Java Developing Environment Setup
Introduction to Java Programming Language

Desenvolvimento de Software e Sistemas Móveis (DSSMV)
Licenciatura em Engenharia de Telecomunicações e Informática
LETI/ISEP

2025/26

Paulo Baltarejo Sousa and Carlos Filipe Freitas
{pbs,caf}@isep.ipp.pt

Disclaimer

Material and Slides

Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

Outline

1 **IntelliJ IDEA**
- Create a project

2 **Java Program Basics**
- The Structure of a Simple Program

3 **Object Oriented Programming Basics**
- Classes
- Objects

4 **Fundamental Data Types**
- Numerical
- Operators

5 **Control Statements**
- Decisions
- Loops
- Jump Statements

6 **Input and Output**

7 **Unit Tests**

8 **Using a Debugger**

9 **Bibliography**

IntelliJ IDEA

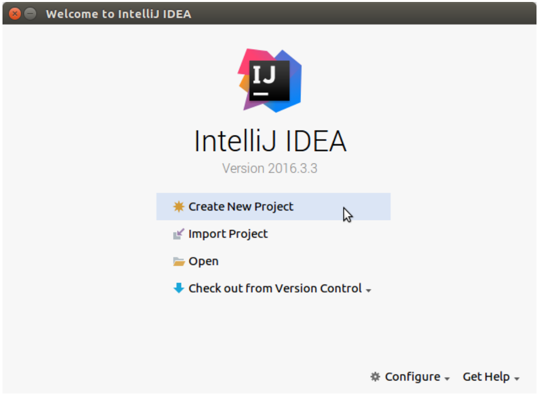
Download

- Go to `https://www.jetbrains.com/idea/`
- Go to download page by clicking on **Download**.
- Choose the **Ultimate** version.
- Request the IntelliJ IDEA key in `https://www.jetbrains.com/shop/eform/students`
    - You have to use your **ISEP email address**.
- Follow the instructions to activate the IntelliJ IDEA.
    - The instructions are sent via email.

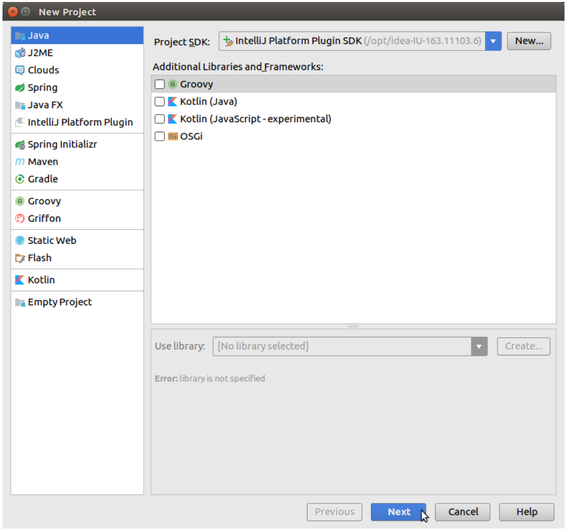
Installation

- Go to `https://www.jetbrains.com/idea/whatsnew/` and following the installation instructions.
  - Enable "Database Tools" and "Android".
- Discover IntelliJ IDEA in `https://www.jetbrains.com/help/idea/discover-intellij-idea.html` and `https://www.jetbrains.com/idea/documentation/`.

Welcome


Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2016.3.3
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help

New Project



Project SDK: IntelliJ Platform Plugin SDK (/opt/idea-IU-163.11103.6)

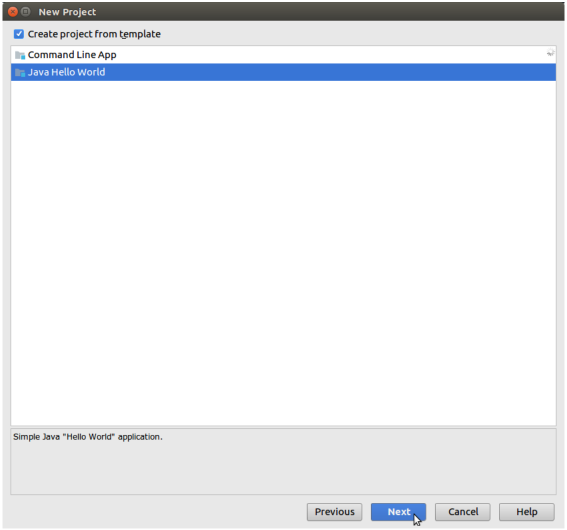
Additional Libraries and Frameworks:

- [ ] Groovy
- [ ] Kotlin (Java)
- [ ] Kotlin (JavaScript - experimental)
- [ ] OSGi

Use library: [No library selected]

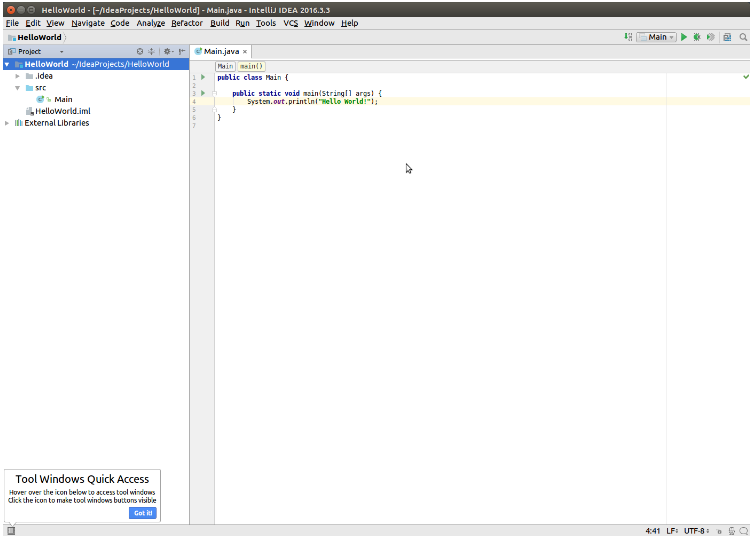
Error: library is not specified

Choosing a template



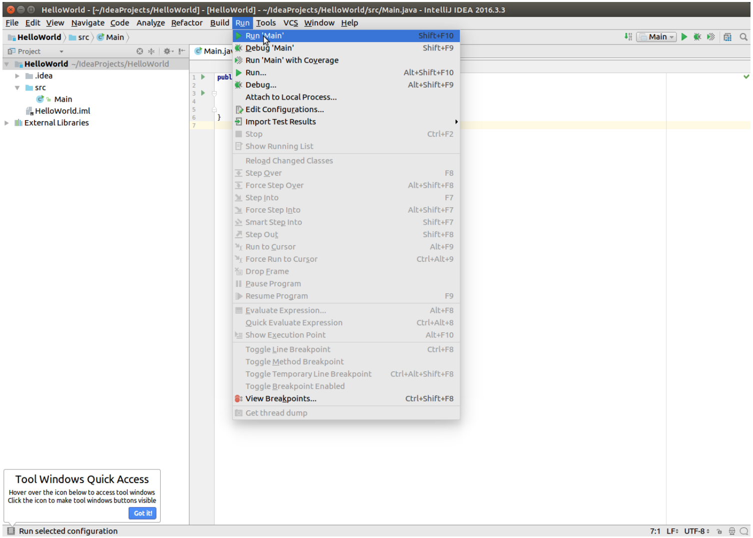
Simple Java "Hello World" application.

IDE

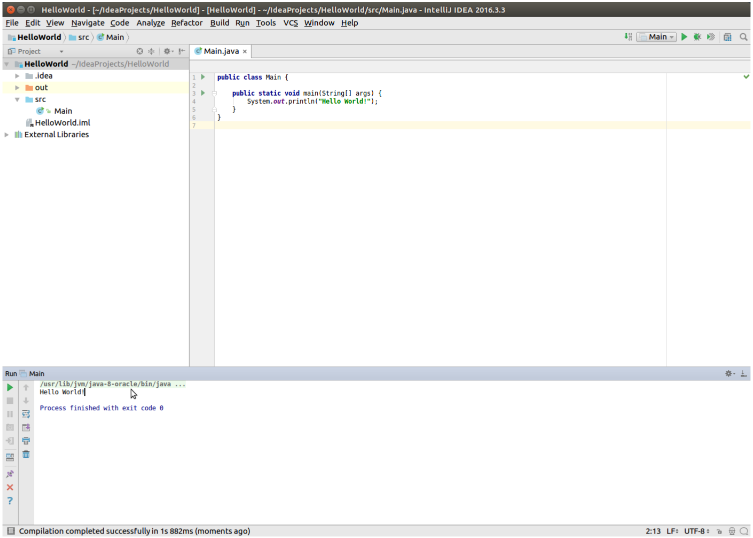

HelloWorld - [~/IdeaProjects/HelloWorld] - Main.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWorld
Project
HelloWorld ~/IdeaProjects/HelloWorld
.idea
src
Main
HelloWorld.iml
External Libraries
Main.java
Main main()
Tool Windows Quick Access
Hover over the icon below to access tool windows
Click the icon to make tool windows buttons visible
Got it!
4:41 LF: UTF-8

```java
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Running


HelloWorld - [~/IdeaProjects/HelloWorld] - [HelloWorld] - ~/IdeaProjects/HelloWorld/src/Main.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWorld src Main
Project
HelloWorld ~/IdeaProjects/HelloWorld
.idea
src
Main
HelloWorld.iml
External Libraries
Main.java
Run 'Main' Shift+F10
Debug 'Main' Shift+F9
Run 'Main' with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Local Process...
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Reload Changed Classes
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Force Step Into Alt+Shift+F7
Smart Step Into Shift+F7
Step Out Shift+F8
Run to Cursor Alt+F9
Force Run to Cursor Ctrl+Alt+9
Drop Frame
Pause Program
Resume Program F9
Evaluate Expression... Alt+F8
Quick Evaluate Expression Ctrl+Alt+8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Method Breakpoint
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
Get thread dump
Tool Windows Quick Access
Hover over the icon below to access tool windows
Click the icon to make tool windows buttons visible
Got it!
Run selected configuration
7:1 LF: UTF-8

Execution


HelloWorld - [~/IdeaProjects/HelloWorld] - [HelloWorld] - ~/IdeaProjects/HelloWorld/src/Main.java - IntelliJ IDEA 2016.3.3
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWorld src Main
Main
Project
Main.java
HelloWorld ~/IdeaProjects/HelloWorld
.idea
out
src
Main
HelloWorld.iml
External Libraries
Run Main

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Hello World!

Process finished with exit code 0
```

Compilation completed successfully in 1s 882ms (moments ago)

2:13 LF: UTF-8

Java Program Basics

Class HelloPrinter

```java
public class HelloPrinter
{
   public static void main(String[] arguments) {
     // Display a greeting in the console window
      System.out.println("Hello, World!");
   }
}
```

- `public class HelloPrinter`
  - Create the `HelloPrinter` class.
    - Every program consists of one or more classes.
    - Every source file can contain at most one public class, and the name of the public class must match the name of the file containing the class.
    - For example, the class `HelloPrinter` must be contained in the `HelloPrinter.java` file.
  - The reserved word `public` denotes that the class is usable by the "public".

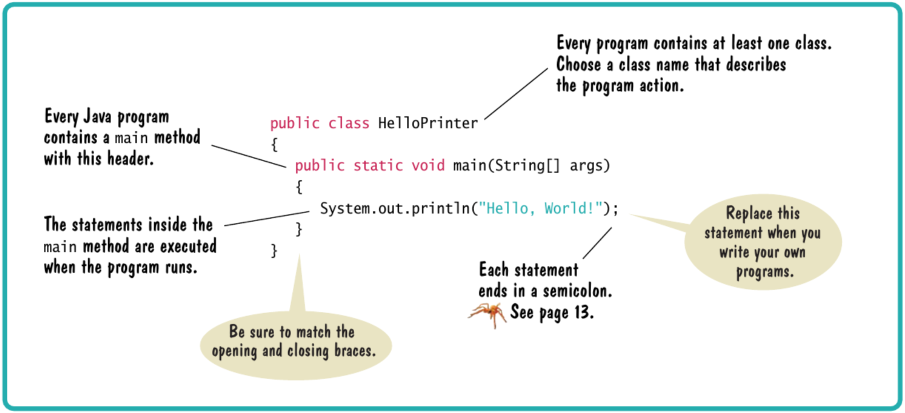
Method `main`

- `public static void main(String[] args)`
  - Every Java application must have a `main` method, which is the entry point.
  - `String[] args` contains command line arguments (passed to the `main` method.
  - The reserved word `static` indicates that the `main` method does not operate on an object.
  - `main` method must always be `static` , because it starts running before the program can create objects.
- `// Display a greeting in the console window`
  - Any text enclosed between `//` and the end of the line is completely ignored by the compiler.
  - Comments are used to explain the program to other programmers or to yourself.

Statements

- The instructions or statements in the body of the `main` method–that is, the statements inside the curly braces (`{ }`) – are executed one by one.
  - Each statement ends in a semicolon (`;`).
- `System.out.println("Hello, World!");`
  - This statement prints a line of text, namely "Hello, World!".
    - The console window is represented in Java by an object called `out`, which it is placed it in the `System` class, which contains useful objects and methods to access system resources.
    - `println` method prints the received parameter, in this case the string "Hello, World!".

Simple program



```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello, World!");
   }
}
```

Every program contains at least one class. Choose a class name that describes the program action.
Every Java program contains a main method with this header.
The statements inside the main method are executed when the program runs.
Replace this statement when you write your own programs.
Each statement ends in a semicolon. See page 13.
Be sure to match the opening and closing braces.

Object Oriented Programming Basics

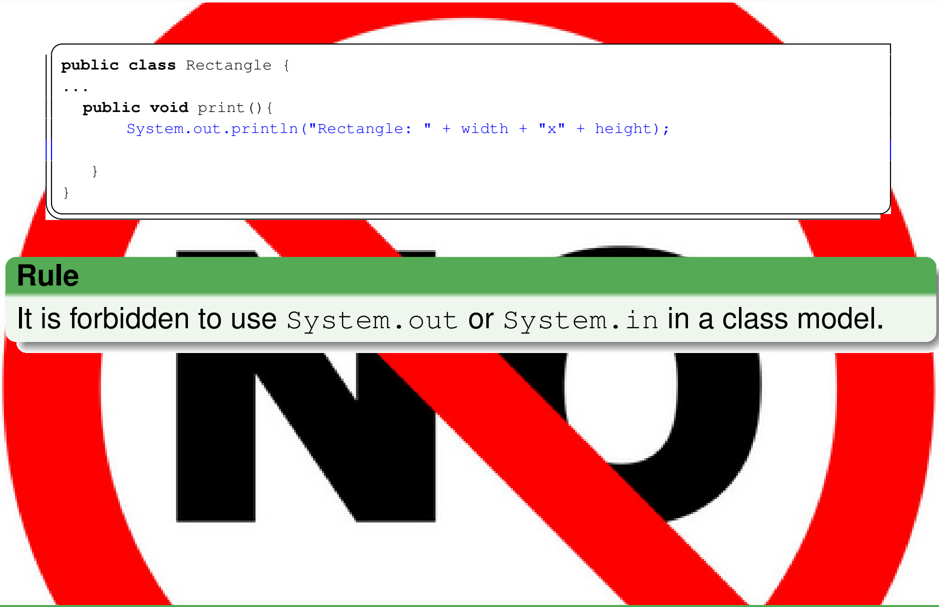
What is a class?

- `class` form the basic building blocks of any Java program.
    - Every program in Java consists of classes because the code for a program can appear only within a class definition.
- It defines a new type.
    - A class is the blueprint from which individual objects are created.
- `class` definition:
    - **Fields/instance variables**: These are variables that store data items that typically differentiate one object of the class from another. They are also referred to as data members of a class.
    - **Constructors**: They are a special type of method that is used to initialize the object. Java constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.
    - **Methods**: These define the operations you can perform for the class–so they determine what you can do to, or with, objects of the class. Methods typically operate on the fields or the data members of the class.

Class definition

```java
public class Rectangle {
   //Variables
   private int width;
   private int height;
   //Constructors
   public Rectangle(){
      width = 0;
      height = 0;
   }
   public Rectangle(int w, int h){
      width = w;
      height = h;
   }
   //methods
   public int getArea(){
      return width * height;
   }
   public int getPerimeter(){
      return 2 * ( width + height);
   }
   public void print(){
      System.out.println("Rectangle: " + width + "x" + height);
   }
}
```

Class definition: Attention



```
public class Rectangle {
...
  public void print(){
      System.out.println("Rectangle: " + width + "x" + height);

   }
}
```

Rule

It is forbidden to use `System.out` or `System.in` in a class model.

Class definition: Solution

```
public class Rectangle {
...
  @Override
   public String toString() {
      String str="Rectangle: " + width + "x" + height;
      return str;
   }
}
```

- Use `toString` method to return the object data in a formatted string.
    - `toString` method will be addressed later.

Class variables

- A variable is a storage location in a computer program. Each variable has a name and holds a value.
- An instance variable declaration consists of the following parts:
  - An access specifier (private)
  - The type of the instance variable (such as int)
  - The name of the instance variable (such as value)

```
public class Rectangle {
   //Variables
   private int width;
   private int height;
  ...
}
```

Class constructors

- A class contains constructors that are invoked to create objects from the class blueprint.
- Constructor declarations look like method declarations–except that they use the name of the class and have no return type.

```
public class Rectangle {
   ...
   //Constructors
   public Rectangle(){
      width = 0;
      height = 0;
   }
   public Rectangle(int w, int h){
      width = w;
      height = h;
   }
...
}
```

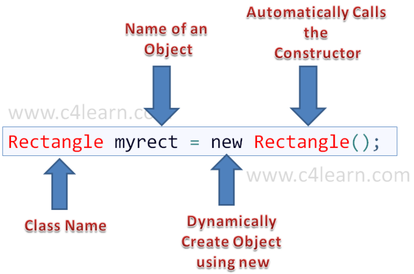
Class methods

- They require a minimum of three items:
  - **Modifier** : public, private, protected
  - **Return Type**: void, int, double, (etc.)
  - **Name**: whatever you want to call the method
  - **Parameters** (optional)

```java
public class Rectangle {
  ...
   //methods
   public int getArea(){
      return width * height;
   }
   public int getPerimeter(){
      return 2 * ( width + height);
   }
}
```

What is an object?

- An object is an instance of a class.

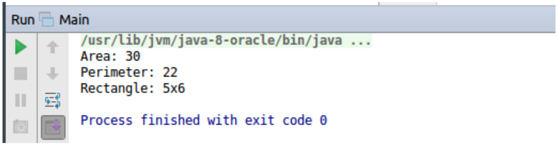

Name of an Object
Automatically Calls the Constructor
Class Name
Dynamically Create Object using new
www.c4learn.com

```
Rectangle myrect = new Rectangle();
```

Invoking methods (I)

- Using an object instance

```
public class Main {
  public static void main(String[] args) {
 // write your code here
      Rectangle rectangle = new Rectangle(5,6);
      int area = rectangle.getArea();
      System.out.println("Area: "+ area);
      int perimeter = rectangle.getPerimeter();
      System.out.println("Perimeter: "+ perimeter);
      String str = rectangle.toString();
      System.out.println(str);
   }
}
```



```
/usr/lib/jvm/java-8-oracle/bin/java ...
Area: 30
Perimeter: 22
Rectangle: 5x6

Process finished with exit code 0
```

Check: `TP1_01.zip`

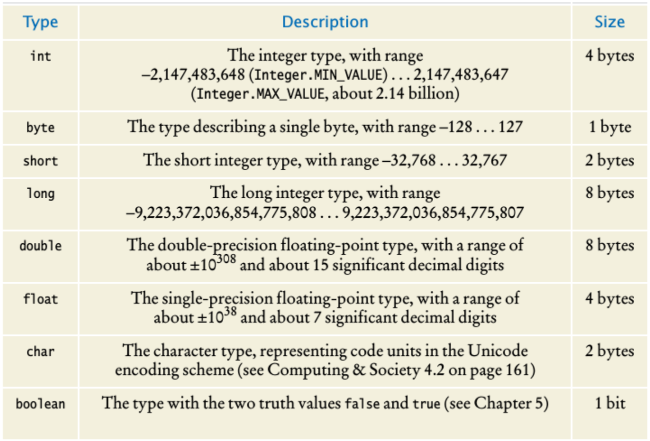
Invoking methods (II)

- Without an object instance
  - A `static` method can be invoked without an object instance of the class.

```
public class Main {

   static boolean isLeapYear(int year) { ...... }
   static boolean isValidDate(int year, int month, int day) { ...... }
   static int getDayOfWeek(int year, int month, int day) { ...... }
   public static void main(String[] args) {
     boolean leapYear = isLeapYear(1900);
     boolean validDate = isValidDate(2012, 2, 29);
     int dayOfWeek = getDayOfWeek(1982, 4, 24);
   }
}
```

Fundamental Data Types

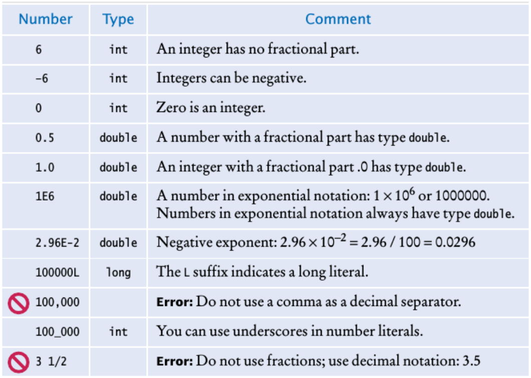
Primitive Types



Type	Description	Size
`int`	The integer type, with range –2,147,483,648 (`Integer.MIN_VALUE`) . . . 2,147,483,647 (`Integer.MAX_VALUE`, about 2.14 billion)	4 bytes
`byte`	The type describing a single byte, with range –128 . . . 127	1 byte
`short`	The short integer type, with range –32,768 . . . 32,767	2 bytes
`long`	The long integer type, with range –9,223,372,036,854,775,808 . . . 9,223,372,036,854,775,807	8 bytes
`double`	The double-precision floating-point type, with a range of about $\pm 10^{308}$ and about 15 significant decimal digits	8 bytes
`float`	The single-precision floating-point type, with a range of about $\pm 10^{38}$ and about 7 significant decimal digits	4 bytes
`char`	The character type, representing code units in the Unicode encoding scheme (see Computing & Society 4.2 on page 161)	2 bytes
`boolean`	The type with the two truth values `false` and `true` (see Chapter 5)	1 bit

Literal

- A **literal** is the source code representation of a fixed value.
  - Literals in Java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables.



Number	Type	Comment
`6`	`int`	An integer has no fractional part.
`-6`	`int`	Integers can be negative.
`0`	`int`	Zero is an integer.
`0.5`	`double`	A number with a fractional part has type `double`.
`1.0`	`double`	An integer with a fractional part .0 has type `double`.
`1E6`	`double`	A number in exponential notation: 1×10^6 or 1000000. Numbers in exponential notation always have type `double`.
`2.96E-2`	`double`	Negative exponent: $2.96 \times 10^{-2} = 2.96 / 100 = 0.0296$
`100000L`	`long`	The `L` suffix indicates a long literal.
🚫 `100,000`		**Error:** Do not use a comma as a decimal separator.
`100_000`	`int`	You can use underscores in number literals.
🚫 `3 1/2`		**Error:** Do not use fractions; use decimal notation: 3.5

Assignment

- =
  - It assigns the value on its right to the operand on its left

```
public static void main (String[] args) {
      int result = 1;
}
```

Aritmetic

- +
  - Additive operator (also used for String concatenation)
- -
  - Subtraction operator
- *
  - Multiplication operator
- /
  - Division operator
- %
  - Remainder operator

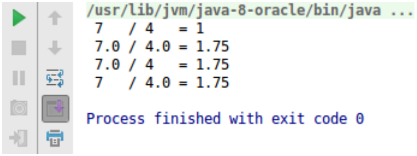
```
public static void main (String[] args) {
      int result = 1 + 2;
      // result is now 3
      System.out.println("1 + 2 = " + result);
}
```

Check: `TP1_02.zip`

Aritmetic: division

- If both arguments of a division (/) are integers, the remainder is discarded.
- If at least one is floating-point number the remainder is not discarded.

```
public static void main (String[] args) {
      System.out.println(" 7 / 4 = " + 7/4);
      System.out.println(" 7.0 / 4.0 = " + 7.0/4.0);
      System.out.println(" 7.0 / 4 = " + 7.0/4);
      System.out.println(" 7 / 4.0 = " + 7/4.0);
}
```



```
/usr/lib/jvm/java-8-oracle/bin/java ...
 7   / 4   = 1
 7.0 / 4.0 = 1.75
 7.0 / 4   = 1.75
 7   / 4.0 = 1.75

Process finished with exit code 0
```

Unary

- +
  - Unary plus operator; indicates positive value (numbers are positive without this, however)
- -
  - Unary minus operator; negates an expression
- ++
  - Increment operator; increments a value by 1
- - -
  - Decrement operator; decrements a value by 1
- !
  - Logical complement operator; inverts the value of a boolean

Check: `TP1_03.zip`

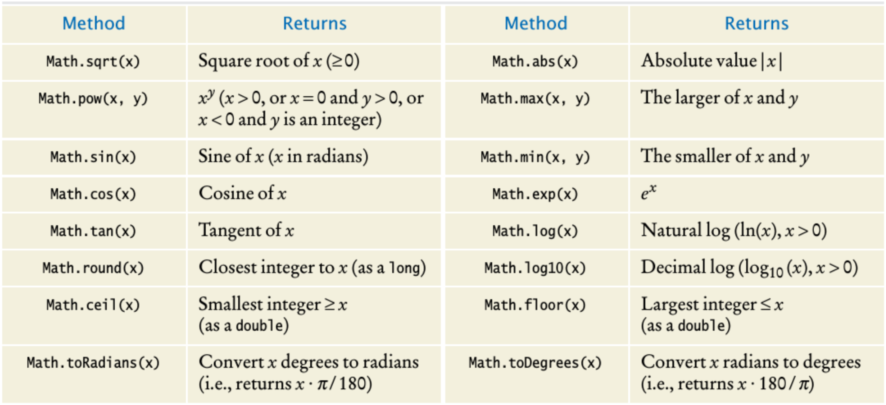
Unary: Increment/decrement operators

- The increment/decrement operators can be applied before (prefix) or after (postfix) the operand.
  - The code `result++;` and `++result;` will both end in result being incremented by one.
  - The only difference is that the prefix version (`++result;`) evaluates to the incremented value, whereas the postfix version (`result++;`) evaluates to the original value.

Check: `TP1_04.zip`

Powers, roots and others

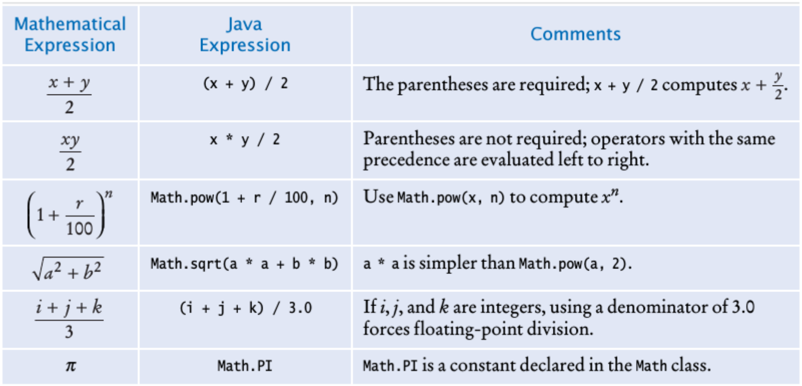
- In Java, there are no symbols for powers and roots. You have to use the Mathematical Java library.



Method	Returns	Method	Returns
`Math.sqrt(x)`	Square root of x (≥ 0)	`Math.abs(x)`	Absolute value $\lvert x \rvert$
`Math.pow(x, y)`	x^y ($x > 0$, or $x = 0$ and $y > 0$, or $x < 0$ and y is an integer)	`Math.max(x, y)`	The larger of x and y
`Math.sin(x)`	Sine of x (x in radians)	`Math.min(x, y)`	The smaller of x and y
`Math.cos(x)`	Cosine of x	`Math.exp(x)`	e^x
`Math.tan(x)`	Tangent of x	`Math.log(x)`	Natural log ($\ln(x)$, $x > 0$)
`Math.round(x)`	Closest integer to x (as a `long`)	`Math.log10(x)`	Decimal log ($\log_{10}(x)$, $x > 0$)
`Math.ceil(x)`	Smallest integer $\geq x$ (as a `double`)	`Math.floor(x)`	Largest integer $\leq x$ (as a `double`)
`Math.toRadians(x)`	Convert x degrees to radians (i.e., returns $x \cdot \pi / 180$)	`Math.toDegrees(x)`	Convert x radians to degrees (i.e., returns $x \cdot 180 / \pi$)

Check: `TP1_05.zip`

Arithmetic expressions



Mathematical Expression	Java Expression	Comments
$\frac{x+y}{2}$	`(x + y) / 2`	The parentheses are required; `x + y / 2` computes $x + \frac{y}{2}$.
$\frac{xy}{2}$	`x * y / 2`	Parentheses are not required; operators with the same precedence are evaluated left to right.
$\left(1 + \frac{r}{100}\right)^n$	`Math.pow(1 + r / 100, n)`	Use `Math.pow(x, n)` to compute x^n.
$\sqrt{a^2 + b^2}$	`Math.sqrt(a * a + b * b)`	`a * a` is simpler than `Math.pow(a, 2)`.
$\frac{i+j+k}{3}$	`(i + j + k) / 3.0`	If i, j, and k are integers, using a denominator of 3.0 forces floating-point division.
π	`Math.PI`	`Math.PI` is a constant declared in the `Math` class.

Cast

- You use a cast (typeName) to convert a value to a different type.
  - You must use the cast operator (int) to convert a convert floating-point value to an integer.
  - Write the cast operator before the expression that you want to convert:

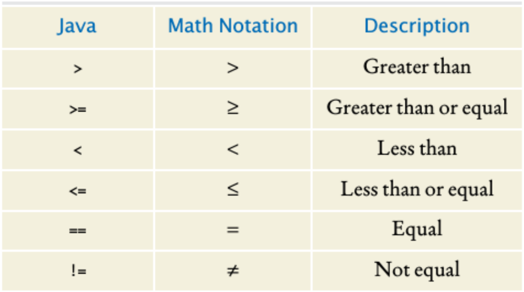
```
double balance = total + tax;
int dollars = (int) balance;
```

  - The cast `(int)` converts the floating-point value `balance` to an integer by discarding the fractional part. For example, if `balance` is 13.75, then `dollars` is set to 13.

Control Statements

Relational Operators

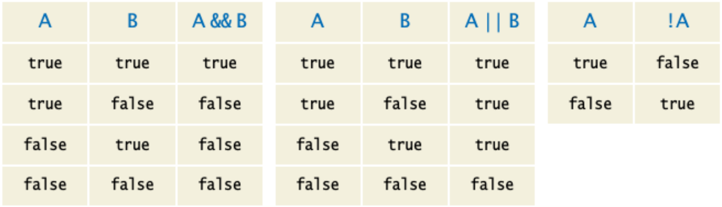
- A relational operator tests the relationship between two values.



Java	Math Notation	Description
`>`	$>$	Greater than
`>=`	$\geq$	Greater than or equal
`<`	$<$	Less than
`<=`	$\leq$	Less than or equal
`==`	$=$	Equal
`!=`	$\neq$	Not equal

Conditional Operators

- `&&` and `||` operators perform Logical AND and Logical OR operations on two boolean expressions.
- `!` operator perform Logical NOT operation of a boolean expression.

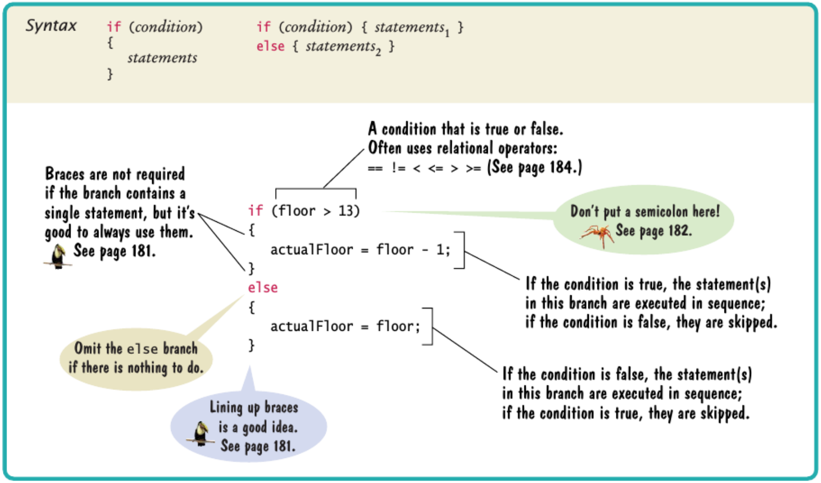


A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

A	!A
true	false
false	true

The `if` statement



Syntax

```
if (condition)
{
   statements
}
```

```
if (condition) { statements1 }
else { statements2 }
```

```
if (floor > 13)
{
   actualFloor = floor - 1;
}
else
{
   actualFloor = floor;
}
```

A condition that is true or false. Often uses relational operators: == != < <= > >= (See page 184.)

Braces are not required if the branch contains a single statement, but it's good to always use them. See page 181.

Don't put a semicolon here! See page 182.

If the condition is true, the statement(s) in this branch are executed in sequence; if the condition is false, they are skipped.

Omit the else branch if there is nothing to do.

Lining up braces is a good idea. See page 181.

If the condition is false, the statement(s) in this branch are executed in sequence; if the condition is true, they are skipped.

Check: `TP1_06.zip`

The ternary Operator

- Java has a conditional operator of the form:

```
condition ? value1 : value2
```

  - The value of that expression is either `value1` if the test passes or `value2` if it fails.
  - For example, we can compute the actual floor number as:

```
actualFloor = floor > 13 ? floor - 1 : floor;
```

which is equivalent to to.

```
if (floor > 13) {
   actualFloor = floor - 1;
} else {
   actualFloor = floor;
}
```

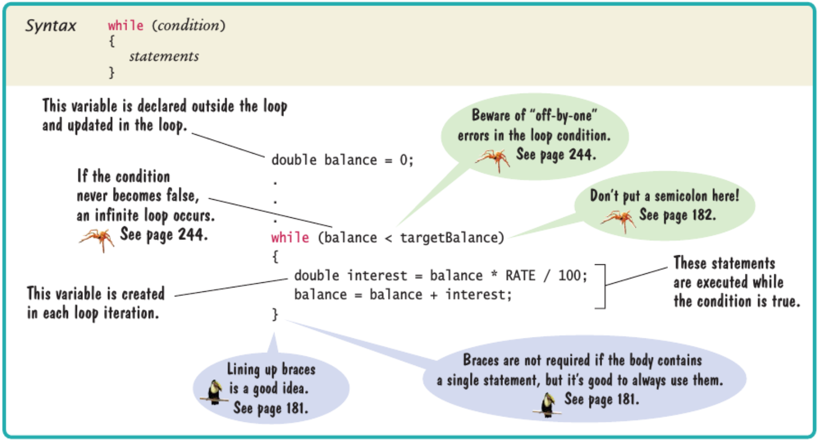
The `switch` statement

```
switch (expression) {
  case value1:
  // statement sequence
  break;
  case value2:
  // statement sequence
  break;
  ...
  case valueN:
  // statement sequence
  break;
  default:
  // default statement sequence
}
```

- The expression must be of type `byte`, `short`, `int`, or `char`; each of the values specified in the case statements must be of a type compatible with the expression.

Check: `TP1_07.zip`

The `while` loop



Syntax

```
while (condition)
{
   statements
}
```

```
double balance = 0;
.
.
.
while (balance < targetBalance)
{
   double interest = balance * RATE / 100;
   balance = balance + interest;
}
```

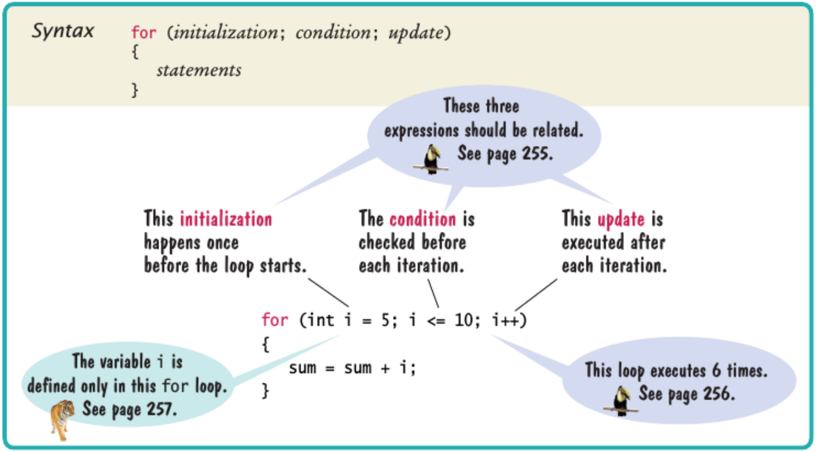
This variable is declared outside the loop and updated in the loop.
If the condition never becomes false, an infinite loop occurs. See page 244.
Beware of "off-by-one" errors in the loop condition. See page 244.
Don't put a semicolon here! See page 182.
These statements are executed while the condition is true.
This variable is created in each loop iteration.
Lining up braces is a good idea. See page 181.
Braces are not required if the body contains a single statement, but it's good to always use them. See page 181.

The `do - while` loop

- `do - while` loop is similar to `while` loop, however there is a single difference between these two.
- Unlike `while` loop, `do - while` guarantees at-least one execution of block of statements.
  - This happens because the `do - while` loop evaluates the boolean expression at the end of the loop?s body.
  - Therefore the set of statements gets executed at-least once before the check of boolean expression.

```
do {
    statement(s)
} while (expression);
```

The `for` loop



Syntax

```
for (initialization; condition; update)
{
   statements
}
```

```
for (int i = 5; i <= 10; i++)
{
   sum = sum + i;
}
```

These three expressions should be related. See page 255.
This initialization happens once before the loop starts.
The condition is checked before each iteration.
This update is executed after each iteration.
The variable i is defined only in this for loop. See page 257.
This loop executes 6 times. See page 256.

The `break` Statement

- The `break` statement is used in the `switch` statement..
- You can also use the `break` statement to terminate a `for`, `while`, or `do-while` loop.

```
for(int i=0; i<100; i++) {
   if(i == 10)
      break;
}
```

- A `break` statement terminates the innermost `switch`, `for`, `while`, or `do-while` statement

```
for(int i=0; i<3; i++) {
   for(int j=0; j<100; j++) {
      if(j == 10)
         break;
   }
}
```

Check: `TP1_08.zip`

The `continue` Statement

- Sometimes it is useful to force an early iteration of a loop. That is, you might want to continue running the loop but stop processing the remainder of the code in its body for this particular iteration.
  - This is, in effect, a `goto` just past the body of the loop, to the loop's end. The `continue` statement performs such an action.
  - In `while` and `do-while` loops, a `continue` statement causes control to be transferred directly to the conditional expression that controls the loop.
  - In a `for` loop, control goes first to the iteration portion of the for statement and then to the conditional expression. For all three loops, any intermediate code is bypassed.

```java
for(int i=0; i<10; i++) {
   System.out.print(i + " ");
   if (i%2 == 0)
      continue;
   System.out.println("");
}
```

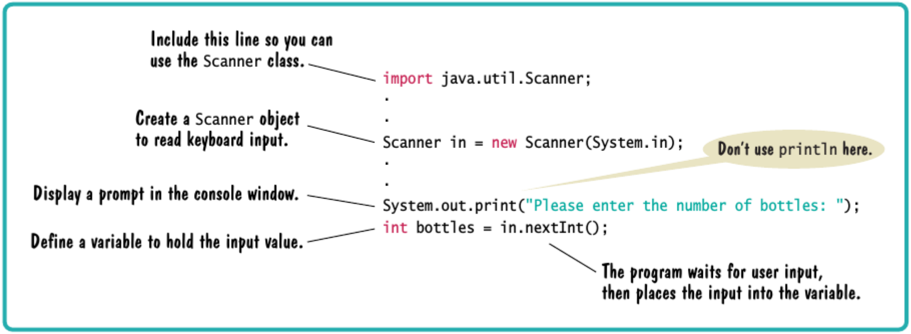
The `return` Statement

- The `return` statement is used to explicitly return from a method. That is, it causes program control to transfer back to the caller of the method.

```
public static void main(String args[]) {
   boolean t = true;
   System.out.println("Before the return.");
   if(t)
      return;
   System.out.println("This won’t execute.");
}
```

Input and Output

The Scanner



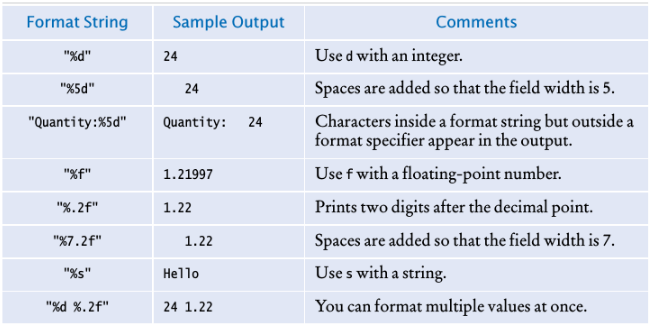
```
import java.util.Scanner;
.
.
Scanner in = new Scanner(System.in);
.
.
System.out.print("Please enter the number of bottles: ");
int bottles = in.nextInt();
```

Include this line so you can use the Scanner class.
Create a Scanner object to read keyboard input.
Don't use println here.
Display a prompt in the console window.
Define a variable to hold the input value.
The program waits for user input, then places the input into the variable.

Formatted output

- Use the `printf` method (of the `System.out` to specify how values should be formatted.

```
System.out.printf("Quantity: %d Total: %10.2f", quantity, total);
```



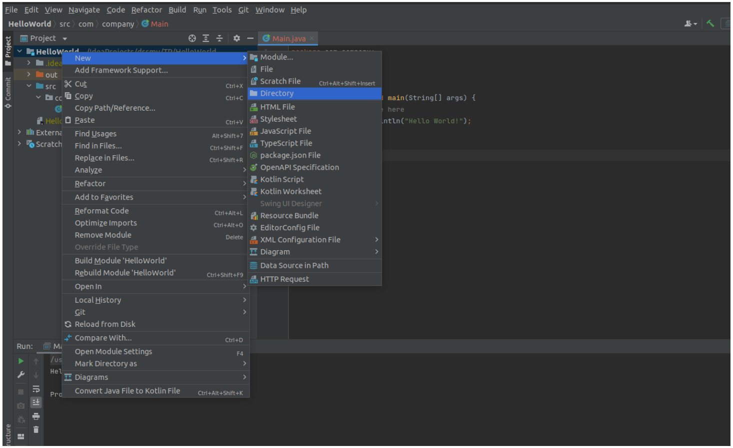
Format String	Sample Output	Comments
`"%d"`	`24`	Use `d` with an integer.
`"%5d"`	`   24`	Spaces are added so that the field width is 5.
`"Quantity:%5d"`	`Quantity:   24`	Characters inside a format string but outside a format specifier appear in the output.
`"%f"`	`1.21997`	Use `f` with a floating-point number.
`"%.2f"`	`1.22`	Prints two digits after the decimal point.
`"%7.2f"`	`   1.22`	Spaces are added so that the field width is 7.
`"%s"`	`Hello`	Use `s` with a string.
`"%d %.2f"`	`24 1.22`	You can format multiple values at once.

Check: `TP1_09.zip`

Unit Tests

Create a `test` folder (I)

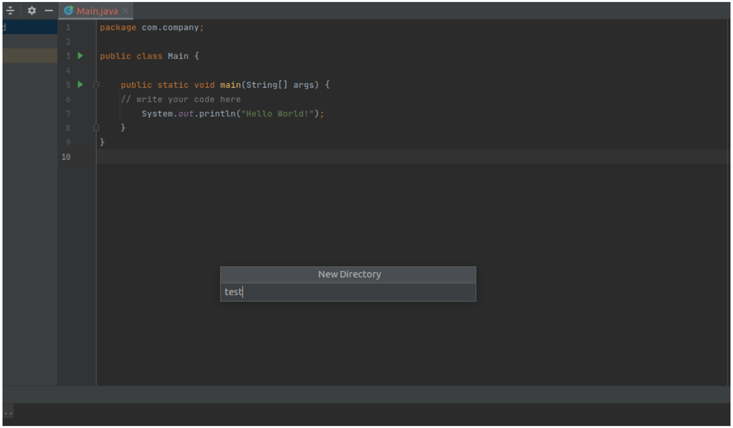
- Right click on your `HelloWorld` project and select `New > Directory`.



```
 main(String[] args) {
 here
ntln("Hello World!");
```

Create a `test` folder (II)

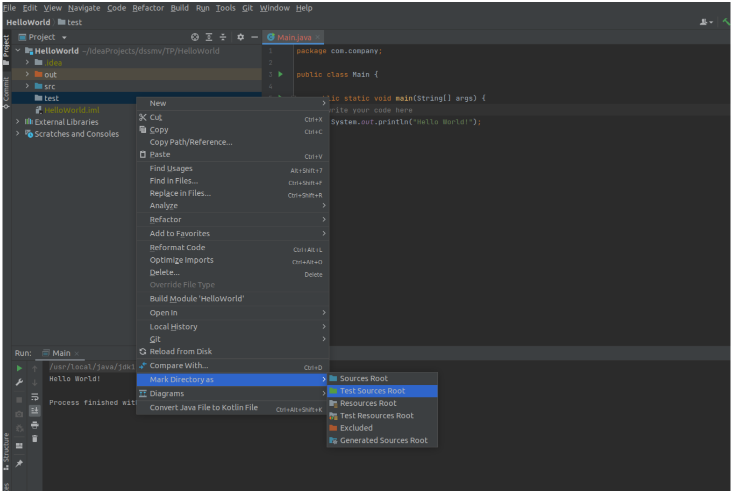
- Enter `test` name.



```
package com.company;

public class Main {

    public static void main(String[] args) {
    // write your code here
        System.out.println("Hello World!");
    }
}
```

Configure `test` folder as test root

- Right click on `test` folder and select `Mark Directory as > Test Sources Root`.

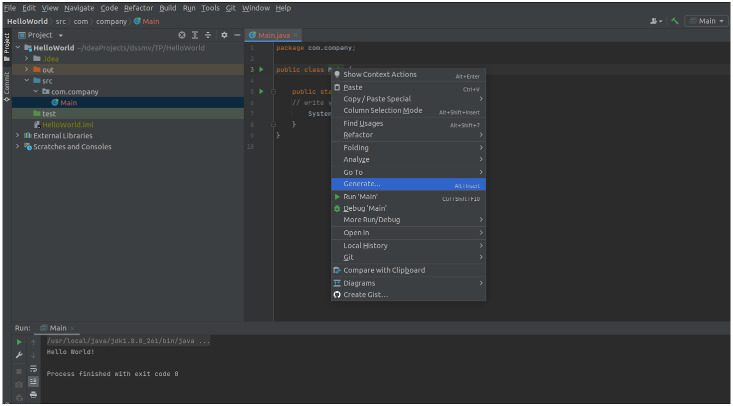


```
package com.company;

public class Main {

    ...lic static void main(String[] args) {
    ...rite your code here
        System.out.println("Hello World!");
```

```
/usr/local/java/jdk1
Hello World!

Process finished wit
```

Create the `MainTest` class (I)

- Right click on `Main` class and select `Generate`.

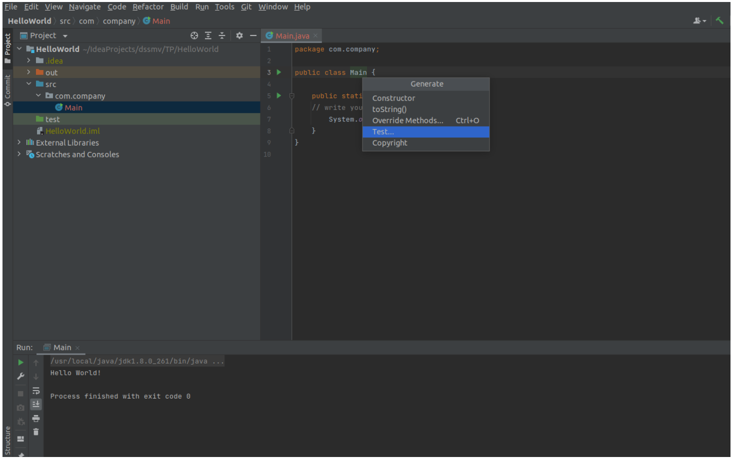


```
package com.company;

public class M

    public sta
    // write y
        System
    }
}
```

```
/usr/local/java/jdk1.8.0_261/bin/java ...
Hello World!

Process finished with exit code 0
```

Create the `MainTest` class (II)

- Select `Test...`.

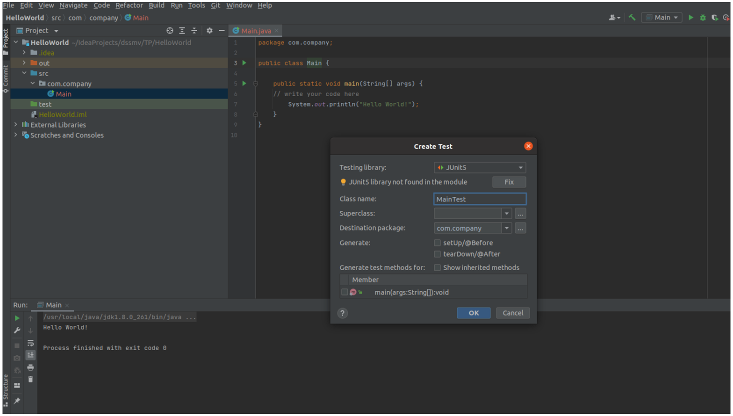

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
HelloWorld › src › com › company › Main
Project
HelloWorld ~/IdeaProjects/dssmv/TP/HelloWorld
.idea
out
src
com.company
Main
test
HelloWorld.iml
External Libraries
Scratches and Consoles
Main.java
Generate
Constructor
toString()
Override Methods... Ctrl+O
Test...
Copyright
Run: Main

```
package com.company;

public class Main {

    public stati
    // write you
        System.o
    }
}
```

```
/usr/local/java/jdk1.8.0_261/bin/java ...
Hello World!

Process finished with exit code 0
```

Create the `MainTest` class (III)

- Configure `MainTest` class.


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
HelloWworld 〉 src 〉 com 〉 company 〉 Main
Project
HelloWorld ~/IdeaProjects/dssmv/TP/HelloWorld
.idea
out
src
com.company
Main
test
HelloWorld.iml
External Libraries
Scratches and Consoles
Main.java

```
package com.company;

public class Main {

    public static void main(String[] args) {
    // write your code here
        System.out.println("Hello World!");
    }
}
```

Create Test

Testing library: JUnit5

JUnit5 library not found in the module — Fix

Class name: MainTest

Superclass:

Destination package: com.company

Generate: setUp/@Before, tearDown/@After

Generate test methods for: Show inherited methods

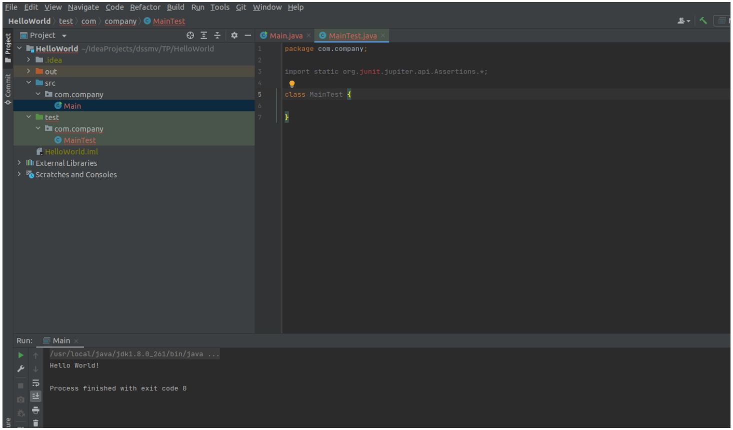
Member
main(args:String[]):void

OK Cancel

Run: Main

```
/usr/local/java/jdk1.8.0_261/bin/java ...
Hello World!

Process finished with exit code 0
```

Create the `MainTest` class (IV)


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
HelloWorld 〉 test 〉 com 〉 company 〉 MainTest
Project
HelloWorld ~/IdeaProjects/dssmv/TP/HelloWorld
.idea
out
src
com.company
Main
test
com.company
MainTest
HelloWorld.iml
External Libraries
Scratches and Consoles
Main.java
MainTest.java
Run: Main

```
package com.company;

import static org.junit.jupiter.api.Assertions.*;

class MainTest {

}
```

```
/usr/local/java/jdk1.8.0_261/bin/java ...
Hello World!

Process finished with exit code 0
```

Adding code (I)

- Add `isEven` method to class `Main`

```java
package com.company;

public class Main {

  public static boolean isEven(int n){
    boolean res = n % 2 == 0 ? true : false;
    return res;
  }

  public static void main(String[] args) {
    // write your code here
    System.out.println("Hello World!");
  }
}
```

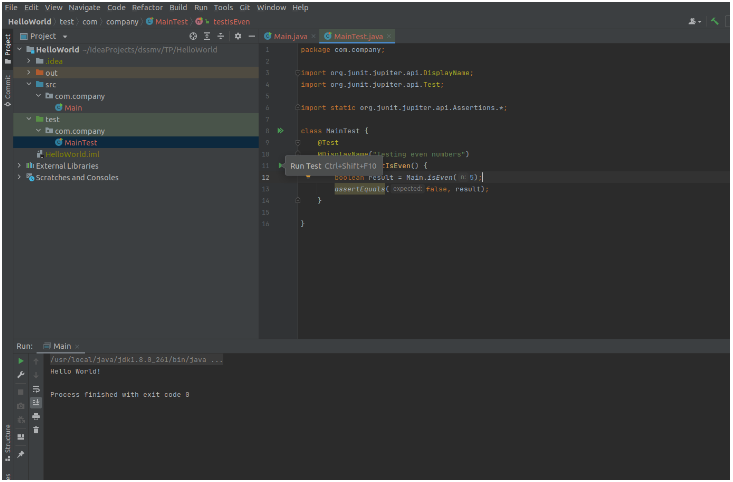
Adding code (II)

- Add isEven method test to class MainTest

```
package com.company;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class MainTest {
  @Test
  @DisplayName("Testing even numbers")
  public void testIsEven() {
    boolean result = Main.isEven(5);
    assertEquals(false, result);
  }
}
```

Running tests (I)

- Left click on green arrow.

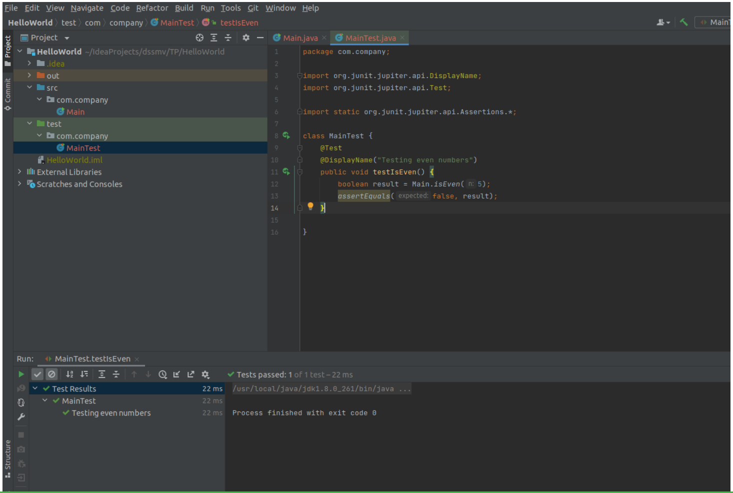

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
HelloWorld › test › com › company › MainTest › testIsEven
Project
HelloWorld ~/IdeaProjects/dssmv/TP/HelloWorld
.idea
out
src
com.company
Main
test
com.company
MainTest
HelloWorld.iml
External Libraries
Scratches and Consoles
Main.java
MainTest.java
Run Test Ctrl+Shift+F10
Run:
Main

```java
package com.company;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class MainTest {
    @Test
    @DisplayName("Testing even numbers")
    IsEven() {
        boolean result = Main.isEven( n: 5);
        assertEquals( expected: false, result);
    }

}
```

```
/usr/local/java/jdk1.8.0_261/bin/java ...
Hello World!

Process finished with exit code 0
```

Running tests (II)

- JUnit doc available at: `https://junit.org/junit5/docs/current/user-guide/`


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
HelloWorld test com company MainTest testIsEven
Project
HelloWorld ~/IdeaProjects/dssmv/TP/HelloWorld
.idea
out
src
com.company
Main
test
com.company
MainTest
HelloWorld.iml
External Libraries
Scratches and Consoles
Main.java
MainTest.java
Run: MainTest.testIsEven

```java
package com.company;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class MainTest {
    @Test
    @DisplayName("Testing even numbers")
    public void testIsEven() {
        boolean result = Main.isEven( n: 5);
        assertEquals( expected: false, result);
    }

}
```

```
Tests passed: 1 of 1 test – 22 ms

Test Results              22 ms
  MainTest                22 ms
    Testing even numbers  22 ms

/usr/local/java/jdk1.8.0_261/bin/java ...

Process finished with exit code 0
```

Using a Debugger

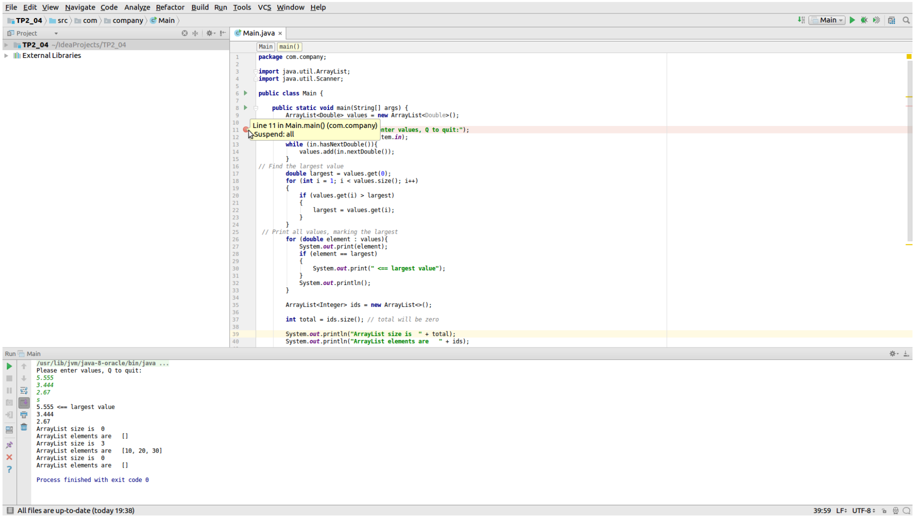
Debugging

- As you have undoubtedly realized by now, computer programs rarely run perfectly the first time.
- At times, it can be quite frustrating to find the bugs. Of course, you can insert print commands, run the program, and try to analyze the printout. If the printout does not clearly point to the problem, you may need to add and remove print commands and run the program again.
- Modern development environments contain special programs, called **debuggers**, that help you locate bugs by letting you follow the execution of a program.
  - You can stop and restart your program and see the contents of variables whenever your program is temporarily stopped.
  - At each stop, you have the choice of what variables to inspect and how many program steps to run until the next stop.

Check: `https://www.jetbrains.com/help/idea/2016.3/debugging.html`

Check: `https://www.youtube.com/watch?v=VdBsUv4lnm4`

Debugging with IntelliJ IDEA (I)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TP2_04 src com company Main
Project
TP2_04 ~/IdeaProjects/TP2_04
External Libraries
Main.java
Main main()
Run Main
39:59 LF: UTF-8

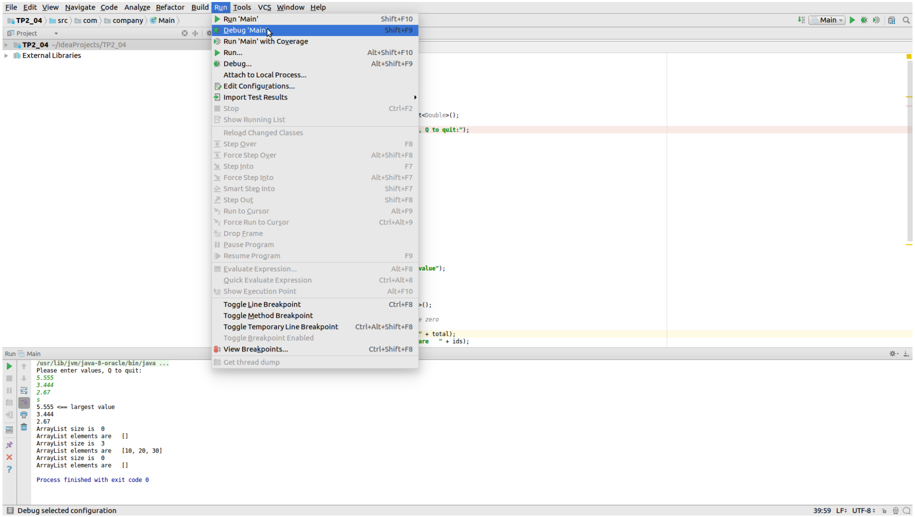
```java
package com.company;

import java.util.ArrayList;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        ArrayList<Double> values = new ArrayList<Double>();

                                    nter values, Q to quit:");
                                    tem.in);
        while (in.hasNextDouble()){
            values.add(in.nextDouble());
        }
// Find the largest value
        double largest = values.get(0);
        for (int i = 1; i < values.size(); i++)
        {
            if (values.get(i) > largest)
            {
                largest = values.get(i);
            }
        }
 // Print all values, marking the largest
        for (double element : values){
            System.out.print(element);
            if (element == largest)
            {
                System.out.print(" <== largest value");
            }
            System.out.println();
        }

        ArrayList<Integer> ids = new ArrayList<>();

        int total = ids.size(); // total will be zero

        System.out.println("ArrayList size is  " + total);
        System.out.println("ArrayList elements are   " + ids);
```

```
Line 11 in Main.main() (com.company)
Suspend: all
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Please enter values, Q to quit:
5.555
3.444
2.67
s
5.555 <== largest value
3.444
2.67
ArrayList size is  0
ArrayList elements are   []
ArrayList size is  3
ArrayList elements are   [10, 20, 30]
ArrayList size is  0
ArrayList elements are   []

Process finished with exit code 0
```

All files are up-to-date (today 19:38)

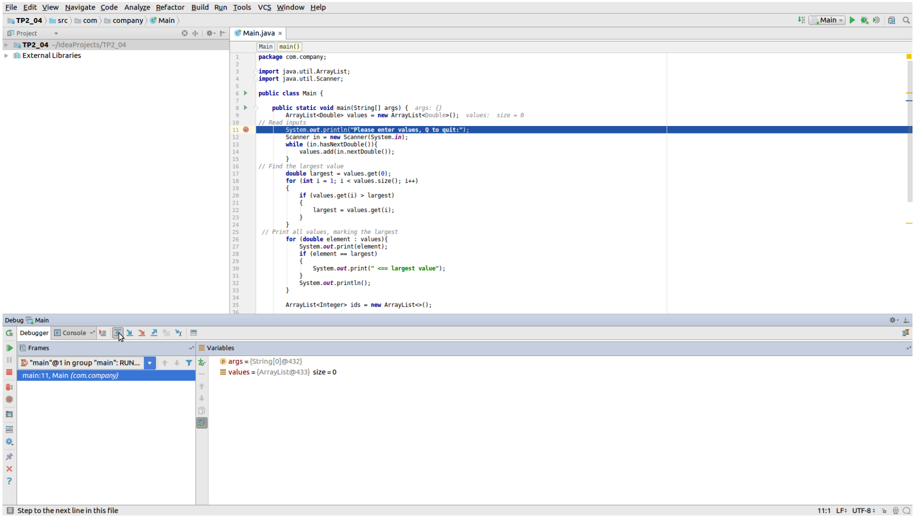
Debugging with IntelliJ IDEA (II)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TP2_04 src com company Main
Project
TP2_04 ~/IdeaProjects/TP2_04
External Libraries
Run 'Main' Shift+F10
Debug 'Main' Shift+F9
Run 'Main' with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Local Process...
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Reload Changed Classes
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Force Step Into Alt+Shift+F7
Smart Step Into Shift+F7
Step Out Shift+F8
Run to Cursor Alt+F9
Force Run to Cursor Ctrl+Alt+9
Drop Frame
Pause Program
Resume Program F9
Evaluate Expression... Alt+F8
Quick Evaluate Expression Ctrl+Alt+8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Method Breakpoint
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
Get thread dump
Main
Run Main
Debug selected configuration
39:59 LF: UTF-8:

```
t<Double>();
, Q to quit:");
value");
>();
e zero
" + total);
are   " + ids);
```

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Please enter values, Q to quit:
5.555
3.444
2.67
s
5.555 <== largest value
3.444
2.67
ArrayList size is  0
ArrayList elements are   []
ArrayList size is  3
ArrayList elements are   [10, 20, 30]
ArrayList size is  0
ArrayList elements are   []

Process finished with exit code 0
```

Debugging with IntelliJ IDEA (III)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TP2_04 src com company Main
Main
Project
TP2_04 ~/IdeaProjects/TP2_04
External Libraries
Main.java
Main main()
Debug Main
Debugger Console
Frames
Variables

```
package com.company;

import java.util.ArrayList;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {  args: {}
        ArrayList<Double> values = new ArrayList<Double>();  values:  size = 0
// Read inputs
        System.out.println("Please enter values, Q to quit:");
        Scanner in = new Scanner(System.in);
        while (in.hasNextDouble()){
            values.add(in.nextDouble());
        }
// Find the largest value
        double largest = values.get(0);
        for (int i = 1; i < values.size(); i++)
        {
            if (values.get(i) > largest)
            {
                largest = values.get(i);
            }
        }
 // Print all values, marking the largest
        for (double element : values){
            System.out.print(element);
            if (element == largest)
            {
                System.out.print(" <== largest value");
            }
            System.out.println();
        }

        ArrayList<Integer> ids = new ArrayList<>();
```

```
"main"@1 in group "main": RUN...
main:11, Main (com.company)
```

```
args = {String[0]@432}
values = {ArrayList@433} size = 0
```

Step to the next line in this file

11:1 LF: UTF-8:

Bibliography

Resources

- "Big Java: Early Objects", 6th Edition by Cay S. Horstmann
- "Java™:The Complete Reference", 7th Edition,Herbert Schildt
- "Java™Programming", 7th Edition, Joyce Farrell
- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html
- http://beginnersbook.com/java-tutorial-for-beginners-with-examples/
- https://www.leepoint.net/index.html
- https://junit.org/junit5/docs/current/user-guide/