AI224: Operating Systems — Lab 2

Dr. Karim Lounis

Jan - Jun 2025

Java (Platform) & Java Programming Language[1]

Named after the Java island in Indonesia, known for its tasty coffee.

[1]Wanna learn Java, check out this link: https://www.w3schools.com/java/default.asp

Java Platform (Java)

Java Platform. Is a set of computer software used for developing cross-platform application software. Java's main components are:

- **Java Virtual Machine (JVM).** Provides runtime environment in which Java bytecode can be executed (loads, verifies, & executes).

  JVM is platform dependent. It runs in the background.

- **Java Runtime Environment (JRE).** Is the component that creates the JVM and its dependencies (JVM + runtime libraries).

  JRE is platform dependent. Minimum requirement to run Java code

- **Java Development Kit (JDK).** Groups the JRE and some development tools (e.g., Compiler, Debugger, JavaDoc, ...).

  JDK is platform dependent. Minimum requirement to write, debug, and run Java code (Clojure, Groovy, Scala, JRuby, and Jython)

Used in a variety of platforms (embedded, mobile, supercomputer, etc).

Java Programming Language ($\sim$ Programs for all OSs)

Java Programming Language. Is an object-oriented, class-based, high-level, WORA, portable, and secure programming language.

- Created in 1995 by James Gosling at Sun Microsystems.
- Initially released as a proprietary, then under GPL-2.0 license (2007).
- Now owned and maintained by Oracle corporation since 2010.
- Runs on everything, from laptops to data centers, game consoles to scientific supercomputers (you just need a JVM for that platform).
- Top 4 programming languages according to TIOBE Index (2023).
- It is simple, platform-independent, OOP, reliable, multi-threaded, applets-capable, secure, and preprocessor-free (i.e., no header files required).
- Used to develop Java applications (Desktop-apps, web-apps, business-apps, distributed-app, mobile-apps, games, etc) and applets.
- Has a very rich API (packages) compared to other languages.

How Does Java Work?

There are three types of programming languages:

- **[Faster] Compiler-based Languages.** The source code is compiled once to produce the executable (ready for execution).

  C, C++, C, Erlang, Haskell, Rust, Go, ...

- **[Slower] Interpreter-based Languages.** The source code is read by an interpreter, which translates it into machine code to run.

  PHP, Ruby, Python, bash, HTML, JavaScript, ...

- **[Both] Hybrid Languages.** The source code is compiled into a platform-independent bytecode. The bytecode is then translated into machine code to run on a specific platform. E.g., Java, Scala, ...

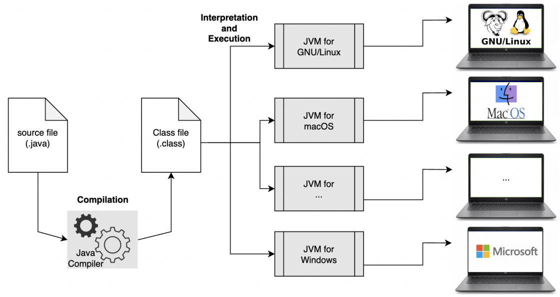
Programs written in compiled-based languages execute and directly interact with the OS. Also, the interpreter-based languages are executed and interact with the OS through their interpreter. Java programs, however, interact with OS thru JVM.

☞ Another classification: General purpose Vs Special purpose PLs.

How Does Java Work?

A Java program (.java) is compiled to a Java bytecode (.class) — machine-independent representation (error-free), using a Java compiler (`javac`).

The bytecode is then executed by an interpreter, the Java Virtual Machine.



The bytecode can be executed on any computer architecture that has the proper JVM for that architecture. The price to pay is a slower execution.

How Does Java Work?

☞ To use Java (write, compile, and execute), you need to download and install the suitable JDK for your operating system (JRE if execution only).

- Go to `https://www.java.com/fr/download/manual.jsp`, or
- On GNU/Linux, run `$sudo apt install default-jdk`

☞ Then, you can check the installation using `$java -version`.

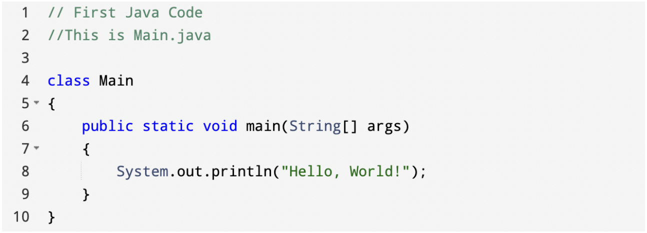
☞ Set the Path: So that the system knows where to find the compiler (`javac`) and the JVM (`java`). Under GNU/Linux, it is automatically set.

☞ Use a text editor (e.g., `nano, gedit, notepad, etc`) and start writing your Java source code.

You may use IDEs (NetBeans, Eclipse, JEdit, etc) to write, compile, and execute Java code (All in one). It is easier and faster, but not legacy.

How Does Java Work ("Hello, World!")?

Open a text editor, and write the following Java code:



```java
// First Java Code
//This is Main.java

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

```
$javac Main.java        (compile to bytecode)
$ls                   (listing current directory)
    Main.class        (created bytecode)
    Main.java        (original Java source file)
$java Main            (interpret and execute)
    Hello, Wolrd!          (our output)
```

Overview: Java Programming Language

Java Fundamental Data Types

Java has 10 fundamental types (called primitives):

I. Numerical Types:

- Integrals: **byte** (8 bits), **short** (16 bits), **int** (32 bits), & **long** (64 bits).
- Floating points: **float** (32 bits) and **double** (64 bits).

II. Non-numerical Types:

- characters: **char** (16 bits). It uses Unicode.
- Booleans: **boolean** (ture | false).
- Nothing: **void**.
- Strings: **String**. E.g., "Operating Systems".

All types are guaranteed to have an initial value. There exists a default value for all these types (i.e., automatically assigned).

☞ Jave does not have pointers (cannot directly access addresses as in C). Of course, it has LinkedLists, ArrayLists, HashMaps, and HashSets.

Java Variables and Literals

Variables are declared by specifying their type, name, [and initialization].

```java
int x;
double f = 0.33;
char c = 'a';
Strings = "abcd";

x = 69;
```

Integers can be written in decimal, hexadecimal, octal, and long forms:

```java
int x = 55;          // decimal value
int y = 0x37;        // hexadecimal
int z = 067;         // octal
long l = 240395922L; // long
```

The floating points are double by default:

```java
double d = 6.33  // 6.33 is a double value
float f = 6.33F; // 6.33F is a float value
```

Java Variables and Literals

Characters are specified with the standard C notation, with extensions for Unicode values:

```java
char c = 'a';
char d = '\n';
char e = '\u2122';
```

Booleans can take two possible values:

```java
boolean v1 = true;
boolean v2 = false;
```

Constants are declared with keyword **final**:

```java
final double pi = 3.145; //Constant PI
final int maxSzie = 224;
final char firstLetter - 'a';
final String coursCode = "AI224";
```

Java Expressions

- **Arithmetic.** +, -, /, *, and %.

```
int z = (((x - y) * 3) / 2) % 5;
int a +=5; int b *=3; b++; ++c;
```

- **Bit level.** Used to deal with bites:
  - & for binary AND.
  - | for binary OR.
  - $\sim$ for complement.
  - $<<$ for left shifting.
  - $>>$ for right shifting.
- **Relational.** $==$, $!=$, $<$, $>$, $<=$, and $>=$ (Returning booleans).
- **Logical.** There are three operators:
  - && for logical AND.
  - || for logical OR.
  - ! for negation.
- **Strings.** There are many. E.g., s1 + s2 is used to concatenate two strings s1 and s2, and `s1.equal(s2)` to check their equality.

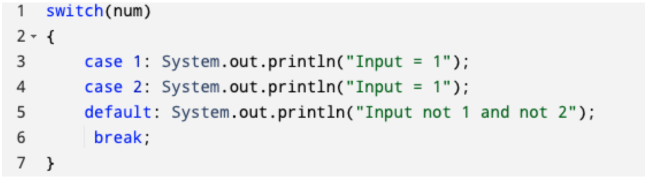
Java Control Structures (Same as in C/C++)

Conditional Statements:

- `If (Condition) { Action };`.
- `If (Condition) Action;`.
- `If (Condition) { Action 1 } else { Action 2};`.
- `If (Cnd1) { Act1 } else if (Cnd2) { Act2 } else { Act3};`

The loops:

- `while (Condition) { Action };`.
- `for (init val; Condition; step val) { Action;}`.
- `switch(var){ case 1: Act 1; ... default Act; break;}`.



```
switch(num)
{
     case 1: System.out.println("Input = 1");
     case 2: System.out.println("Input = 1");
     default: System.out.println("Input not 1 and not 2");
      break;
}
```

Note: continue and break in loop-if.

Java Arrays

Arrays are used to store a number of elements of the same type:

```java
int[] a;    //an unitialized array of interger
String[] c; //an unitialized array of Strings
int[] a = {12, 55, 2088, 5678}; //size: 4
String[] c = {"Java", "is", "great"}; //size: 3

int T;
int size = 50;
T = new int[size]; //array of 50 integers

T[4] = 699;
int len = T.length; //Get the size of the array
```

An array is an object. You can get its attributes or call a method on it.

Java Arrays

The main method has to be defined as follows:

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

The argument is an array of Strings. The i-th passed-parameter is:

```
args[i]
```

We can restrict a program to only take three arguments:

```
if(args.length != 3) System.exit(0);
```

Classes in Java

A Java class is a template for objects. It is defined using the `class` keyword and a name: `class House { ... }`.

A class contains attributes and methods.

Then, we can create instances of the class (i.e., objects) using `new`:

```
House H1 = new House();
House H2 = new House();
H3 = new House();
```

A Constructor can be used to create a customized instance of a given class. It is a void-returning method that holds the same name as the class:

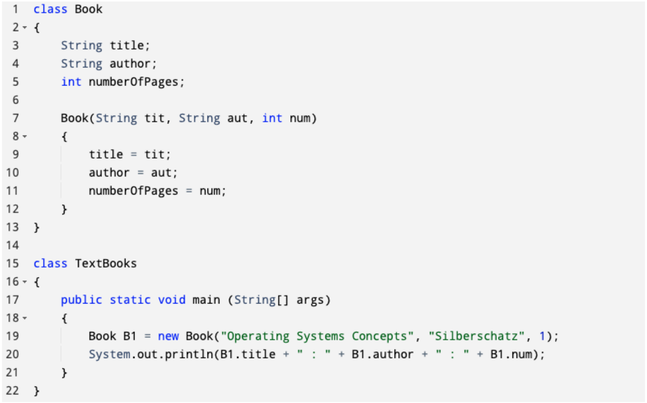
```
H1 = new House("90 Virginia Street", "K7K5Y4", "Lounis");
```

A Java class contains instance variables and methods (functions in C):

`H1.address;` — this is an attribute
`H2.owner;` — this is an attribute
`H3.AddTanent();`— this is a method

Classes in Java

Here is an example of creating an instance book of the class Books using its customized constructor:

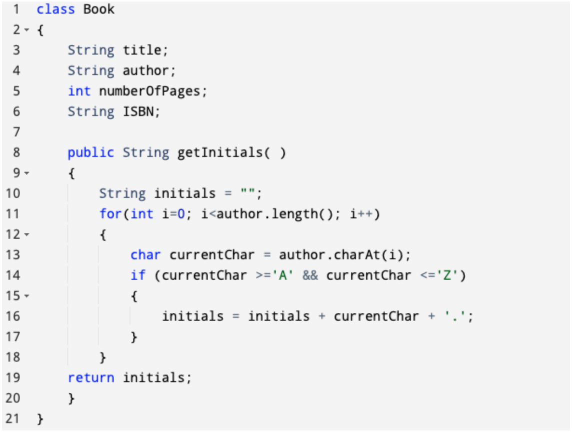


```java
class Book
{
    String title;
    String author;
    int numberOfPages;

    Book(String tit, String aut, int num)
    {
        title = tit;
        author = aut;
        numberOfPages = num;
    }
}

class TextBooks
{
    public static void main (String[] args)
    {
        Book B1 = new Book("Operating Systems Concepts", "Silberschatz", 1);
        System.out.println(B1.title + " : " + B1.author + " : " + B1.num);
    }
}
```

The default constructor is implicit. You can use it when no customized constructor is defined.

Methods in Java

Is used to implement the messages that an instance (or class) can receive:

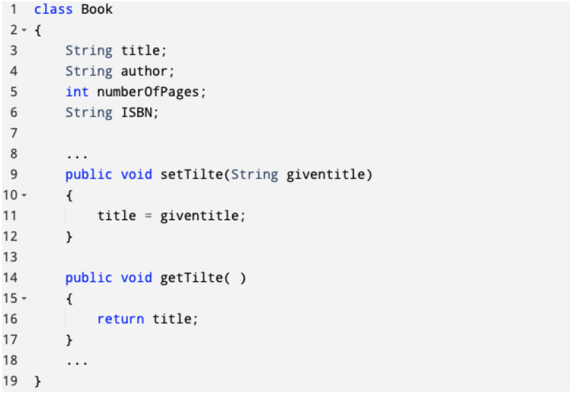


```java
class Book
{
    String title;
    String author;
    int numberOfPages;
    String ISBN;

    public String getInitials( )
    {
        String initials = "";
        for(int i=0; i<author.length(); i++)
        {
            char currentChar = author.charAt(i);
            if (currentChar >='A' && currentChar <='Z')
            {
                initials = initials + currentChar + '.';
            }
        }
    return initials;
    }
}
```

The method is called as: `String s = B1.getInitials();`

Methods in Java (Getters and Setters)

Given the variables of a class, you can create methods that can return (Getters) or update (Setters) the values of those variables:



```
class Book
{
    String title;
    String author;
    int numberOfPages;
    String ISBN;

    ...
    public void setTilte(String giventitle)
    {
        title = giventitle;
    }

    public void getTilte( )
    {
        return title;
    }
    ...
}
```

The method is called as: `String s = B1.getInitials();`

Java Public and Static Keywords

Static:

When you declare a variable or a method as static, it belongs to the class, rather than a specific instance. It will be shared by all objects.

Also, a static method belongs to the class rather than instances. Thus, it can be called without creating an instance of the class. It is used for altering the static contents of the class.

E.g., A static variable could be used by multiple cooperative threads to produce a final result.

Public:

Declares a member's access as public. Public members are visible to all other classes (i.e., can be updated by any other class if not `final`).

Demo: Complex Number Class

Java Inheritance

Inheritance allows defining a new class (subclass) by reusing another prewritten class (superclass), specifying just the difference.

```java
class it_Courses extends Courses
{
    String areaOfExpertise;
    boolean aiRelated;
}
```

Inheriting Method. New methods can be defined in the subclass.

If a method is called on the subclass, then the method is searched in that subclass first, and if not found, it is searched higher up on the hierarchy.

☞ Once we do threads in course, we're gonna have a lab that uses threads by declaring a class that inherits from the prewritten Java Thread class.

Java Input/Output

In Java, there are two data streams: input and output. They can be byte streams or character streams, buffered and non buffered:

- **Input.** Is the stream used to get data from outside (e.g., keyboard, network, files, etc). To read input from keyboard, do the following:

```
Scanner s = new Scanner(System.in);
String name = s.nextLine();
```

- **Output.** Is the stream used to output data to the outside (e.g., display, files, network, etc). To display message on the monitor, do this:

```
System.out.println("Message");
System.out.print("Message");
```

Java Packages

A package is a structure in which classes can be organized (i.e., a folder). The standard classes in the system are organized in packages. We can use them after importing them using the keyword `import`, e.g.,:

`import java.util.*;` (import all classes)
`import java.util.Scanner;` (import Scanner only)

This imported package (here `java.util`) contains the class `java.util.Date` which can be used to retrieve the current date and time.

The Java API constitutes a large library of prewritten packages and classes.

So, there are two types of packages:

- Built-in Packages (packages from the Java API): `https://docs.oracle.com/javase/8/docs/api/`
- User-defined Packages (create your own packages).

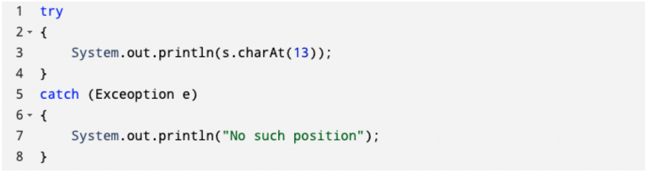
`package mypackage;`

Java Exceptions

The usual behavior on runtime errors is to abort the execution.

```
s="Hello"; System.out.println(s.charAt(13));
Exception in thread "main"
java.lang.StringIndexoutOfBoundsException:
String index out of range:  13
at java.lang.String.charAt(String.java:499)
at TestExceptions1.main(TestException1.java:11)
```

In Java, exceptions can be handled:



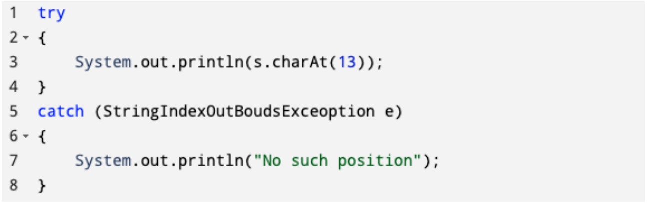
```
try
{
    System.out.println(s.charAt(13));
}
catch (Exceoption e)
{
    System.out.println("No such position");
}
```

Java Exceptions

The usual behavior on runtime errors is to abort the execution.

```
s="Hello"; System.out.println(s.charAt(13));
Exception in thread "main"
java.lang.StringIndexoutOfBoundsException:
String index out of range:  13
at java.lang.String.charAt(String.java:499)
at TestExceptions1.main(TestException1.java:11)
```

In Java, exceptions can be handled (Precisely):



```
try
{
    System.out.println(s.charAt(13));
}
catch (StringIndexOutBoudsExceoption e)
{
    System.out.println("No such position");
}
```