Module 1

Java introduction

History overview

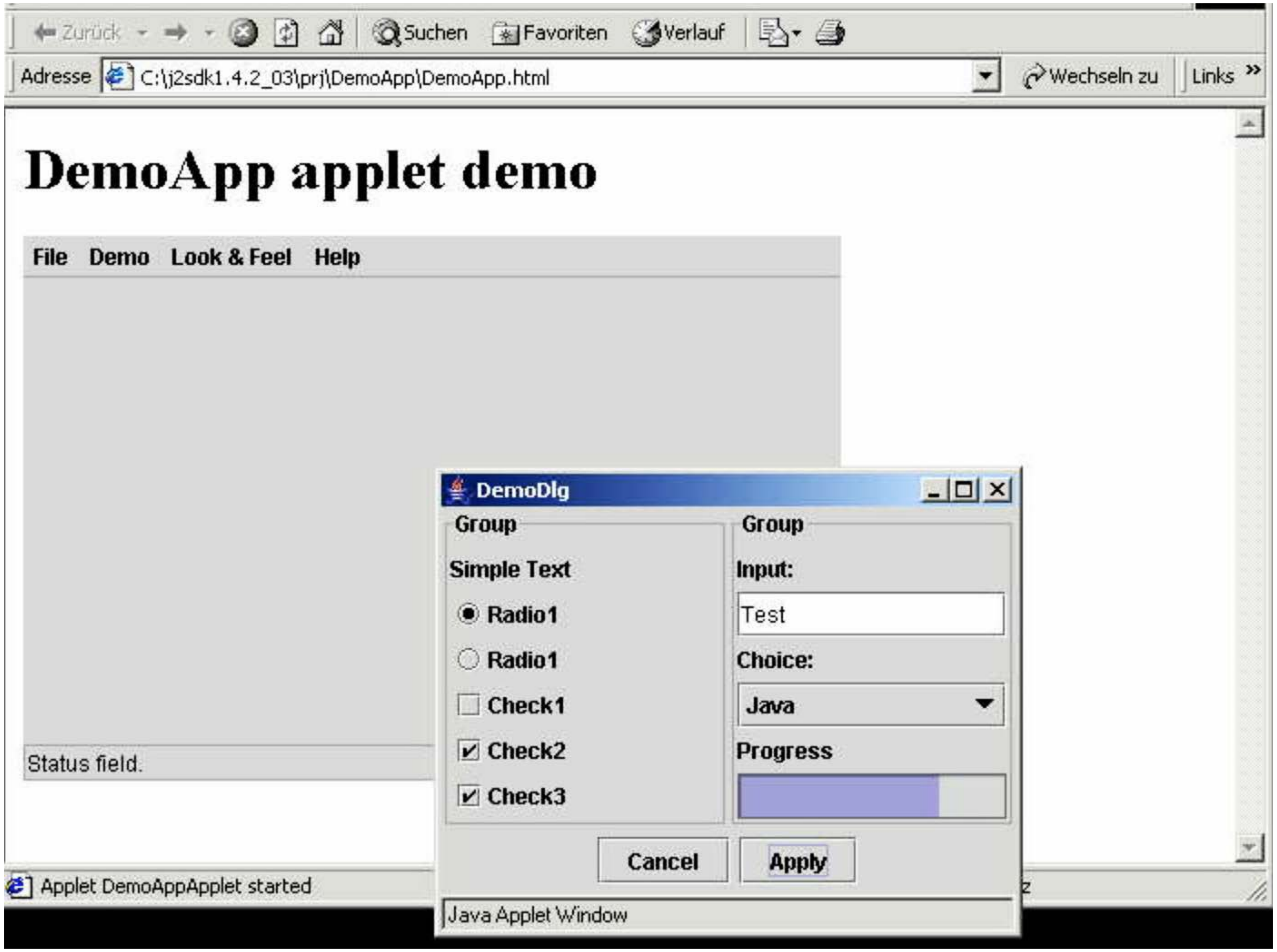
1992 - Green Project – project of Sun Microsystems to create a language for the embeddable systems – Oak. Author – James Gosling.

History overview

1992 - First demo device on new platform – PDA Star 7.

History overview

1994 - Internet epoch is coming. Java is refocused to applets development. Language is renamed to Java.



History overview

1996 - Java Development Kit.

Welcome to the Installation Wizard for Java(TM) SE Development Kit 6 Update 13

Java(TM) SE Development Kit 6 Update 13 Setup is preparing the Installation Wizard which will guide you through the program setup process. Please wait.

Versions overview

1996 – Java Development Kit.

1997 – JDK 1.1

1998 – JDK 1.2, “Java 2”, introducing ME/SE/EE editions

2000 – J2SE 1.3

2002 – J2SE 1.4

2004 – J2SE 5.0, numeration has changed.

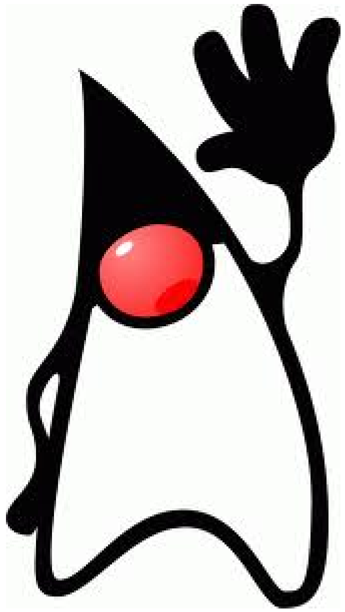
2006 – Java SE 6, leaving “J2SE”.

2011 – Java SE 7

2014 – Java SE 8

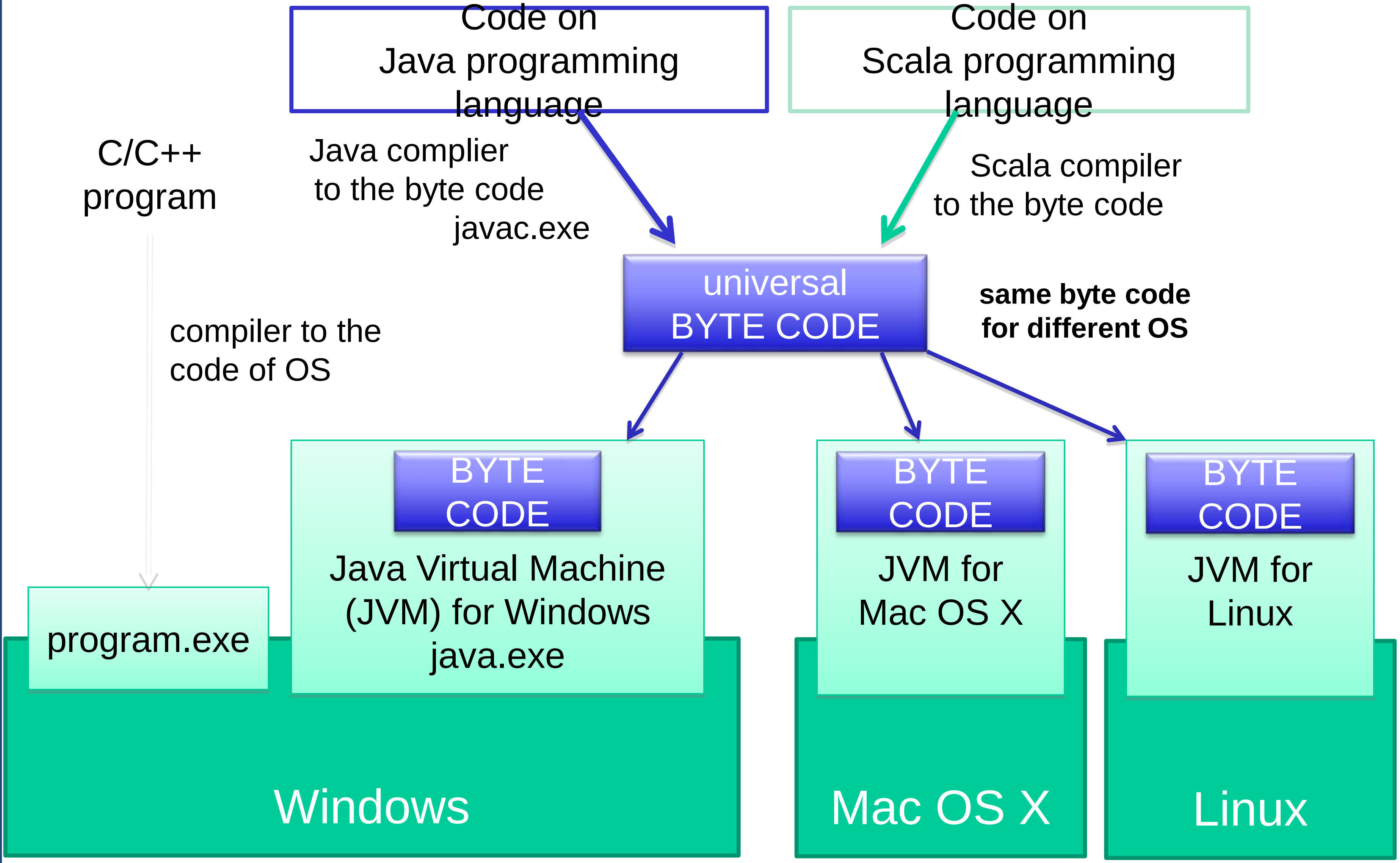
Java features

- Simplicity
- Portability
- Multithreading support
- Garbage collection
- Safety



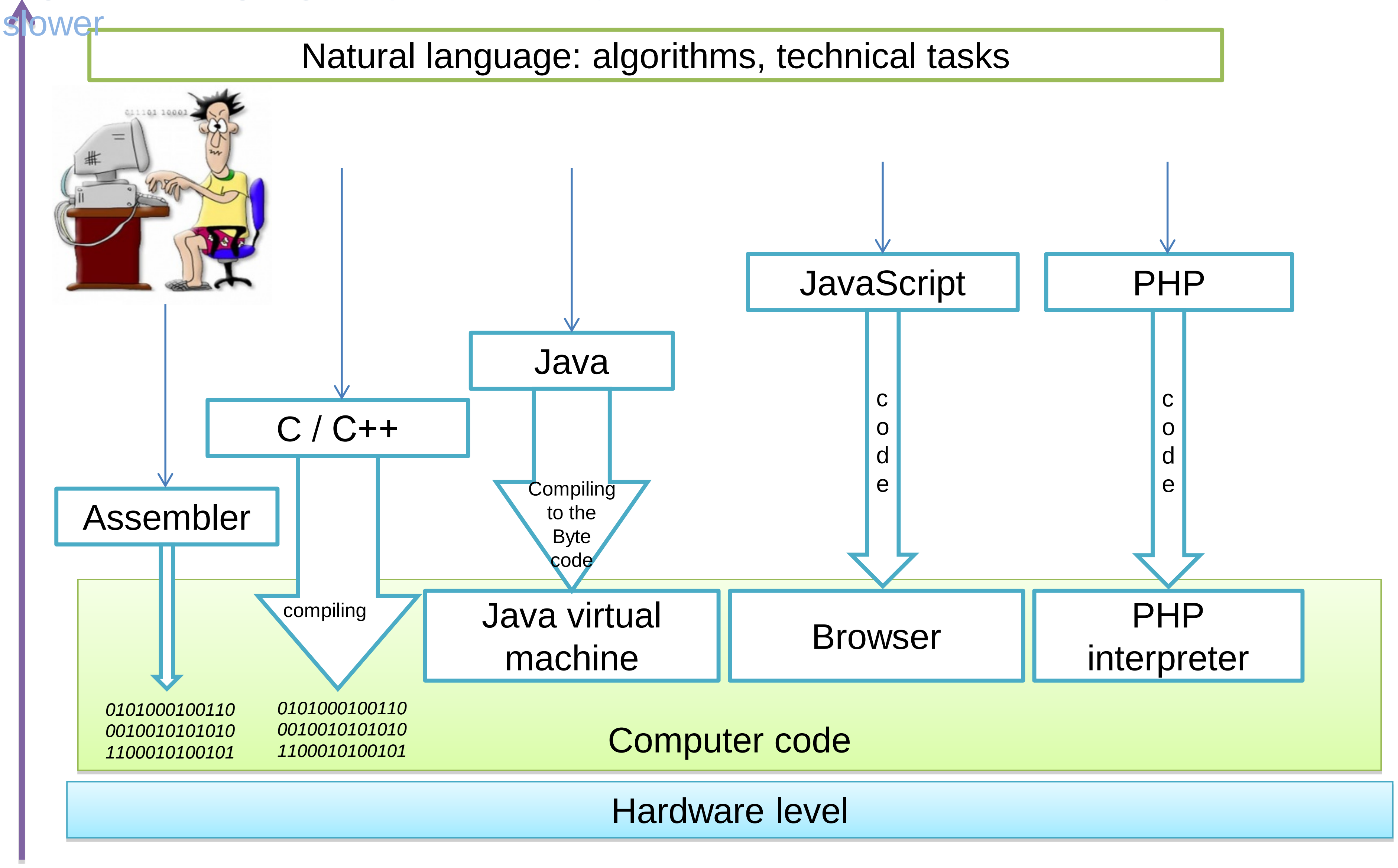
Virtual machine

WORE: **W**rite **O**nce – **R**un **E**verywhere


Code on Java programming language
Code on Scala programming language
C/C++ program
Java complier to the byte code javac.exe
Scala compiler to the byte code
universal BYTE CODE
same byte code for different OS
compiler to the code of OS
BYTE CODE
Java Virtual Machine (JVM) for Windows java.exe
BYTE CODE
JVM for Mac OS X
BYTE CODE
JVM for Linux
program.exe
Windows
Mac OS X
Linux

High and low level languages

High level languages: quick and easy to write a code, but code is running slower

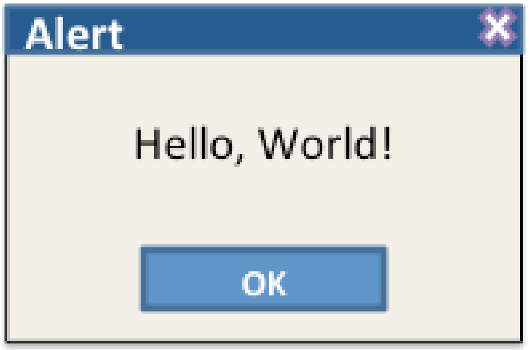


Low level languages: more difficult to write a code, code works faster

Platform and language

Platform = environment + libraries
Platform defines *which* commands to be executed
Language = syntax: defines *how to* call commands

platform	command
Windows API	CreateWindow()
.NET	New Window2().Show()
Java	new JFrame().setVisible(true)
JavaScript	window.open()



Java platform

Java language:

```
public static void main() {
   JFrame frame = new JFrame(“Alert");
   frame.setVisible(true);
}
```

Scala:

```
def main(args: Array[String]) {
   var frame: JFrame = new
JFrame(“Alert”)
   frame.setVisible(true)
}
```

.NET platform

Java for .NET:

```
public static void main() {
   Window2 win = new Window2();
   win.show();
}
```

Scala for .NET:

```
def main(args: Array[String]) {
   var win: Window2 = new
Window2(“Alert”)
   win.show()
}
```

Build tools

- Apache ANT

- Apache Maven

IDE – Integrated development environments

- Eclipse IDE
- NetBeans
- IntelliJ IDEA

Java implementation

- Oracle Java

  `http://java.oracle.com`

Java implementations

- **Oracle Java** - official implementation

  `http://java.oracle.com`

- **OpenJDK**

  `http://openjdk.java.net`

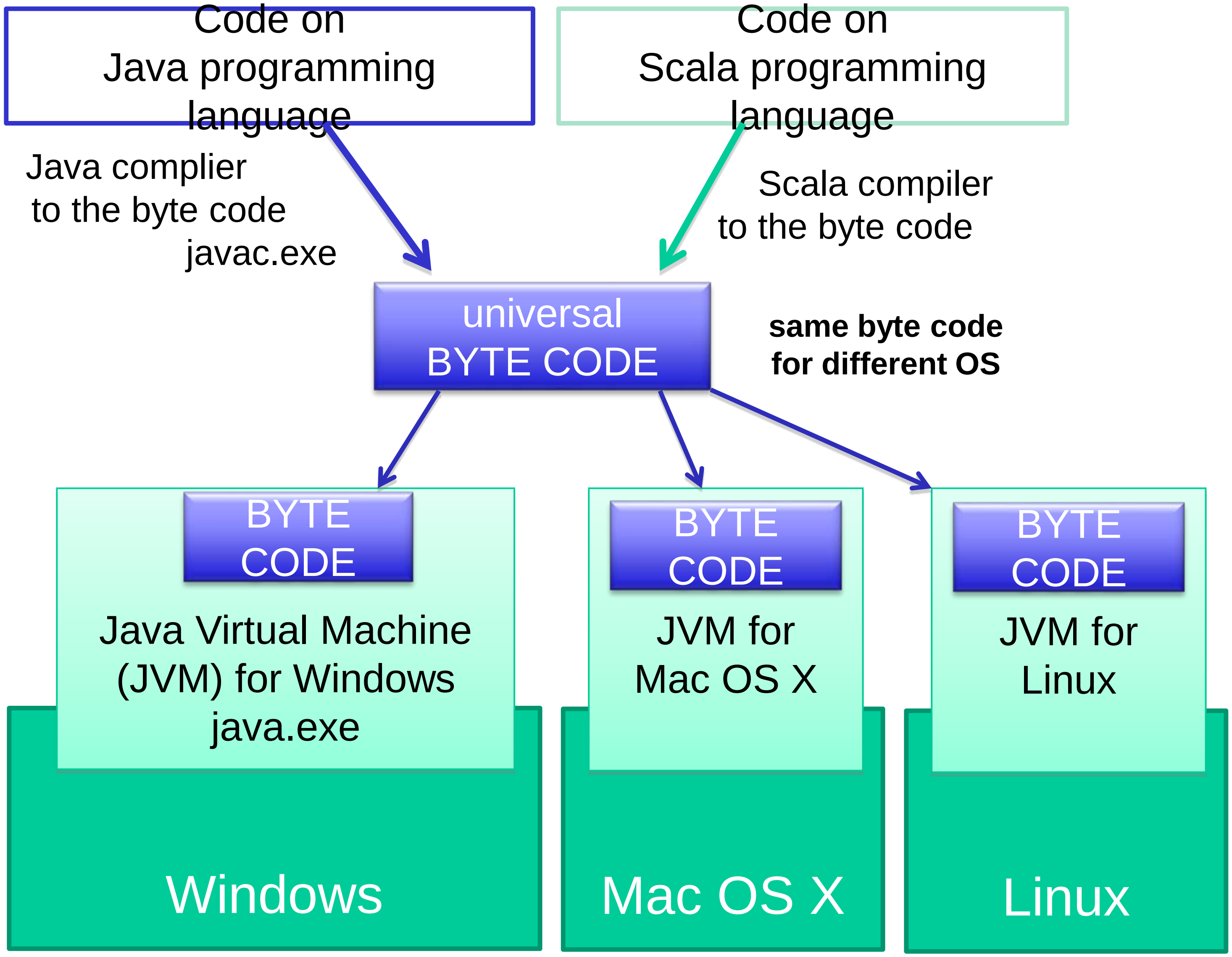
- **Iced Tea**

  `http://icedtea.classpath.org`

- **JRockit**

JVM-running languages

- Groovy
- JRuby
- Jython
- Clojure
- Scala
- Kotlin
- Rhino
- Ceylon
- Phantom
- etc.


Code on Java programming language
Code on Scala programming language
Java complier to the byte code javac.exe
Scala compiler to the byte code
universal BYTE CODE
same byte code for different OS
BYTE CODE
Java Virtual Machine (JVM) for Windows java.exe
BYTE CODE
JVM for Mac OS X
BYTE CODE
JVM for Linux
Windows
Mac OS X
Linux

Java editions

- Java SE (Standard Edition)
- Java ME (Micro Edition)
- Java EE (Enterprise Edition)

Java modes

- **Client mode**

  The Client VM compiler does not try to execute many of the more complex optimizations performed by the compiler in the Server VM, but in exchange, it requires less time to analyze and compile a piece of code. This means the Client VM can start up faster and requires a smaller memory footprint.

- **Server mode**

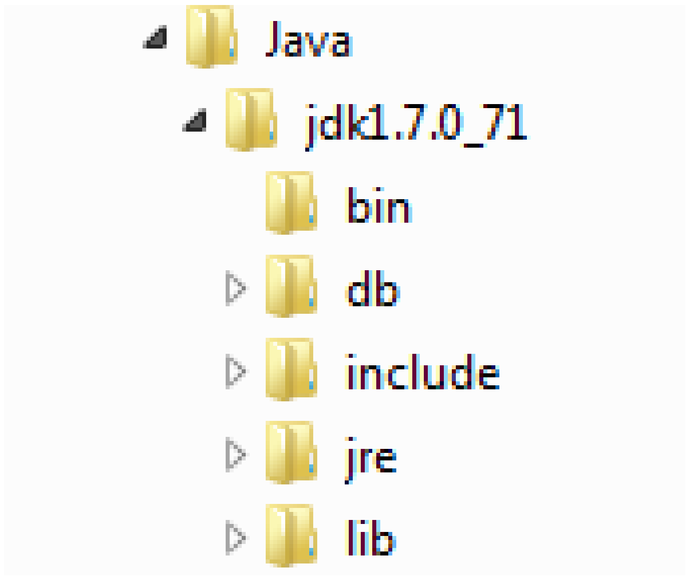
  The Server VM contains an advanced adaptive compiler that supports many of the same types of optimizations performed by optimizing C++ compilers, as well as some optimizations that cannot be done by traditional compilers, such as aggressive inlining across virtual method invocations.

```
java -client
java -server
```

In "client" mode JVM gives some unused memory back to the operating system - whereas with "server" mode, once the JVM grabs the memory, it won't give it back.

JDK overview

- JDK can be downloaded at www.oracle.com/technetwork/java.
- JDK is installed to C:\Program Files\Java by default
- JDK can be installed only with JRE



JDK tools overview

- javac - compiler
- java – JVM implementation
- jar – .class archiver
- javadoc – documentation generator
- jdb – debugger

javac

- **Java Compiler**

- Compiles Java code (*.java) to byte code (*.class)

  - `javac     MyClass.java  OneMoreClass.java`
  - `javac    -d classes     MyClass.java`
  - `javac    -classpath     library.jar -d classes MyClass.java`
  - `javac    -version`

JavaDoc

JavaDoc tags list

@author	Author of class/interface
@version	Version of the class/interface
@since	Since which version accessible
@see	Link to another place in documentation
@param	Method parameter
@return	Returned value
@throws	Exception description
@deprecated	Marks old code

JavaDoc

```
/**
 * This is the simplest Java class. It prints the “Hello, World”
message.
 * @author Peter Pan
 */
public class HelloWorld {

    /**
     * Definition for hello world message.
     */
    public static final String HELLO_MESSAGE = "Hello, World";

    /**
     * Main methods which is executed by JVM and prints the message.
     * @param args Command line arguments
     */
    public static void main(String[] args) {
        System.out.println(HELLO_MESSAGE);
    }
}
```

- javadoc HelloWorld.java
- index.html will be generated

Exercise

Lab guide:

- Exercise 1
- Exercise 2