Java Chapter 2: Getting Started with Java

Contents

Time required: 90 minutes

How Popular is Java?

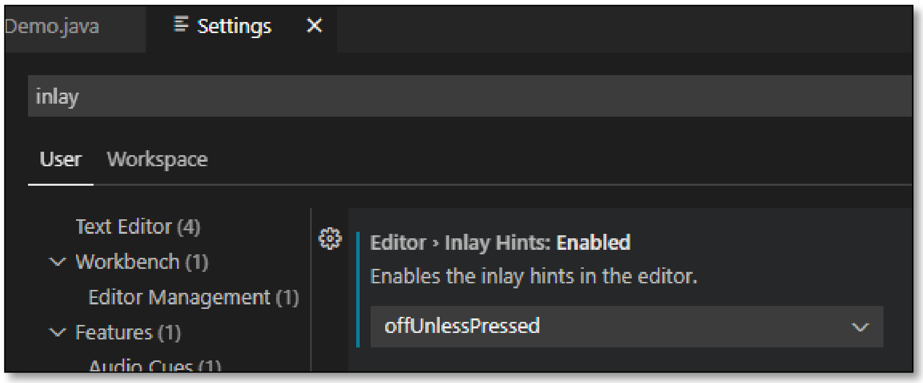
https://pypl.github.io/PYPL.html

https://www.tiobe.com/tiobe-index/

Visual Studio Code Inlay Hints

You might notice some extra characters in your Java code you did not type in. This is called Inlay Hints. You may not want those hints at this time.

1. In Visual Studio Code → **File** → **Preferences** → **Settings**.
2. Type **inlay** in the search bar.
3. Change the setting to **offUnlessPressed** as shown.



Editor › Inlay Hints: Enabled
Enables the inlay hints in the editor.
offUnlessPressed

Do: Think Java, 2nd Ed. (Interactive Edition)

Think Java, Second Edition
by Allen Downey and Chris Mayfield

If you wish to read offline, the entire book is available in pdf: Think Java

Read the following interactive chapters. These have built in executable code snippets.

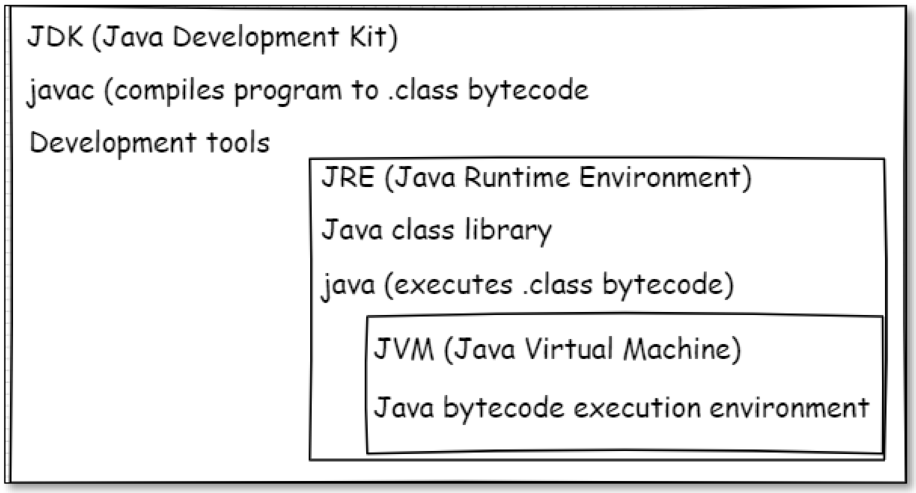
- Chapter 1: Computer Programming
- Chapter 2: Variables and Operators

Do: Java Fundamentals Tutorials

- LearnJavaOnline.org Hello World!
- LearnJavaOnline.org Variables and Types
- Java Tutorial
- Java Intro
- Java Get Started (Skip to Java Quickstart)
- Java Syntax
- Java Output
- Java Comments

- Java Variables
- Java Data Types
- Java User Input

How Does Java Work?


JDK (Java Development Kit)
javac (compiles program to .class bytecode
Development tools
JRE (Java Runtime Environment)
Java class library
java (executes .class bytecode)
JVM (Java Virtual Machine)
Java bytecode execution environment

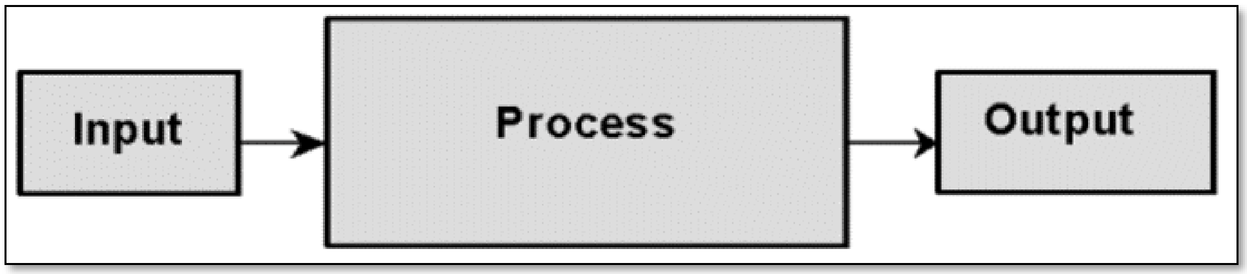
JDK (Java Development Kit) used for developing Java program. Converts program into executable bytecode (*.class)

JRE (Java Runtime Environment) runs the Java program.

JVM (Java Virtual Machine) is used by the JRE to execute the bytecode.

Input - Process - Output

All programs have three basic components.



Java Data Types and Sizes

Primitive Data Types

A primitive data type stores the value directly in the memory address.

There are eight primitive data types in Java.

Data Type	Size	Description
byte	1 byte	Whole numbers from -128 to 127
short	2 bytes	Whole numbers from -32,768 to 32,767
int	4 bytes	Whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Fractional numbers. 6 to 7 decimal digits
double	8 bytes	Fractional numbers. 15 decimal digits
BigDecimal	32 bytes	Handles very large and very small floating point numbers. It is a class, not a primitive data type.
boolean	1 bit	True or false values
char	2 bytes	Single character/letter or ASCII values

Doubles, not Floats

Even though the float data type is listed, and has fairly good precision, it is recommended to always use doubles.

It's actually faster to process on many modern computers. That's because computers have, at the chip level, the functionality to actually deal with these double numbers faster than the equivalent float.

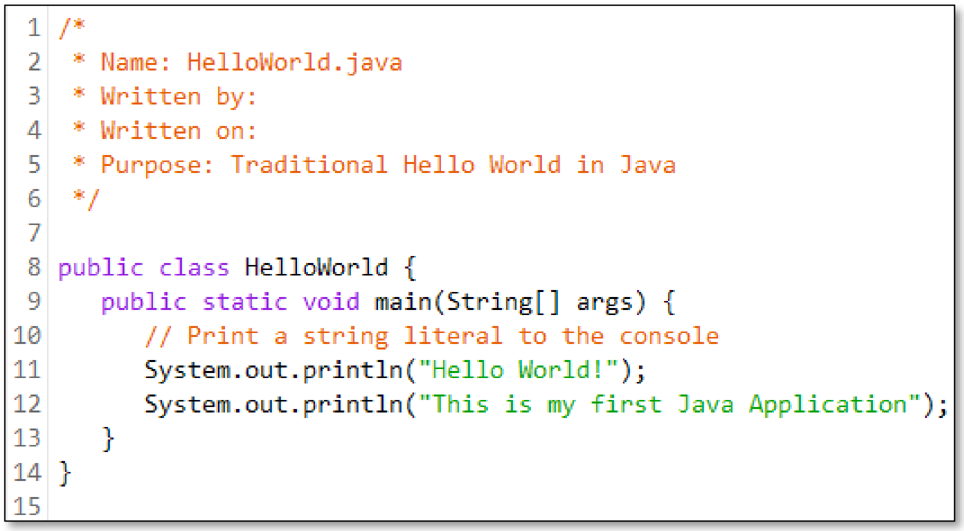
The Java libraries that we'll get into later in the course, particularly math functions, are often written to process doubles and not floats, and to return the result as a double.

The creators of Java selected the double because it's more precise, and it can handle a larger range of numbers.

Tutorial 1: Hello World

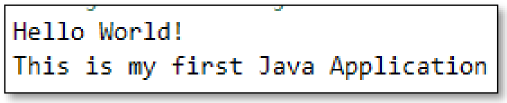
Hello World is the traditional first program you write in any programming language. Doing this confirms that your development environment is working.

Create a Java program named **HelloWorld.java**



```java
/*
 * Name: HelloWorld.java
 * Written by:
 * Written on:
 * Purpose: Traditional Hello World in Java
 */

public class HelloWorld {
   public static void main(String[] args) {
      // Print a string literal to the console
      System.out.println("Hello World!");
      System.out.println("This is my first Java Application");
   }
}

```

Example run:



```
Hello World!
This is my first Java Application
```

How Does This Work?

```
// Traditional Hello World program in Java
// Single line comment

*/
Multiline comment
*/
```

Any line in a Java program starting with // is a comment. Comments are intended for others reading the code to understand the intent and functionality of the program. Comments are also for the programmer of the code to remember what they were doing with the code. Come back to a project 6 months or a year later and see how much you remember.

Comments are completely ignored by the Java compiler (the application that translates Java program to Java bytecode that computer can execute).

```
public class HelloWorld {
   ...
}
```

In Java, every application begins with a class definition. In the program, HelloWorld is the name of the class, and the class definition is between the curly braces {}

Every Java application has a class definition. The name of the class must match the filename in Java.

```
public static void main(String[] args) {
  ...
}
```

This is the main method is the entry point of your Java application. Every application in Java must contain the main method. The Java compiler starts executing the code from the main method.

```
System.out.println("Hello, World!");
```

The code above is a print statement. It prints the text Hello, World! to standard output (your screen). The text inside the quotation marks is called a String literal.

Notice the print statement is inside the main function, which is inside the class definition.

Tutorial 2: Hello Dialog

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes can be used to display information or get input from the user.

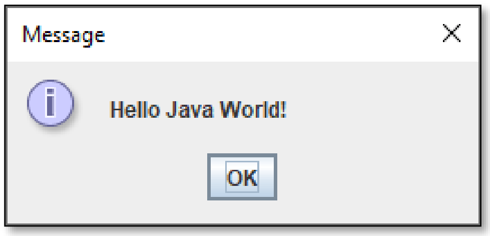
A library is a collection of prewritten code to add capabilities to a program. **JOptionPane** is not included in the Java standard library. This is the syntax for importing a library in Java.

```
// Import JOptionPane library for Java GUI dialog boxes
import javax.swing.JOptionPane;
```

Create a Java program named **HelloDialog.java**

```java
/*
 * Name: HelloDialog.java
 * Written by:
 * Written on:
 * Purpose: Traditional Hello World in Java JOptionPane
 */

// Import JOptionPane library for Java GUI dialog boxes
import javax.swing.JOptionPane;

public class HelloDialog {
   public static void main(String[] args) {
      // Show a message to the user with dialog box
      JOptionPane.showMessageDialog(null, "Hello Java World!");
   }
}
```

Example run:


Message

Hello Java World!

Tutorial 3: Miles to Kilometers

We are going to work with constants, variables, and expressions.

- **Constants:** store values that do not change
- **Variables:** store values that can change
- **Expression:** performs the work of the program, a + b

Constant: A constant is a variable whose value cannot change once it has been assigned. It is something we know before the program starts. Constants use the keyword "**final**".

```java
// Constant for converting Miles to KM
final double KM_PER_MILE = 1.609;
```

Variables: A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type. A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed. It is "able" to "vary".

```java
// Variables for input and calculations
double miles = 100.0;
double kilometers;
```

Expressions: Expressions perform the work of a program. Expressions are used to compute and assign values to variables and to help control the execution flow of a program. The job of an expression is two-fold: perform the computation indicated by the elements of the expression and return some value.

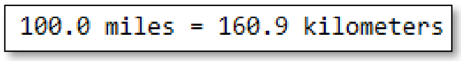
```java
// Convert miles to kilometers
kilometers = miles * KM_PER_MILE;
```

This program is hard coded to convert 100 miles to kilometers.

Create a Java program named: **Miles100ToKM.java**

```java
/*
 * Name: Miles100ToKM.java
 * Written by:
 * Written on:
 * Purpose: Convert 100 miles to Kilometers
 */

public class Miles100ToKM {

    public static void main(String[] args) {
        // Constant for converting Miles to KM
        final double KM_PER_MILE = 1.609;

        // Variables for input and calculations
        double miles = 100.0;
        double kilometers;

        // Convert miles to kilometers
        kilometers = miles * KM_PER_MILE;

        // Display answer to user
        System.out.println(miles + " miles = " + kilometers + " kilometers");
    }
}
```

Example run:



```
100.0 miles = 160.9 kilometers
```

Tutorial 3: Get Input

Scanner: The Scanner library is used to get input from the user during runtime. The input is to be of primitive data types, like int, float, double, string, etc.

In order to use the object of Scanner, we need to import **java.util.Scanner** package.

```
import java.util.Scanner
```

We create an object of the `Scanner` class. We can use the object to take input from the user.

```
// Create an object of Scanner
Scanner keyboard = new Scanner(System.in);

// Get integer input from the user
int number = keyboard.nextInt();
```

Create a Java program named **GetInput.java**

```java
/*****************************************
 * Name: GetInput.java
 * Written by:
 * Written on:
 * Purpose: Demonstrate methods of input
 *****************************************/

// Import Scanner library for input
import java.util.Scanner;

class GetInput {
    public static void main(String[] args) {
        // Create scanner object
        Scanner keyboard = new Scanner(System.in);

        // Get int input from user
        System.out.print("Enter int: ");
        int myInt = keyboard.nextInt();
        System.out.println("Int entered = " + myInt);

        // Get double input from user
        System.out.print("Enter double: ");
        double myDouble = keyboard.nextDouble();
        System.out.println("Double entered = " + myDouble);

        // Get String input from user
        System.out.print("Enter text: ");
        String myString = keyboard.next();
        System.out.println("Text entered = " + myString);

        // Close Scanner resources
        keyboard.close();
    }
}
```

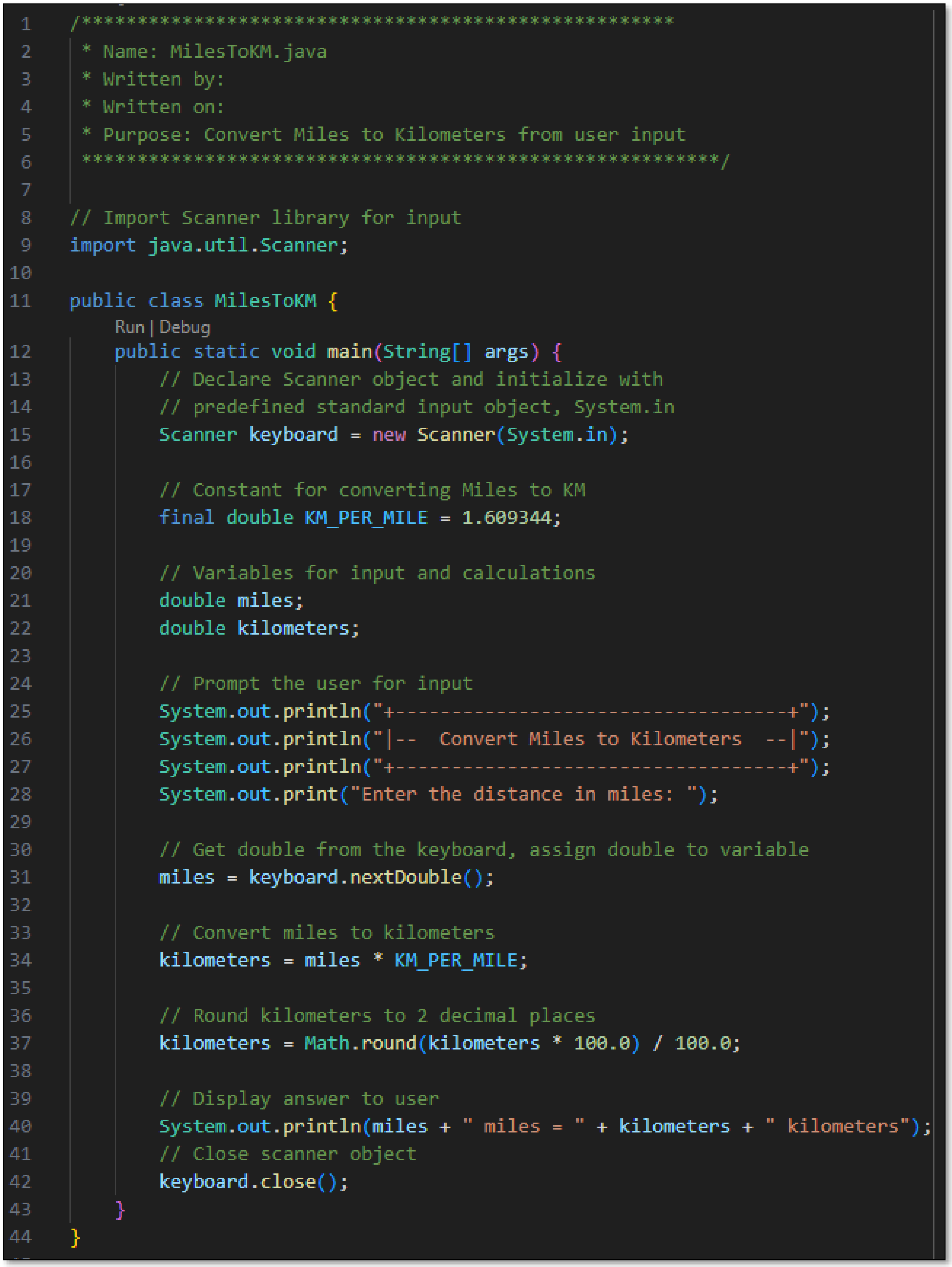
Example run:

```
Enter int: 256
Int entered = 256
Enter double: 3.145
Double entered = 3.145
Enter text: Hello!
Text entered = Hello!
```

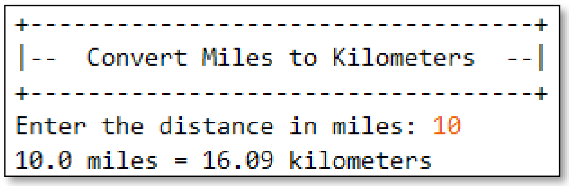
Tutorial 4: Miles to Kilometers with User Input

Let’s convert miles to kilometers based on user input.

Create a Java program named: **MilesToKM.java**


```java
/********************************************************
 * Name: MilesToKM.java
 * Written by:
 * Written on:
 * Purpose: Convert Miles to Kilometers from user input
 ********************************************************/

// Import Scanner library for input
import java.util.Scanner;

public class MilesToKM {
    Run | Debug
    public static void main(String[] args) {
        // Declare Scanner object and initialize with
        // predefined standard input object, System.in
        Scanner keyboard = new Scanner(System.in);

        // Constant for converting Miles to KM
        final double KM_PER_MILE = 1.609344;

        // Variables for input and calculations
        double miles;
        double kilometers;

        // Prompt the user for input
        System.out.println("+-----------------------------------+");
        System.out.println("|--  Convert Miles to Kilometers  --|");
        System.out.println("+-----------------------------------+");
        System.out.print("Enter the distance in miles: ");

        // Get double from the keyboard, assign double to variable
        miles = keyboard.nextDouble();

        // Convert miles to kilometers
        kilometers = miles * KM_PER_MILE;

        // Round kilometers to 2 decimal places
        kilometers = Math.round(kilometers * 100.0) / 100.0;

        // Display answer to user
        System.out.println(miles + " miles = " + kilometers + " kilometers");
        // Close scanner object
        keyboard.close();
    }
}
```

Example run:



```
+----------------------------------+
|--  Convert Miles to Kilometers  --|
+----------------------------------+
Enter the distance in miles: 10
10.0 miles = 16.09 kilometers
```

Assignment 1: Bake Your Own Cookies

- Ask a user for 2 numbers.
- Multiply them together.
- Display the answer.

Too easy? Get creative!

Assignment Submission

1. Use pseudocode or TODO.
2. Comment your code to show understanding.
3. Attach the program files.
4. Attach screenshots showing the successful operation of the program.
5. Submit in Blackboard.