Java Intro

Chapter 1

Somaiyah Sultani, 12-01-2022

Introduction

Compiling and Running a Java Program

Compilation

Why to learn the underlying process

- To better understand why we have to follow certain rules and regulations within Java Language
- We would make less mistakes while coding because we know the reason why things to be in a certain way

Source code

- We need an editor
  - Could use any simple text editor, such as notepad in windows
  - Linux editors, vim (recommend for the beginners)
  - IDE, integrated development environment
- Write source code file
- Save the file with .java extension

Source code

- Java is case-sentive meaning Hello is different from hello
- Free-form layout: the white spaces in the code is ignored by compiler.
  - Could write the entire source code into one line; without any line break
  - Could add as much white space as we would like
  - How to use this characteristic?
    - Add white spaces to keep the related lines of the code together and keep them apart from the rest of the source code; it creates a more readable code; good coding style.

Compile

- Compiler jobs
  - Parse the source code
    - Check the language syntax (vocabulary and grammar of the language)
    - Check for any syntax error
  - Allocate memory for the data that will be needed through out the execution of the program.
    - Hence the declaration of the variables and declaring their data type
    - dataType variableName;
    - dataType variableName = initialValue;
    - Compiler needs to know what is the type of the variables to know exactly how much memory they require.
  - Creates a file with the .class extension where it will save the compiled and translated code there

Compilation process

How to compile

- We need to use javac command to compile a file
  - Username~ javac [options] fileName.java
- Once compiled: create a fileName.class file

Virtual Machine

- For each fileName.java, there will be a corresponding fileName.class file
- All these .class files will be sent to the Java Virtual Machine (JVM) plus any library files
- The program will be executed at this stage and it will start running
-

Virtual Machine

How to run the program

- Use the java command
  - Username~ java className
  - Since the name of each Java file must be exactly the same as the class that it contains then the class name can be used to run the program, it is provided by the compiler
  - Hence the reason the name of the file must match the name of the class EXACTLY!

From Source Code to Running Program

Overview of the process

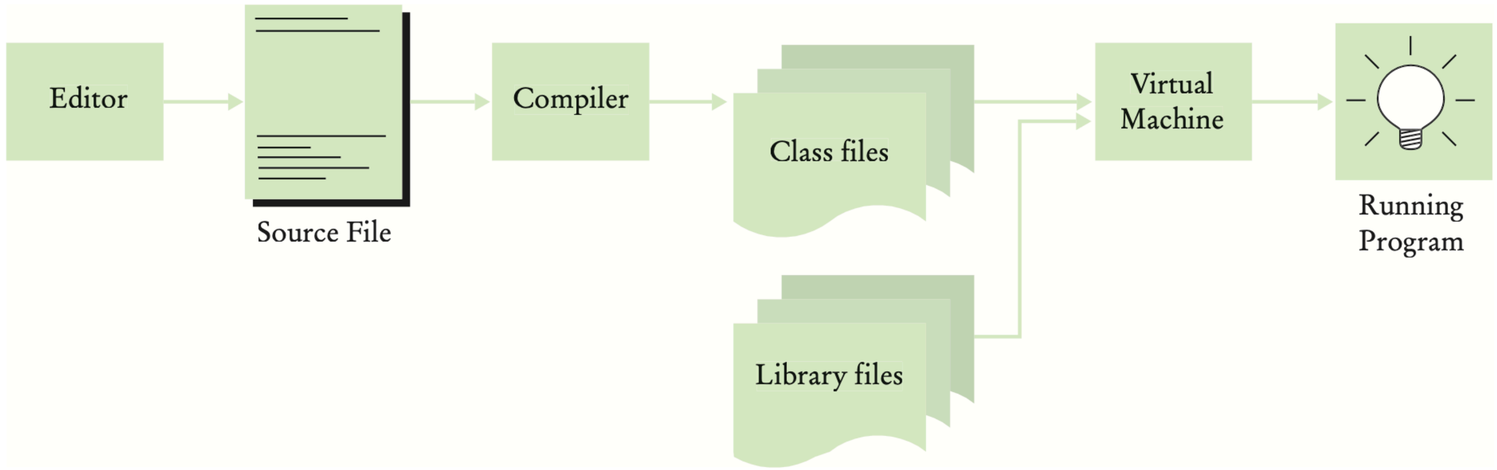

Figure 10 From Source Code to Running Program

User Input/Output

Input

- To read the user's input we need to use Scanner class
  - Just as the name suggests, it scans the user's input
  - It can read it into various datatypes
- Where it is and how to use it
  - It is in the java.util package
  - To use this class we need to import it into the source code file
    - Use the import keyword
      - At the very top of the file and outside of the definition of the class
      - import java.util.Scanner;

- Since Scanner is a class to use it we need to create an **instance/Object of this class**
  - ClassName variableName = new ClassName(some parameters);
  - ClassName is the dataType of the object/variable
  - ClassName(...) is the constructor
  - new keyword, which will be ALWAYS before the constructor.
- To read the user input from the command line, we create the following Scanner object
  - Scanner scan = new Scanner(system.in);

Scanner

- Once the object is created, we will use the various ***next*** methods provided by the scanner class to read the input from the scanner object.
- To call a method on an object, we use the ***dot operator***
  - objectName.methodName(method'sParameters)
- To read data from the Scanner object
  - String line = scan.nextLine(); => read the entire line as a String and assign it to the variable *line*