Java Introduction

Java Origins

Computer language innovation and development occurs for two fundamental reasons:

1) to adapt to changing environments and uses
2) to implement improvements in the art of programming

The development of Java was driven by both in equal measures.

Many Java features are inherited from the earlier languages:

B → C → C++ → Java

Before Java: C

Designed by Dennis Ritchie in 1970s.

Before C, there was no language to reconcile: ease-of-use versus power, safety versus efficiency, rigidity versus extensibility.

BASIC, COBOL, FORTRAN, PASCAL optimized one set of traits, but not the other.

C- structured, efficient, high-level language that could replace assembly code when creating systems programs.

Designed, implemented and tested by programmers, not scientists.

Before Java: C++

Designed by Bjarne Stroustrup in 1979.

Response to the increased complexity of programs and respective improvements in the programming paradigms and methods:

1) assembler languages
2) high-level languages
3) structured programming
4) object-oriented programming (OOP)

OOP – methodology that helps organize complex programs through the use of inheritance, encapsulation and polymorphism.

C++ extends C by adding object-oriented features.

Java History

Designed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems in 1991.

The original motivation is not Internet: platform-independent software embedded in consumer electronics devices.

With Internet, the urgent need appeared to break the fortified positions of Intel, Macintosh and Unix programmer communities.

Java as an “Internet version of C++”? No.

Java was not designed to replace C++, but to solve a different set of problems. There are significant practical/philosophical differences.

Java Technology

There is more to Java than the language.

Java Technology consists of:

1) Java Programming Language
2) Java Virtual Machine (JVM)
3) Java Application Programming Interfaces (APIs)

Java Language Features

1) simple
2) object-oriented
3) robust
4) multithreaded
5) architecture-neutral
6) interpreted and high-performance
7) distributed
8) dynamic
9) secure

Java Language Features 1

1) simple – Java is designed to be easy for the professional programmer to learn and use.
2) object-oriented – a clean, usable, pragmatic approach to objects, not restricted by the need for compatibility with other languages.
3) robust – restricts the programmer to find the mistakes early, performs compile-time (strong typing) and run-time (exception-handling) checks, manages memory automatically.

Java Language Features 2

4) multithreaded – supports multi-threaded programming for writing program that perform concurrent computations
5) architecture-neutral – Java Virtual Machine provides a platform-independent environment for the execution of Java bytecode
6) interpreted and high-performance – Java programs are compiled into an intermediate representation – bytecode:
   a) can be later interpreted by any JVM
   b) can be also translated into the native machine code for efficiency.

Java Language Features 3

7) distributed – Java handles TCP/IP protocols, accessing a resource through its URL much like accessing a local file.
8) dynamic – substantial amounts of run-time type information to verify and resolve access to objects at run-time.
9) secure – programs are confined to the Java execution environment and cannot access other parts of the computer.

Execution Platform

What is an execution platform?

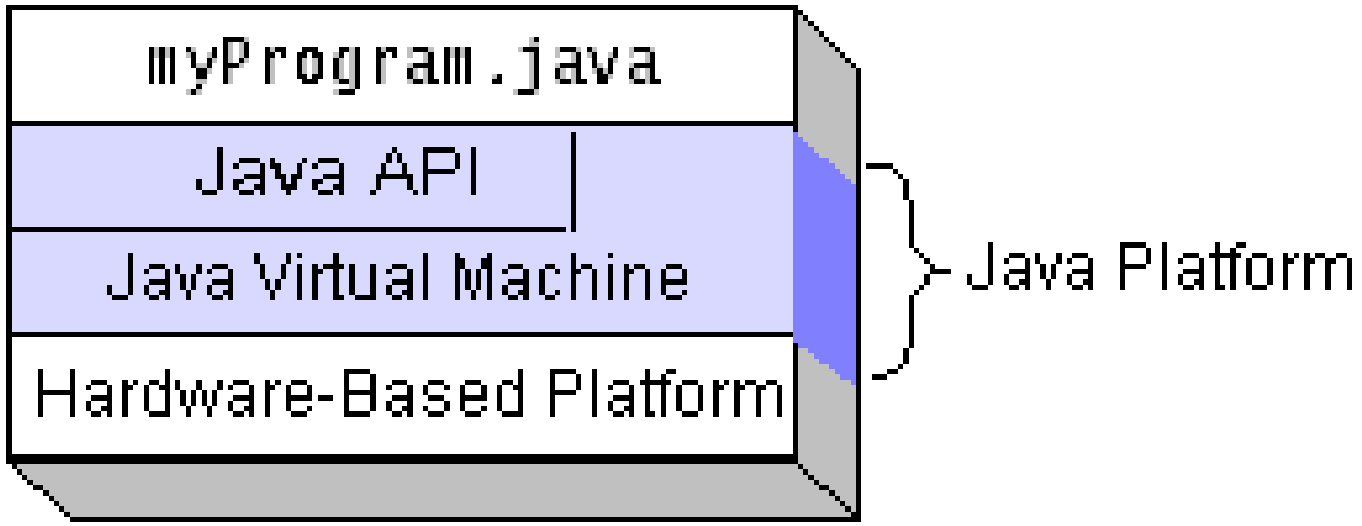
1) An execution platform is the hardware or software environment in which a program runs, e.g. Windows, Linux, Solaris or MacOS.
2) Most platforms can be described as a combination of the operating system and hardware.

Java Execution Platform

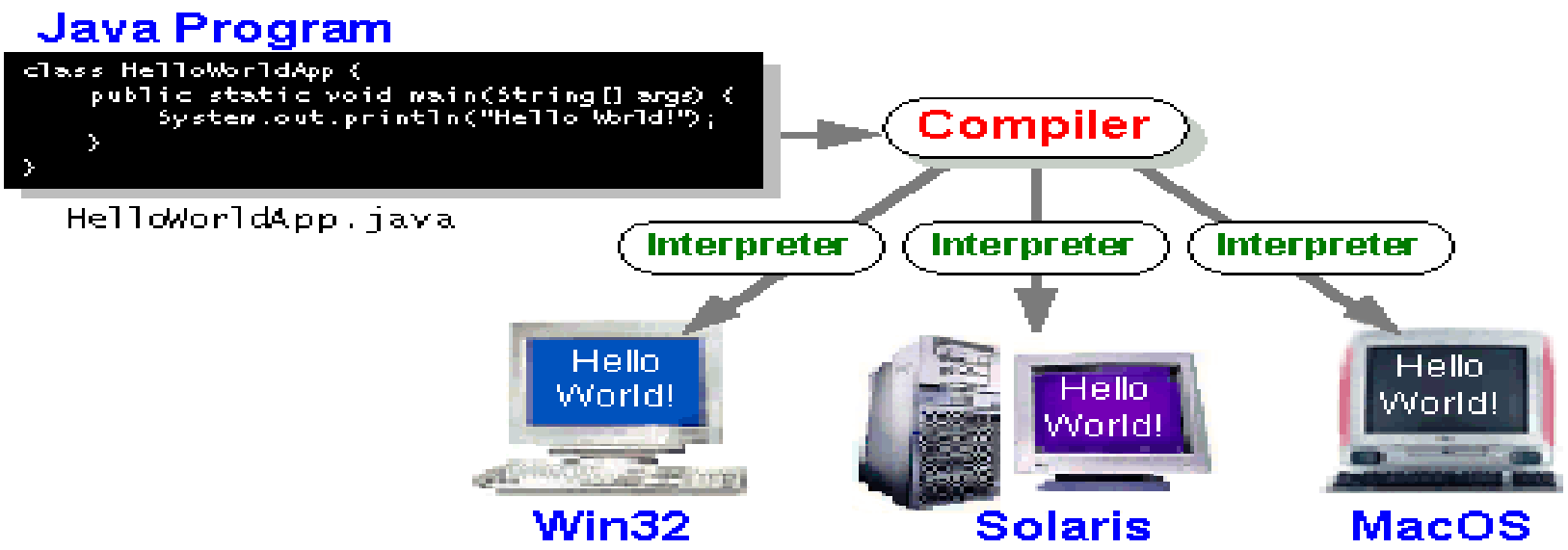
What is Java Platform?

1) A software-only platform that runs on top of other hardware-based platforms.
2) Java Platform has two components:
    a) Java Virtual Machine (JVM) – interpretation for the Java bytecode, ported onto various hardware-based platforms.
    b) The Java Application Programming Interface (Java API)

Java Execution Platform



Java Platform Independence



Java Program Execution

Java programs are both compiled and interpreted:

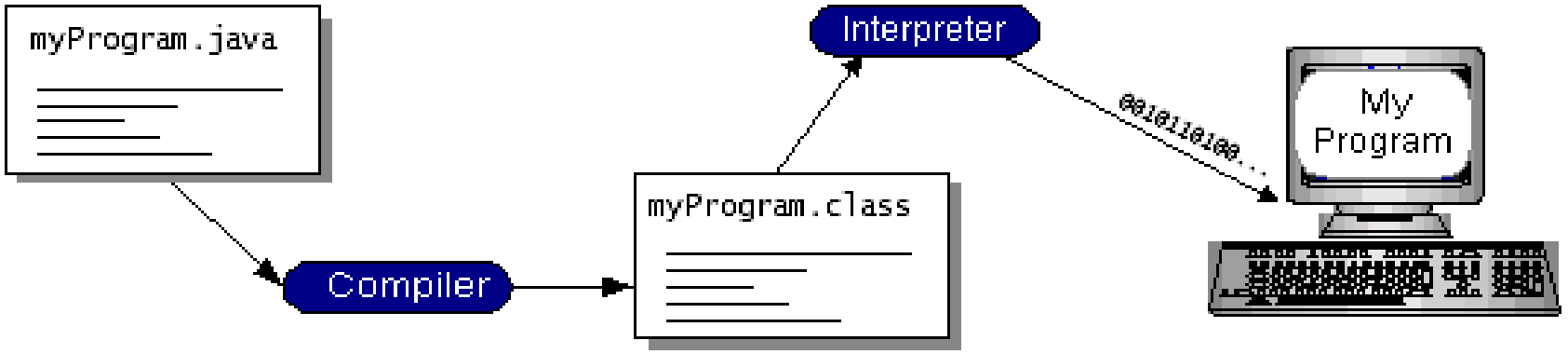
Steps:

- write the Java program
- compile the program into bytecode
- execute (interpret) the bytecode on the computer through the Java Virtual Machine

Compilation happens once.

Interpretation occurs each time the program is executed.

Java Execution Process



Java API

What is Java API?

1) a large collection of ready-made software components that provide many useful capabilities, e.g. graphical user interface
2) grouped into libraries (packages) of related classes and interfaces
3) together with JVM insulates Java programs from the hardware and operating system variations

Java Program Types

Types of Java programs:

1) applications – standalone (desktop) Java programs, executed from the command line, only need the Java Virtual Machine to run
2) applets – Java program that runs within a Java-enabled browser, invoked through a “applet” reference on a web page, dynamically downloaded to the client computer
3) servlets – Java program running on the web server, capable of responding to HTTP requests made through the network
4) etc.

Java Platform Features 1

1) essentials - objects, strings, threads, numbers, input/output, data structures, system properties, date and time, and others.
2) networking:
    1) Universal Resource Locator (URL)
    2) Transmission Control Protocol (TCP)
    3) User Datagram Protocol (UDP) sockets
    4) Internet Protocol (IP) addresses
3) internationalization - programs that can be localized for users worldwide, automatically adapting to specific locales and appropriate languages.

Java Platform Features 2

4) security – low-level and high-level security, including electronic signatures, public and private key management, access control, and certificates
5) software components – JavaBeans can plug into an existing component architecture
6) object serialization - lightweight persistence and communication, in particular using Remote Method Invocation (RMI)
7) Java Database Connectivity (JDBC) - provides uniform access to a wide range of relational databases

Java Technologies

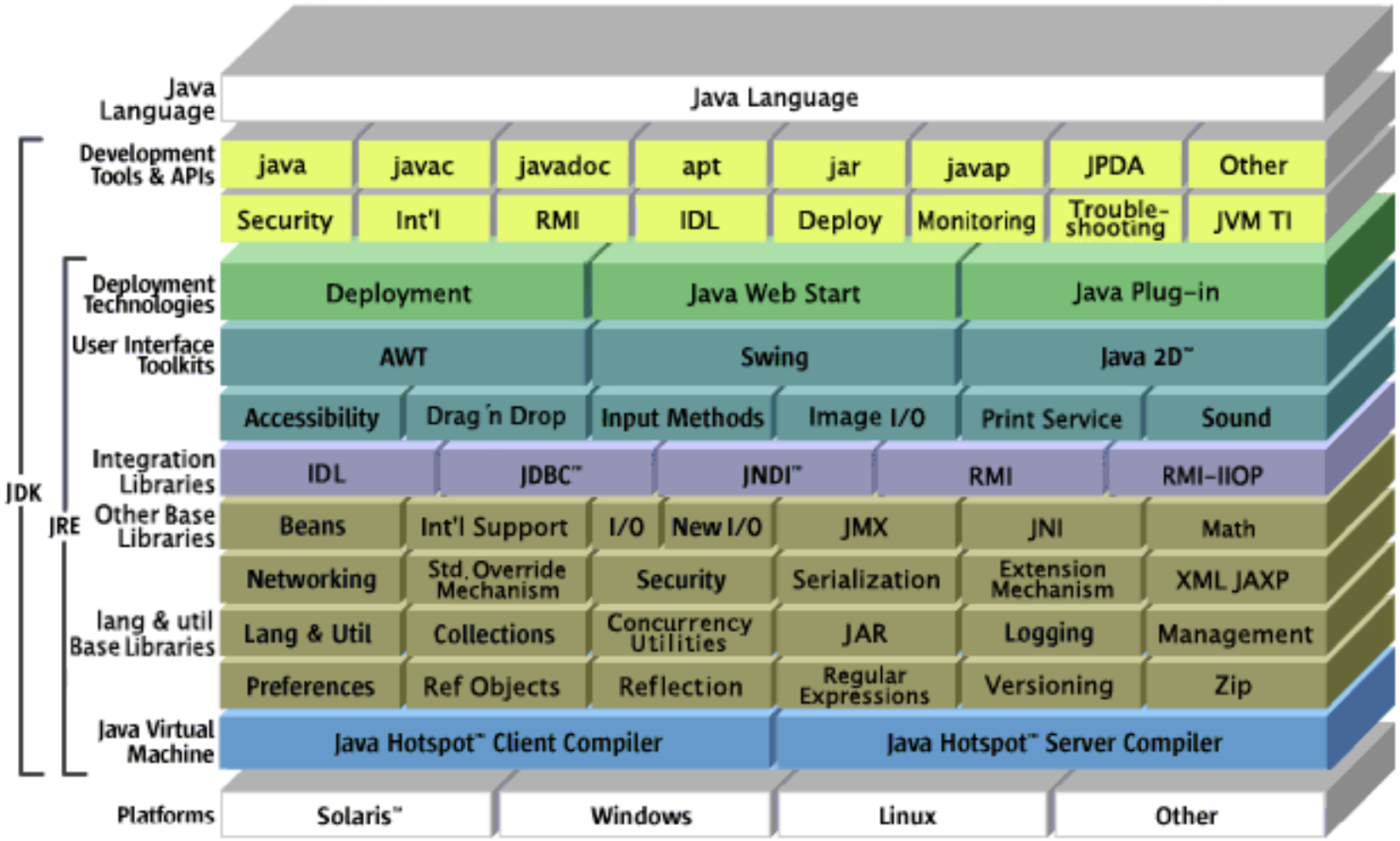
Different technologies depending on the target applications:

1) desktop applications - Java 2 Standard Edition (J2SE)
2) enterprise applications – Java 2 Enterprise Edition (J2EE)
3) mobile applications – Java 2 Mobile Edition (J2ME)
4) smart card applications – JavaCard
5) etc.

Each edition puts together a large collections of packages offering functionality needed and relevant to a given application.

The Java Virtual Machine remains essentially the same.

Java Technology: SDK



Exercise: Java Technology

1) Explain the statement “There is more to Java than the Language”.
2) Enumerate and explain Java design goals.
3) How does Java maintain a balance between Interpretation and High Performance?.
4) Java program is termed “Write once run everywhere”. Explain.
5) Why is it difficult to write viruses and malicious programs with Java?

Simple Java Program

A class to display a simple message:

```
class MyProgram {
      public static void main(String[] args) {
            System.out.println("First Java program.");
      }
}
```

Running the Program

Type the program, save as `MyProgram.java`.

In the command line, type:

```
> dir
MyProgram.java
> javac MyProgram.java
> dir
MyProgram.java, MyProgram.class
> java MyProgram
First Java program.
```

Explaining the Process

1) creating a source file - a source file contains text written in the Java programming language, created using any text editor on any system.

2) compiling the source file Java compiler (javac) reads the source file and translates its text into instructions that the Java interpreter can understand. These instructions are called bytecode.

3) running the compiled program Java interpreter (java) installed takes as input the bytecode file and carries out its instructions by translating them on the fly into instructions that your computer can understand.

Java Program Explained

`MyProgram` is the name of the class:

```
class MyProgram {
```

`main` is the method with one parameter `args` and no results:

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

`println` is a method in the standard `System` class:

```
      System.out.println("First Java program.");
   }
}
```

Classes and Objects

A class is the basic building block of Java programs.

A class encapsulates:

a) data (attributes) and

b) operations on this data (methods)

and permits to create objects as its instances.

Main Method

The `main` method must be present in every Java application:

1) `public static void main(String[] args)` where:
   a) `public` means that the method can be called by any object
   b) `static` means that the method is shared by all instances
   c) `void` means that the method does not return any value
2) When the interpreter executes an application, it starts by calling its `main` method which in turn invokes other methods in this or other classes.
3) The main method accepts a single argument – a string array, which holds all command-line parameters.

Exercise: Java Program

1) Personalize the `MyProgram` program with your name so that it tells you "`Hello, my name is …`"
2) Write a program that produces the following output:

```
Welcome to e-Macao Java Workshop!
I hope you will benefit from the training.
```

3) Here is a slightly modified version of `MyProgram`:

```
class MyProgram2 {
   public void static Main(String args) {
     system.out.println("First Java Program!);
   }
}
```

The program has some errors. Fix the errors so that the program successfully compiles and runs. What were the errors?

Using API Documentation 1

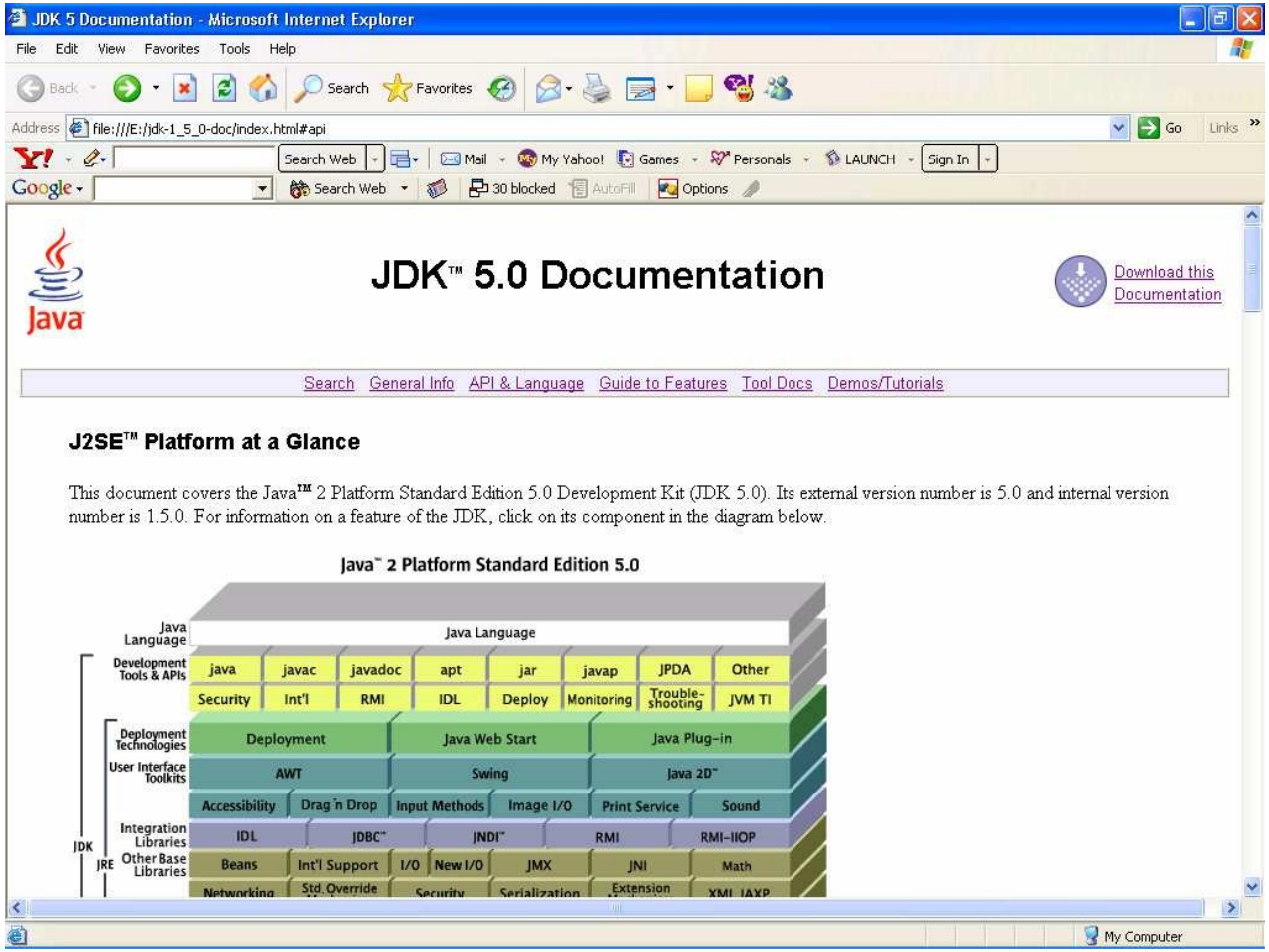
Ability to use Java API Specification is crucial for any practicing programmer. Here are the steps:

1) download the API documentation from
   http://java.sun.com/j2se/1.5.0/download.jsp

2) Extract the archive file

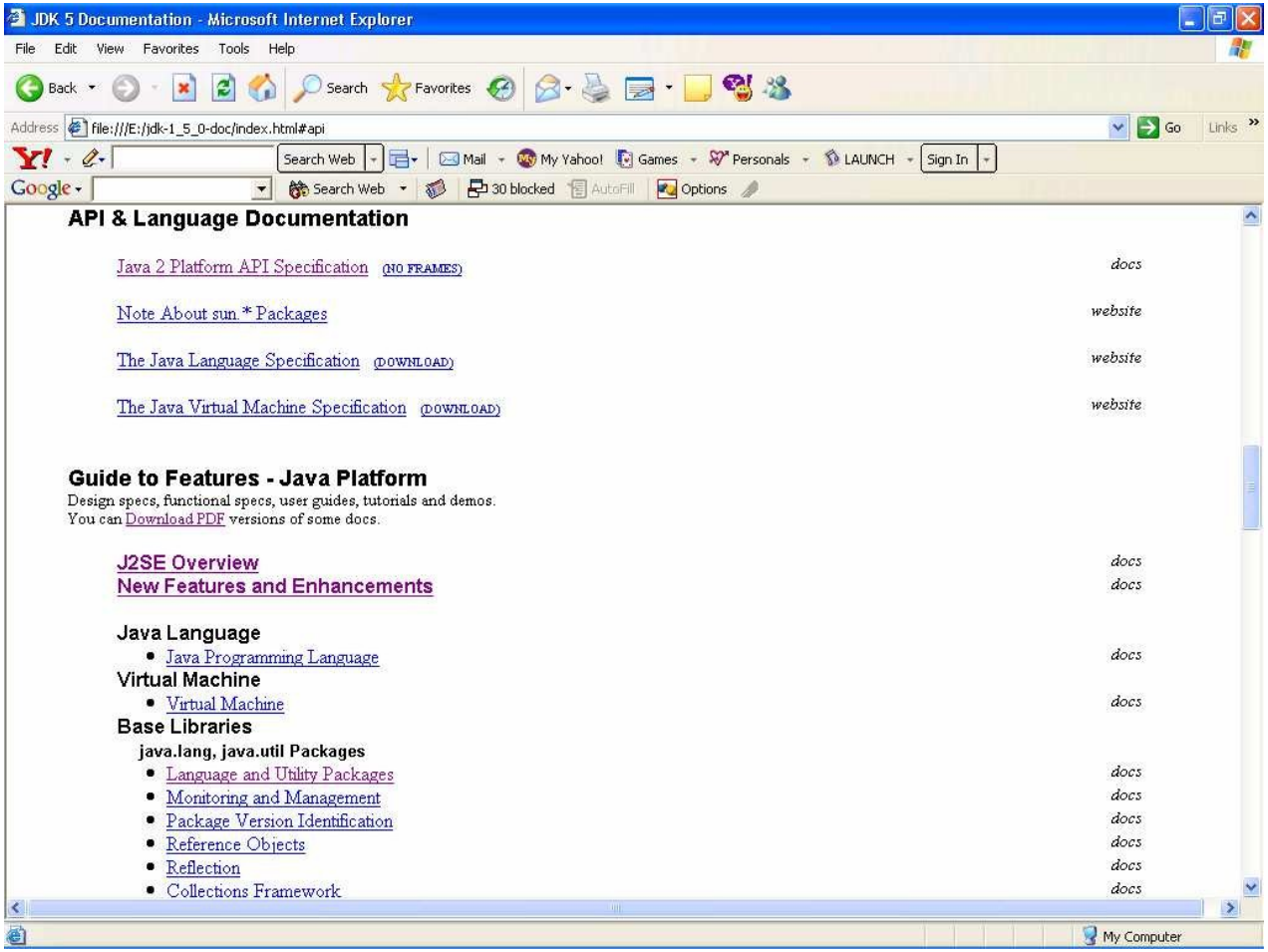
Using API Documentation 2

3) open `index.html`



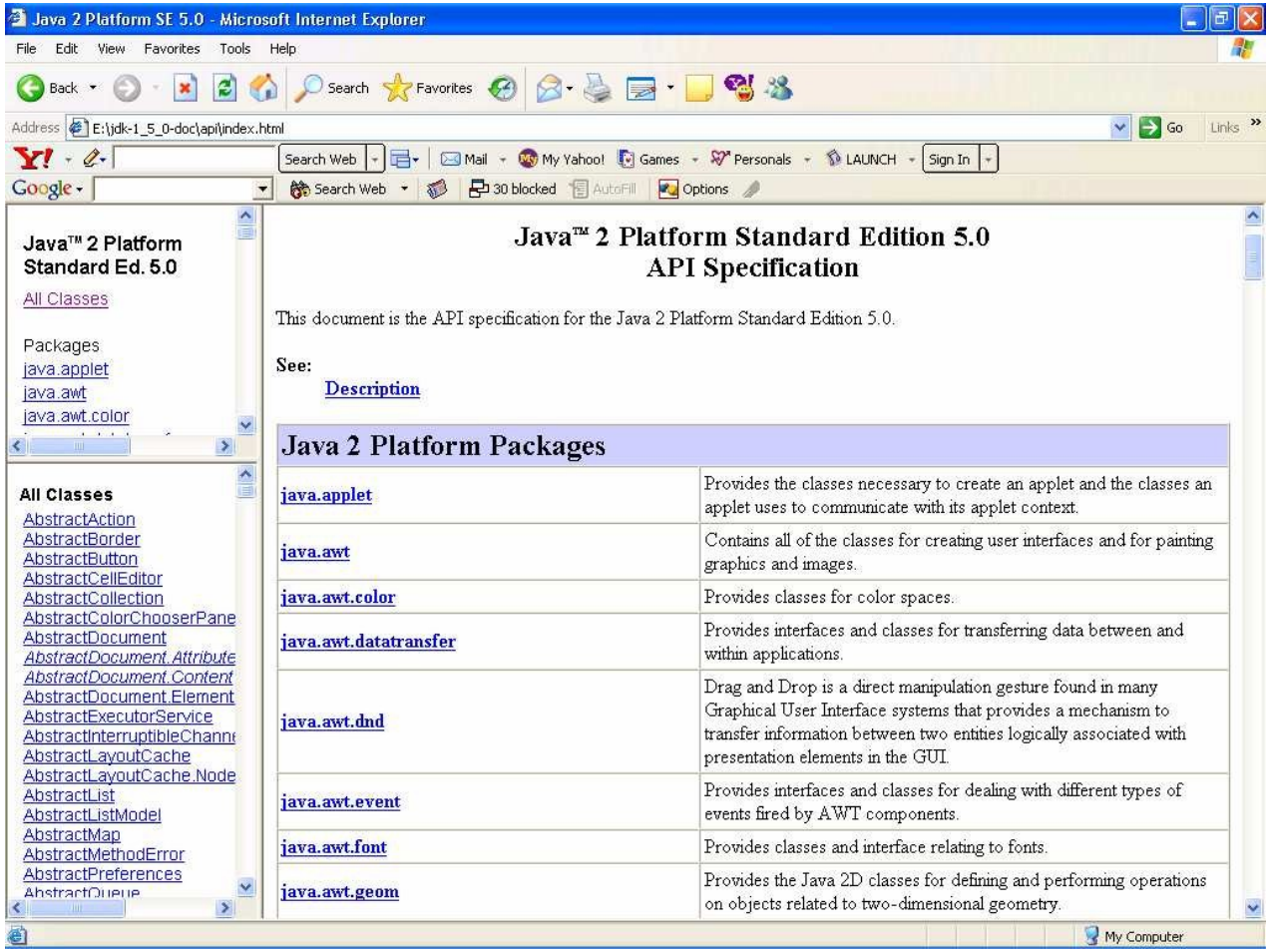
Using API Documentation 3

4) click on `API & Language`



Using API Documentation 4

5) click on `Java 2 Platform API Specification`



Java 2 Platform API Specification

The window is divided into three panes:

1) specification pane - click on any package, package pane will display all classes, interfaces, exceptions and errors that belong to that package.
2) package pane - click on any class, class pane will display all information about the class.
3) class pane – contains such information as:
    1) inheritance hierarchy
    2) implemented interfaces
    3) constructors
    4) attributes
    5) methods

Java Environment Setup 1

Setting up `JAVA_HOME` and `PATH` environment variables is necessary for the Java tools and applications to work properly from any directory.

Steps to carry out:

1) `JAVA_HOME` variable should point to the top installation directory of Java environment.
2) `PATH` should point to the `bin` directory where the Java Virtual Machine - interpreter (`java`), compiler (`javac`) and other executables are located.

Java Environment Setup 2

Example:

1) Windows - add `%JAVA_HOME%\bin` to the beginning of the `PATH` variable.
2) Linux - include `PATH="$PATH:$JAVA_HOME/bin:."` in your `/etc/profile` or `.bashrc` files.

To test the environment, open the command window and run:

```
java -version
```

The current version number of the installed JVM should appear.

Exercise: API Specification

1) Download and extract the latest JDK Documentation.
2) Using the Documentation
   a) Locate the `java.math` package
   b) How many classes does it have?
   c) List all the classes.
   d) List their inheritance hierarchies.
   e) List their implemented interfaces.
   f) How many constructors does each have?
   g) How many fields does each have?
   h) How many methods does each have?