less Than or Equals	<code><=</code>

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
```



The If-else Statement

Implementing Control-Flow Logic

- The simplest conditional statement
 - Test for a condition
- Example: Take as an input a grade and check if the student has passed the exam (grade \geq 3.00)

```
double grade = Double.parseDouble(sc.nextLine());  
if (grade  $\geq$  3.00) {  
    System.out.println("Passed!");  
}
```

In Java the opening bracket stays on the same line

The If-else Statement

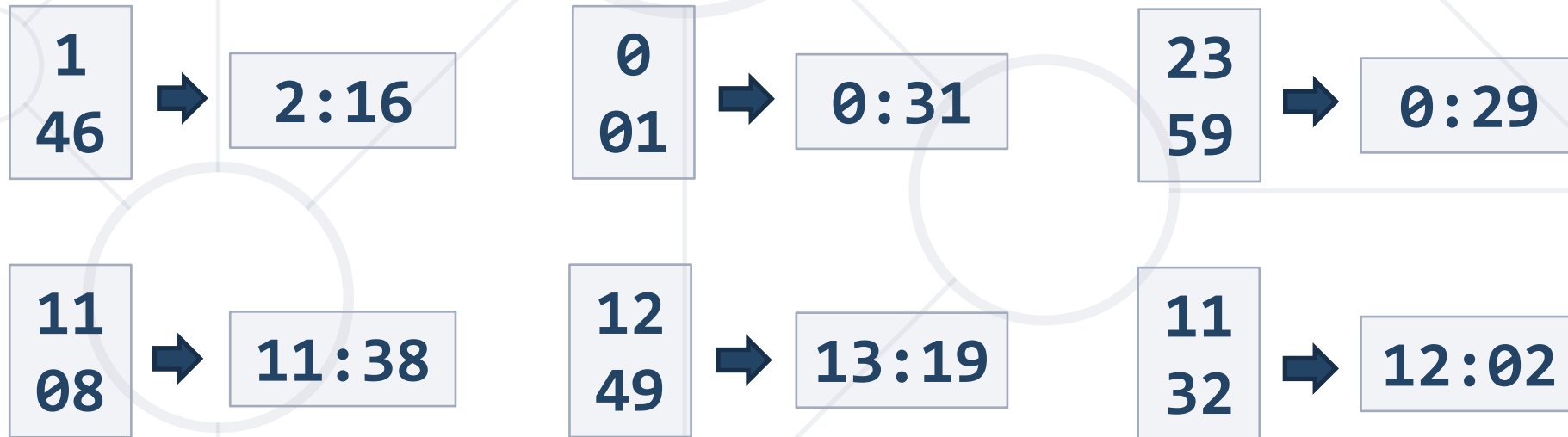
- Executes **one branch** if the condition is **true** and **another**, if it is **false**
- Example: **Upgrade** the last example, so it prints "**Failed!**", if the mark is lower than 3.00:

The **else** keyword stays on a new line

```
if (grade >= 3.00) {  
    System.out.println("Passed!");  
} else {  
    // TODO: Print the message  
}
```

Problem: I Will Be Back in 30 Minutes

- Write a program that reads hours and minutes from the console and calculates the time after 30 minutes
 - The hours and the minutes come on separate lines
- Example:



Solution: I Will Be Back in 30 Minutes (1)

```
int hours = Integer.parseInt(sc.nextLine());  
int minutes = Integer.parseInt(sc.nextLine()) + 30;  
  
if (minutes > 59) {  
    hours += 1;  
    minutes -= 60;  
}  
  
// Continue on the next slide
```


Solution: I Will Be Back in 30 Minutes (2)

```
if (hours > 23) {  
    hours = 0;  
}  
if (minutes < 10) {  
    System.out.printf("%d:%02d%n", hours, minutes);  
} else {  
    System.out.printf("%d:%d", hours, minutes);  
}
```

%n goes on
the next line



The Switch-Case Statement

Simplified If-else-if-else

The Switch-case Statement

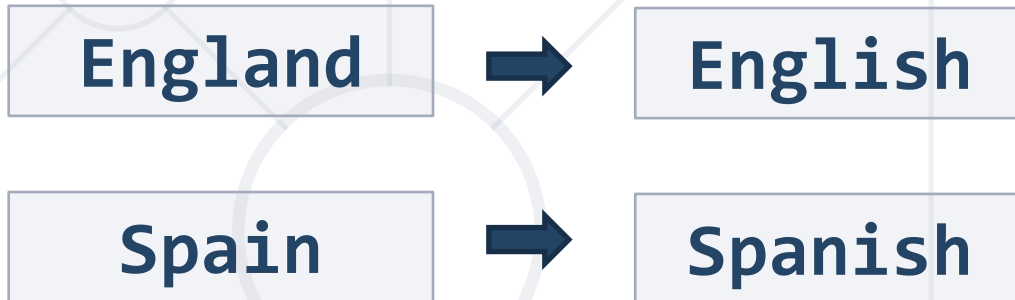
- Works as sequence of **if-else** statements
- Example: read input a number and print its corresponding month:

```
int month = Integer.parseInt(sc.nextLine());
switch (month) {
    case 1: System.out.println("January"); break;
    case 2: System.out.println("February"); break;
    // TODO: Add the other cases
    default: System.out.println("Error!"); break;
}
```

Check your solution here: <https://judge.softuni.bg/Contests/1190/>

Problem: Foreign Languages

- By given country print its typical language:
 - English -> England, USA
 - Spanish -> Spain, Argentina, Mexico
 - other -> unknown



Solution: Foreign Languages

```
//TODO: Read the input  
switch (country) {  
    case "USA":  
    case "England": System.out.println("English"); break;  
    case "Spain":  
    case "Argentina":  
    case "Mexico": System.out.println("Spanish"); break;  
    default: System.out.println("unknown"); break;  
}
```

A background network diagram consisting of a grid of light gray lines intersecting at various points. Some of these intersections are marked with small, empty light gray circles. A larger, solid dark blue circle is centered in the upper half of the image, containing the text '&&'.

&&

Logical Operators

Writing More Complex Conditions

- 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

Problem: Theatre Promotions

- A theatre has the following ticket prices according to the age of the visitor and the type of day. If the age is < 0 or > 122 , print "Error!":

Day / Age	$0 \leq \text{age} \leq 18$	$18 < \text{age} \leq 64$	$64 < \text{age} \leq 122$
Weekday	12\$	18\$	12\$
Weekend	15\$	20\$	15\$
Holiday	5\$	12\$	10\$



Solution: Theatre Promotions (1)

```
String day = sc.nextLine().toLowerCase();
int age = Integer.parseInt(sc.nextLine());
int price = 0;
if (day.equals("weekday")) {
    if ((age >= 0 && age <= 18) || (age > 64 && age <= 122)) {
        price = 12;
    }
    // TODO: Add else statement for the other group
}
// Continue...
```

Solution: Theatre Promotions (2)

```
else if (day.equals("weekend")) {  
    if ((age >= 0 && age <= 18) || (age > 64 && age <= 122)) {  
        price = 15;  
    } else if (age > 18 && age <= 64) {  
        price = 20;  
    }  
} // Continue...
```

Solution: Theatre Promotions (3)

```
else if (day.equals("holiday")){  
    if (age >= 0 && age <= 18)  
        price = 5;  
    // TODO: Add the statements for the other cases  
}  
if (price != 0)  
    System.out.println(price + "$");  
else  
    System.out.println("Error!");
```



Loops

Code Block Repetition

Loop: Definition

- A **loop** is a control statement that repeats the execution of a block of statements. The loop can:
 - **for** loop
 - Execute a code block a fixed number of times
 - **while** and **do...while**
 - Execute a code block while a given condition returns true





For-Loops

Managing the Count of the Iteration

- The for loop executes statements a fixed number of times:

Initial value

End value

Increment

Loop body

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

Executed
at each
iteration

The bracket is
again on the
same line

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);  
}
```



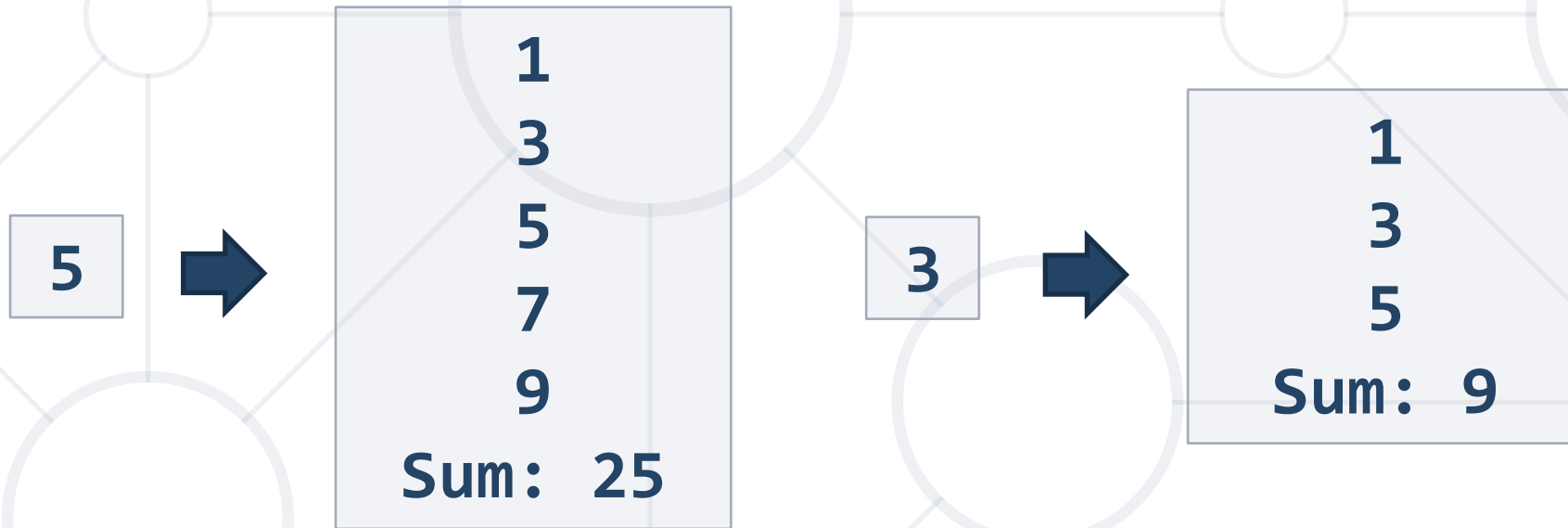
- You can use "**fori**" live template in IntelliJ



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


Problem: Sum of Odd Numbers

- Write a program to print the first **n** odd numbers and their sum



Solution: Sum of Odd Numbers

```
int n = Integer.parseInt(sc.nextLine());  
int sum = 0;  
  
for (int i = 1; i <= n; i++) {  
    System.out.println(2 * i - 1);  
    sum += 2 * i - 1;  
}  
  
System.out.printf("Sum: %d", sum);
```



While Loops

Iterations While a Condition is True

While Loops

- Executes commands while the condition is true:

Initial value

Condition

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

Loop body

Increment the counter

Problem: Multiplication Table

- Print a table holding $\text{number} \times 1$, $\text{number} \times 2$, ..., $\text{number} \times 10$

```
int number = Integer.parseInt(sc.nextLine());
int times = 1;
while (times <= 10) {
    System.out.printf("%d X %d = %d\n",
                      number, times, number * times);
    times++;
}
```



Do...While Loop

Execute a Piece of Code One or More Times

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

Problem: Multiplication Table 2.0

- Upgrade your program and take the initial times from the console

```
int number = Integer.parseInt(sc.nextLine());  
int times = Integer.parseInt(sc.nextLine());  
do {  
    System.out.printf("%d X %d = %d%n",  
                      number, times, number * times);  
    times++;  
} while (times <= 10);
```




Debugging the Code

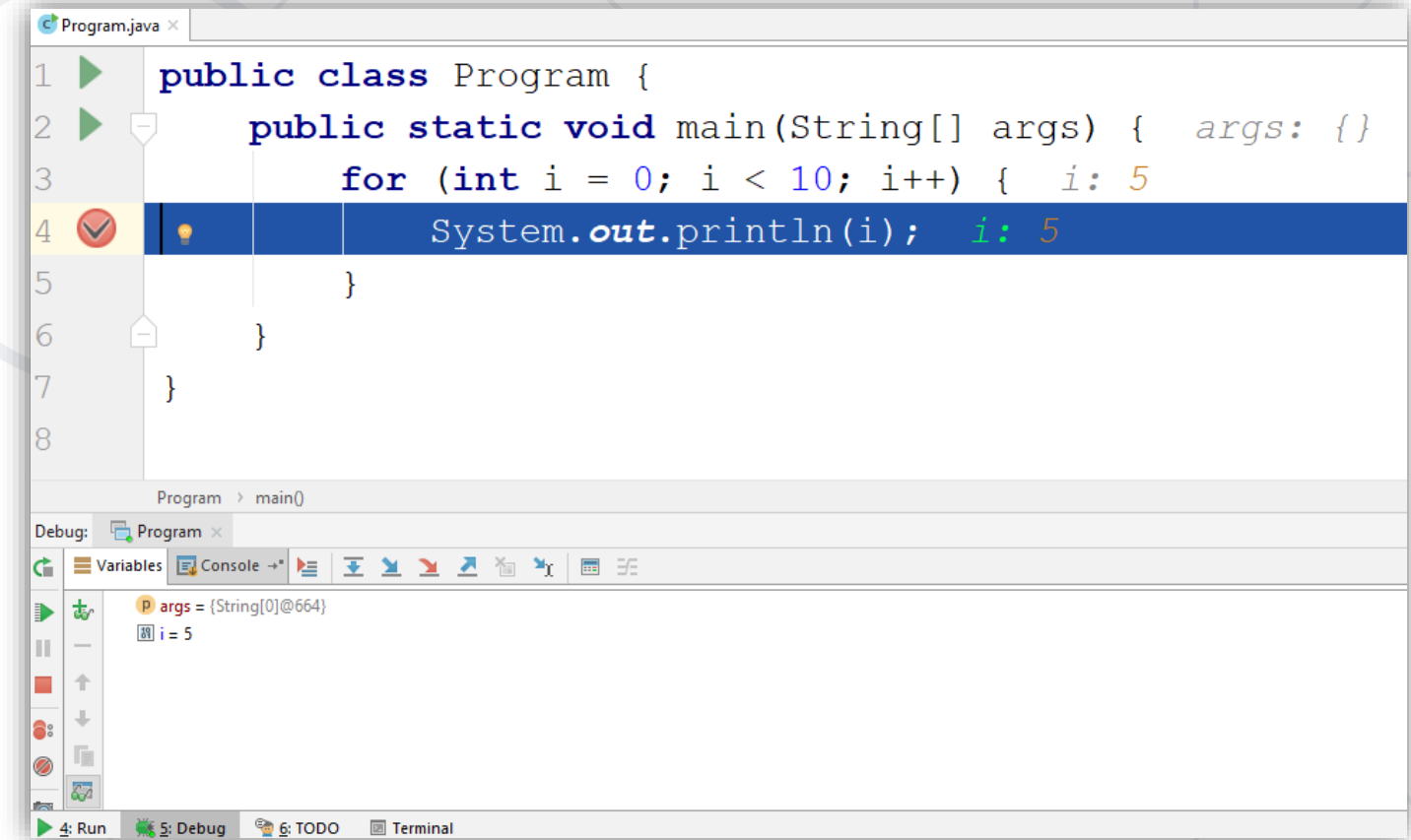
Using the IntelliJ Debugger

- The process of **debugging application** includes:
 - Spotting an error
 - Finding the lines of code that cause the error
 - Fixing the error in the code
 - Testing to check if the error is gone and no new errors are introduced
- Iterative and continuous process



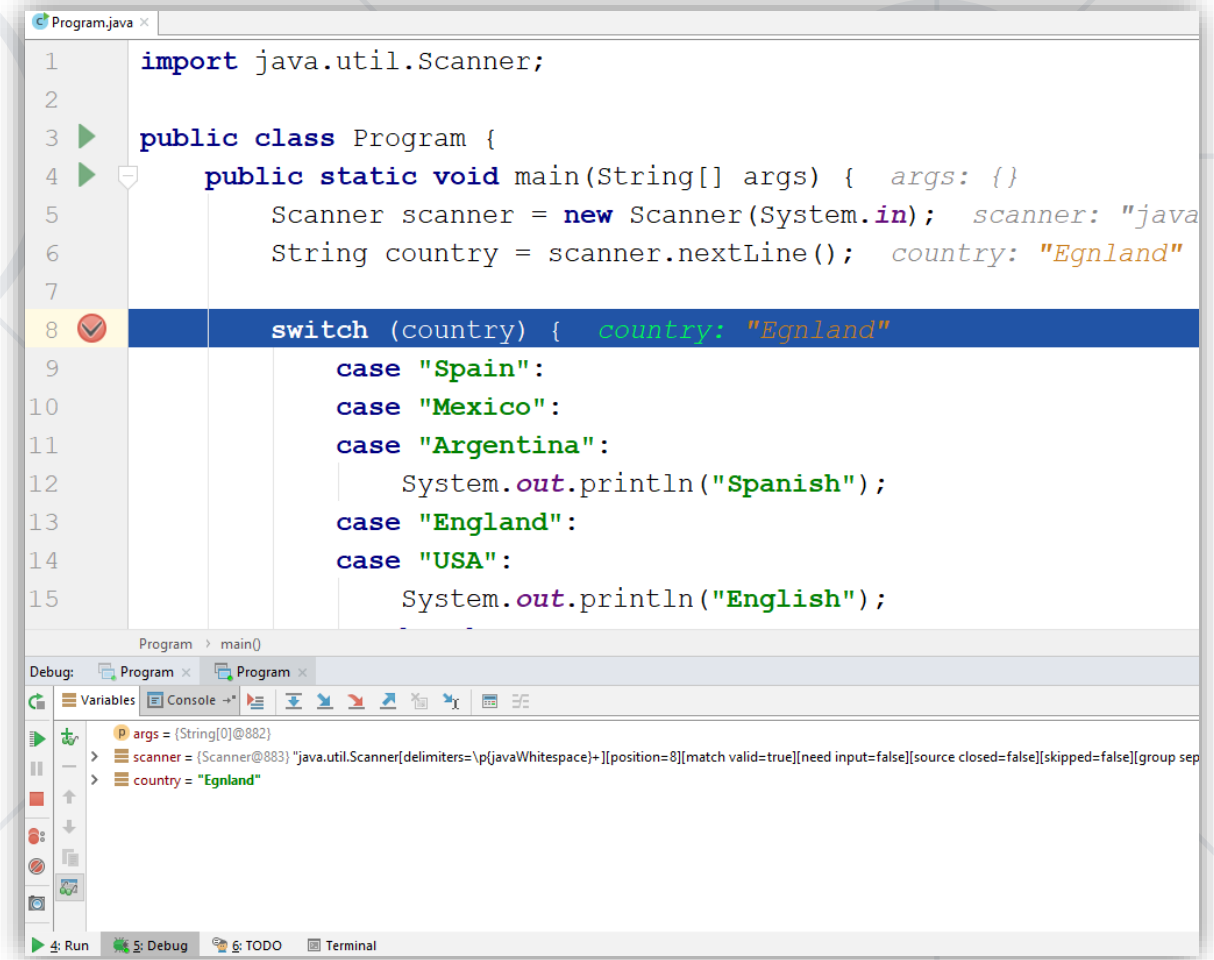
Debugging in IntelliJ

- 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



```
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");
```

Program > main()

Debug: Program x Program x

Variables Console

args = {String[0]@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"

4: Run 5: Debug 6: TODO 7: Terminal

Problem: Find and Fix the Bugs in the Code

- A program aims to print the first **n** odd numbers and their sum

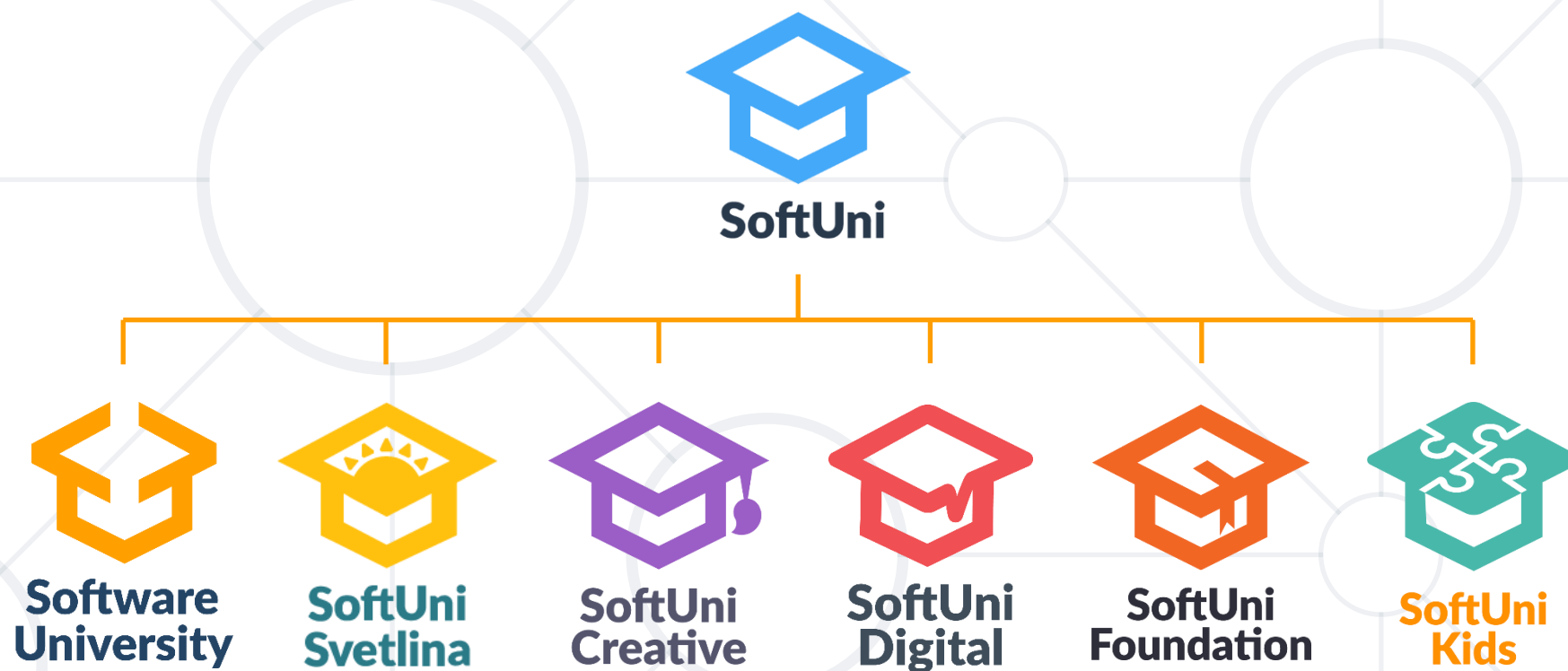
```
Scanner sc = new Scanner(System.in);
int n = Integer.parseInt(sc.nextLine());
int sum = 1;
for (int i = 0; i <= n; i++) {
    System.out.print(2 * i + 1);
    sum += 2 * i;
}
System.out.printf("Sum: %d\n", sum);
```

10

- Declaring **Variables**
- **Reading** from / **Printing** to the **Console**
- **Conditional Statements** allow implementing programming logic
- **Loops** repeat code block multiple times
- Using the debugger



Questions?



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

