HBO Graduaat Informatica
Optie Programmeren

Java Basics

JRE and JDK Tools

cvo leerstad

volwassenenonderwijs

Chapter Overview

- JRE: virtual machine
- JDK: compiler and accessory tools
- Text editors
- Integrated Development Environment

Introduction

- Objectives
  - To explain the basics of Java
  - To introduce the components of Java
  - To give you a feel for the Java environment
- Chapter content
  - Basics - history and design criterion of Java
  - VM - the Java Virtual Machine
  - Apps - types of Java program: applications and applets
  - Environment - the Java development environment
- Practical content
  - Familiarisation with Java development environment

What is Java?

- Summary
- An object oriented language with associated object-based API
- Designed by Sun in early 1990s
  - Needed a safe alternative to C++ for consumer electronics
- Developed from scratch
  - Looks like C and C++ to make it familiar to many programmers
  - Does not have the syntactical redundancy of C++
- Sun’s Java Language White Paper describes it as:

> A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded and dynamic language.

Java Design Criteria

1. Platform-independence
2. Robust
3. Small and fast
4. Secure

Design Criteria 1

Platform-independence

- Java code (*xyz.java*) is compiled into processor-independent bytecodes (*xyz.class*)
- Bytecodes are then interpreted, like Basic, so same program runs on any platform that supports Java

Design Criteria 2

Robust

- Fully object oriented: every line of code belong to a class
- Strict type checking
- Built-in and enforced exception handling
- No pointers!

Design Criteria 3

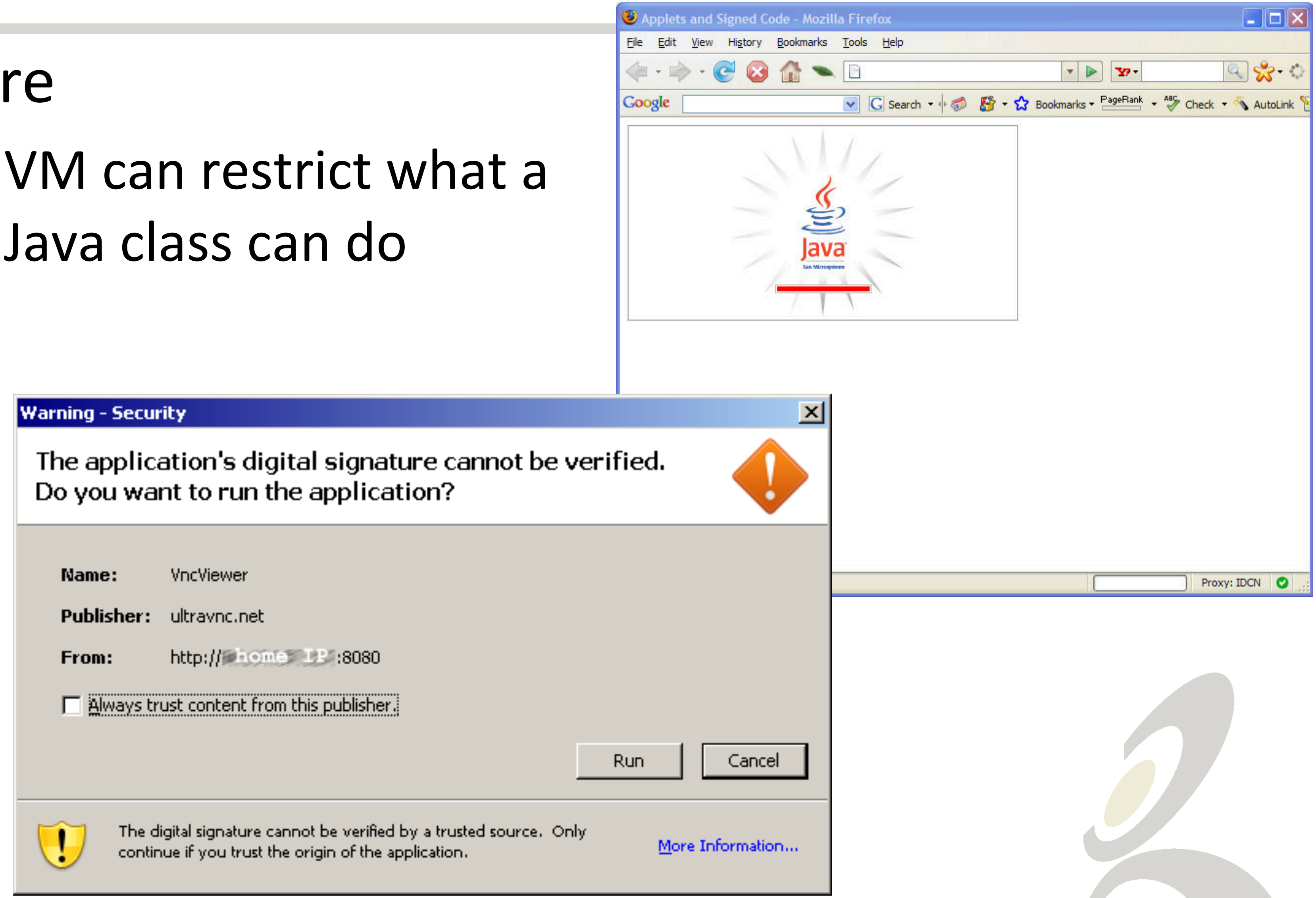
Small and fast

- Each class is only loaded if needed
- Built-in multi-threading

Design Criteria 4

Secure

- VM can restrict what a Java class can do


Applets and Signed Code - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
Search
Bookmarks
PageRank
Check
AutoLink
Java
Proxy: IDCN

Warning - Security

The application's digital signature cannot be verified.
Do you want to run the application?

Name: VncViewer

Publisher: ultravnc.net

From: http://home IP:8080

□ Always trust content from this publisher.

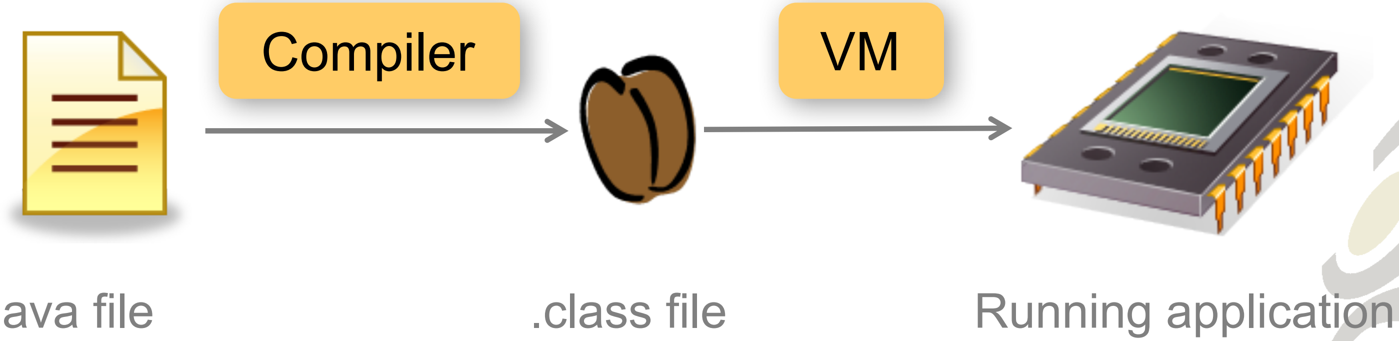
Run Cancel

The digital signature cannot be verified by a trusted source. Only continue if you trust the origin of the application.

More Information...

How Does Java Run?

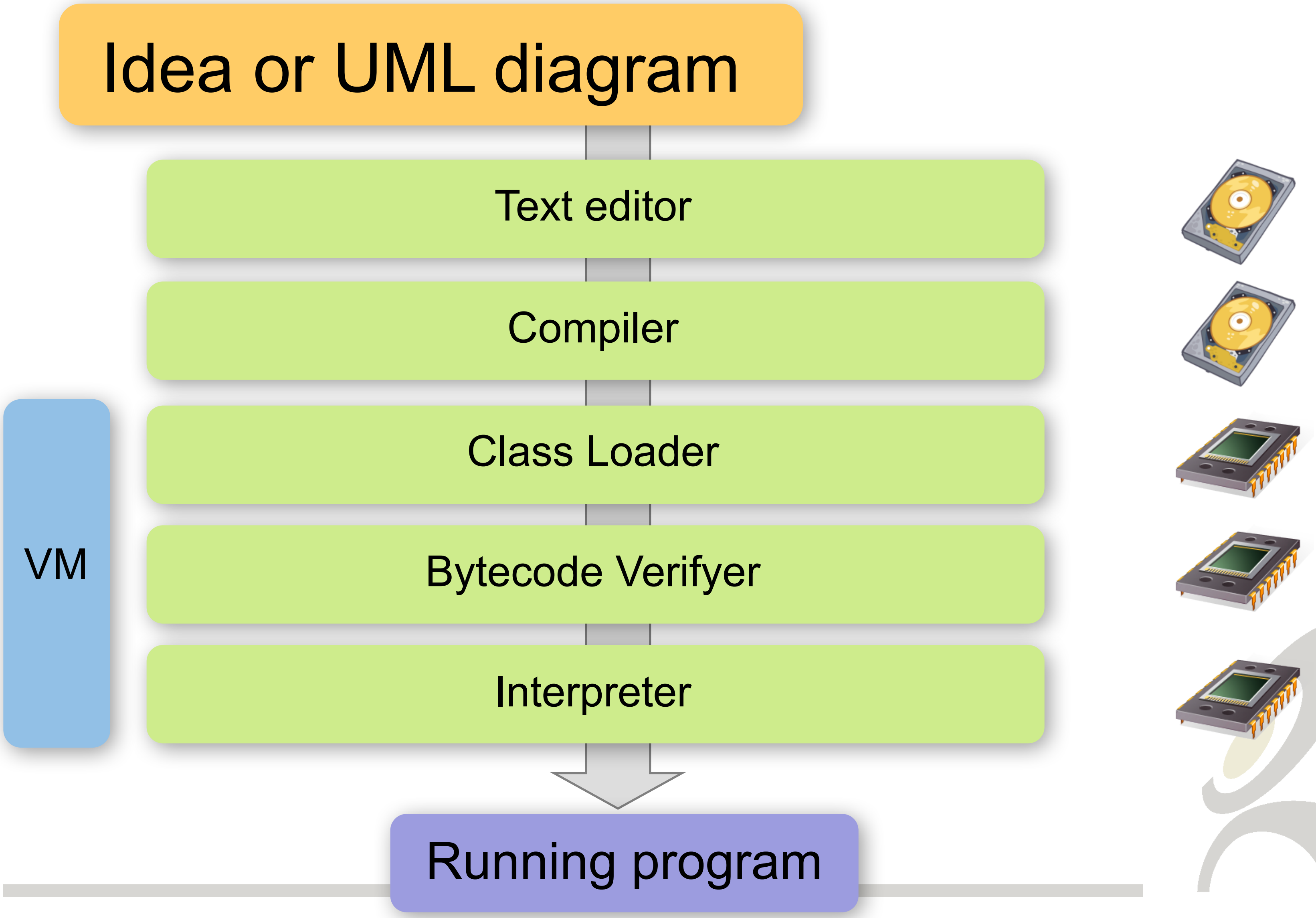
- Java runs within the Java Virtual Machine (VM)
  - This must be ported to the appropriate platform
  - Built into browsers such as Mozilla Firefox and Microsoft Internet Explorer
- Java VM runs the given `.class` file



Java Virtual Machine

- In order to run a Java program, you need a Java Virtual Machine
- Also called Java Runtime Environment (JRE)
  - Understands *.class* files independently of platform (operating system)
  - Provides implementation of the required API classes (libraries) such as Input/Output, graphical user environment (GUI), XML parsing, ...

The Java Virtual Machine (VM)



Idea or UML diagram

Text editor

Compiler

Class Loader

Bytecode Verifyer

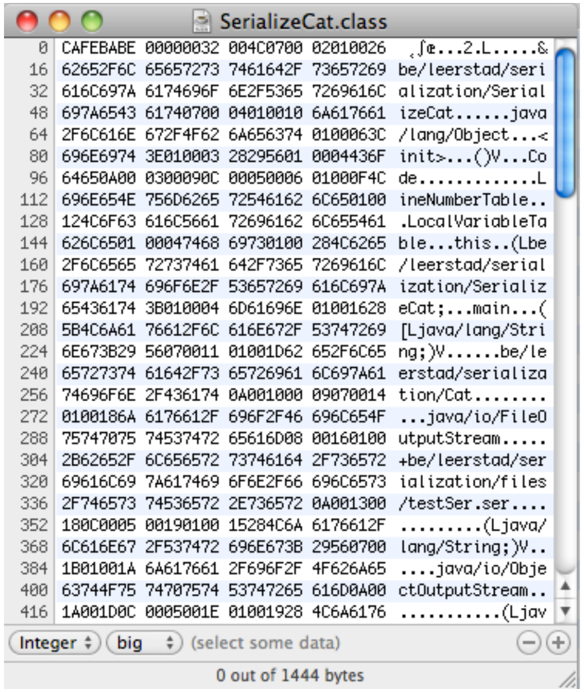
Interpreter

VM

Running program

Java Virtual Machine

1. Class loader finds required class
   - E.g. by searching `CLASSPATH`



SerializeCat.class

```
  0  CAFEBABE 00000032 004C0700 02010026   ¸∫æ...2.L.....&
 16  62652F6C 65657273 7461642F 73657269  be/leerstad/seri
 32  616C697A 6174696F 6E2F5365 7269616C  alization/Serial
 48  697A6543 61740700 04010010 6A617661  izeCat......java
 64  2F6C616E 672F4F62 6A656374 0100063C  /lang/Object...<
 80  696E6974 3E010003 28295601 0004436F  init>...()V...Co
 96  64650A00 0300090C 00050006 01000F4C  de.............L
112  696E654E 756D6265 72546162 6C650100  ineNumberTable..
128  124C6F63 616C5661 72696162 6C655461  .LocalVariableTa
144  626C6501 00047468 69730100 284C6265  ble...this..(Lbe
160  2F6C6565 72737461 642F7365 7269616C  /leerstad/serial
176  697A6174 696F6E2F 53657269 616C697A  ization/Serializ
192  65436174 3B010004 6D61696E 01001628  eCat;...main...(
208  5B4C6A61 76612F6C 616E672F 53747269  [Ljava/lang/Stri
224  6E673B29 56070011 01001D62 652F6C65  ng;)V......be/le
240  65727374 61642F73 65726961 6C697A61  erstad/serializa
256  74696F6E 2F436174 0A001000 09070014  tion/Cat........
272  0100186A 6176612F 696F2F46 696C654F  ...java/io/FileO
288  75747075 74537472 65616D08 00160100  utputStream.....
304  2B62652F 6C656572 73746164 2F736572  +be/leerstad/ser
320  69616C69 7A617469 6F6E2F66 696C6573  ialization/files
336  2F746573 74536572 2E736572 0A001300  /testSer.ser....
352  180C0005 00190100 15284C6A 6176612F  .........(Ljava/
368  6C616E67 2F537472 696E673B 29560700  lang/String;)V..
384  1B01001A 6A617661 2F696F2F 4F626A65  ....java/io/Obje
400  63744F75 74707574 53747265 616D0A00  ctOutputStream..
416  1A001D0C 0005001E 01001928 4C6A6176  ...........(Ljav
```

0 out of 1444 bytes

2. Bytecode verifier
   - No illegal bytecodes
   - No invalid register, stack and data type usage...
3. Interpreter
   - Reads bytecodes, translates them into a language that the computer can understand

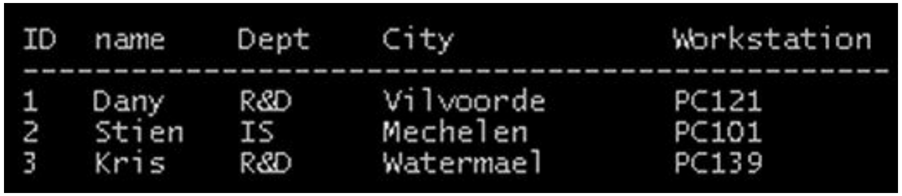
Java Virtual Machine

Advanced concepts:

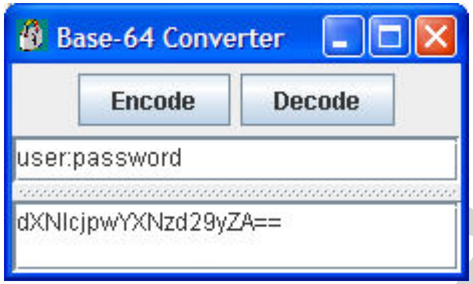
- Security Manager
  - prevents unauthorized operations (listening on network ports, reading data, ...)
- Garbage Collector
  - recycles memory used by past objects

Java Applications

- Type 1: Standalone applications
  - Program runs directly on the Java VM
  - Console or no console
  - Graphical User Interface or simple textual input/output
  - The main class is the one containing a special method called `main()`



```
ID   name     Dept     City          Workstation
---------------------------------------------------
1    Dany     R&D      Vilvoorde     PC121
2    Stien    IS       Mechelen      PC101
3    Kris     R&D      Watermael     PC139
```

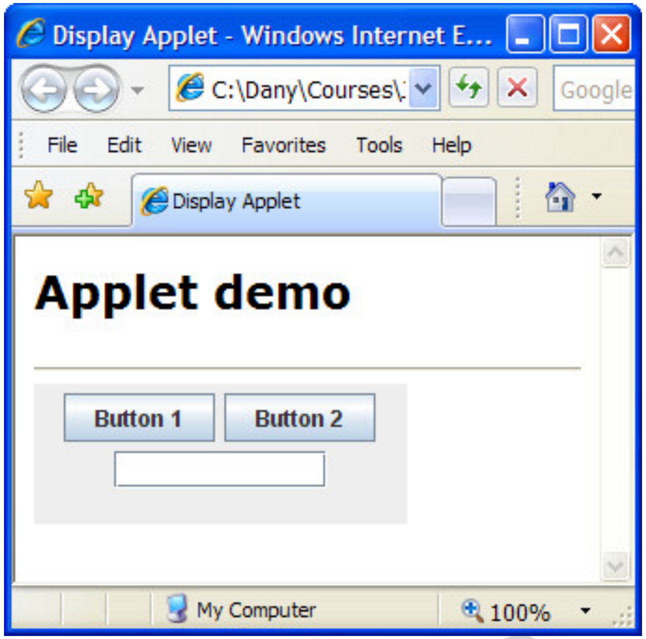


user:password

dXNlcjpwYXNzd29yZA==

Java Applications

- Type 2: Applets
  - Embedded into an HTML document
  - Needs a Web Browser to run
  - The main class extends `JApplet` and contains a special method called `init()`



Creating/Running a Java Program

- The main class

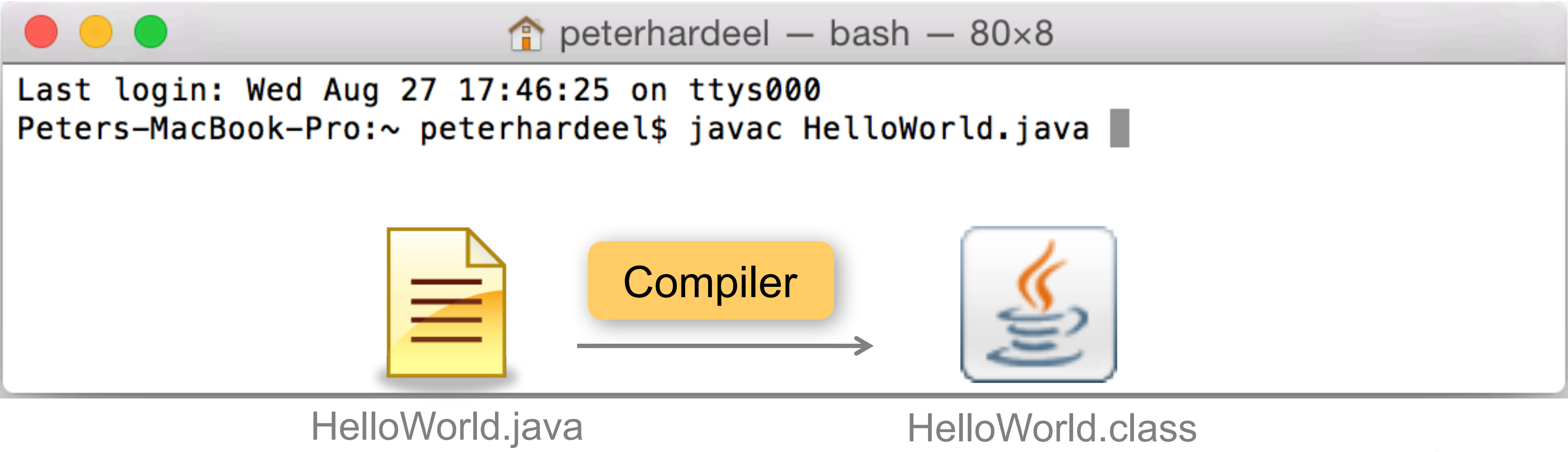
```
public class HellloWorld {

 public static void main(String[] args) {

   System.out.println("Hello World");
   LocalDate now = LocalDate.now();
   System.out.println("Today is : " + now);

   }
}
```

HellloWorld.java

- Filenames are always case sensitive!
- Source files are text files; they can be edited by any text editor (Notepad, Gedit, Emacs, Textedit, VI, ...).

Creating/Running a Java Program

- Compiling the source file with *javac.exe*

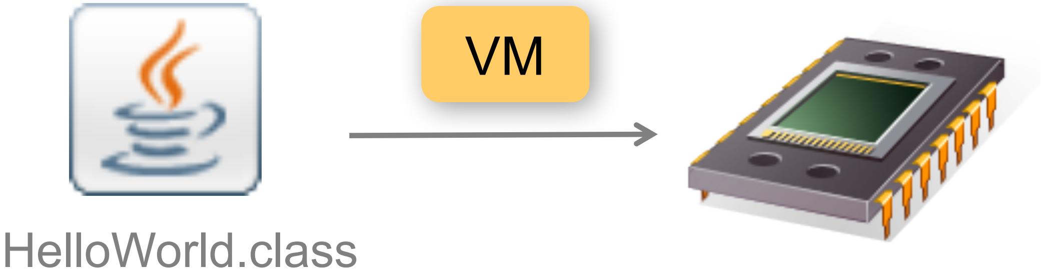


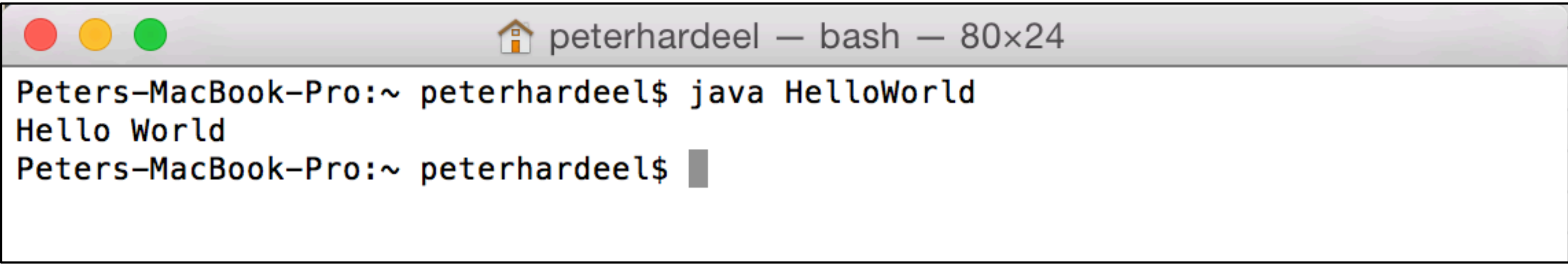
```
Last login: Wed Aug 27 17:46:25 on ttys000
Peters-MacBook-Pro:~ peterhardeel$ javac HelloWorld.java
```

- The directory of javac.exe must be in the PATH environment variable
- You can use wildcards: `javac Hello*.java`

Creating/Running a Java Program

- Running the program with *java.exe*



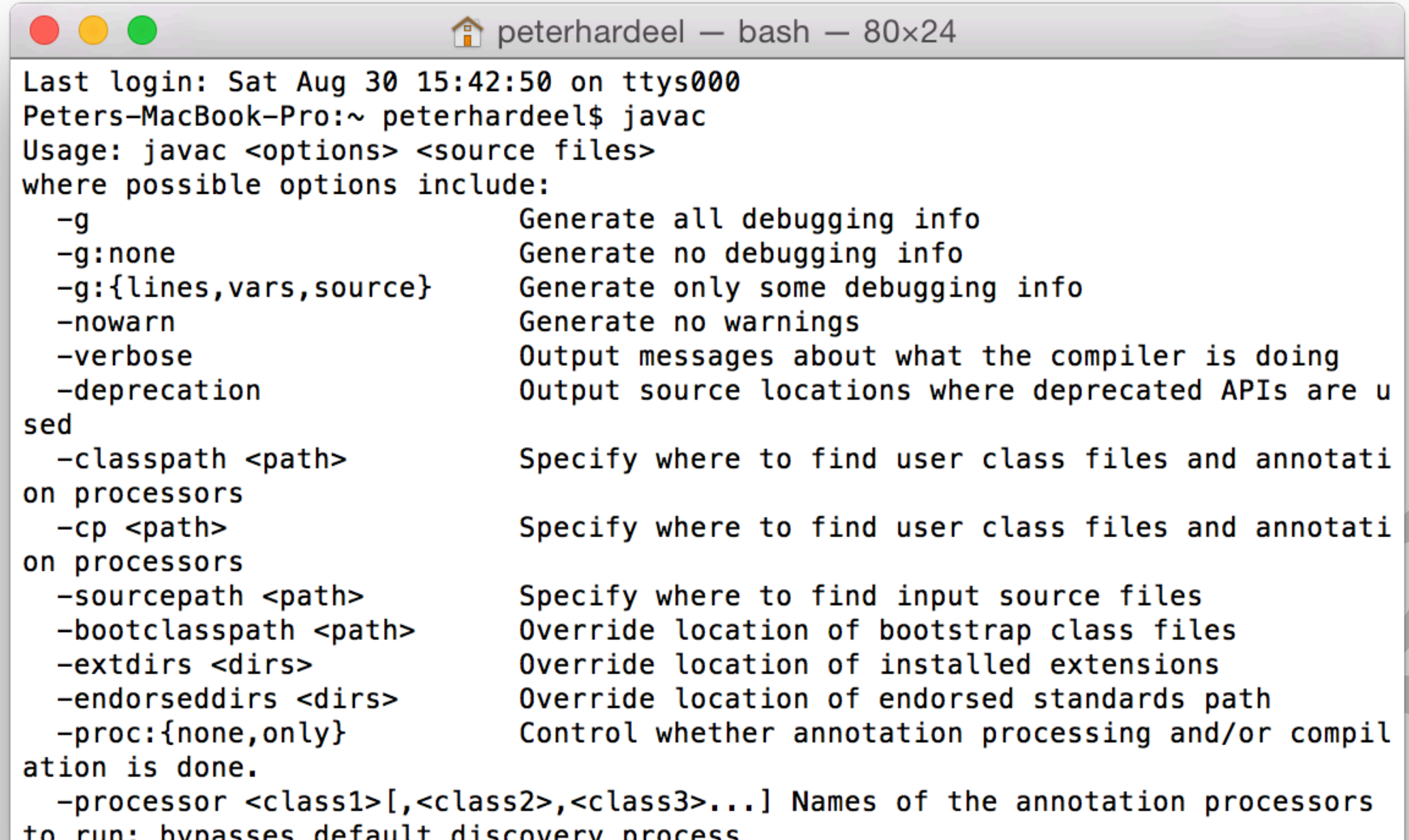


```
Peters-MacBook-Pro:~ peterhardeel$ java HelloWorld
Hello World
Peters-MacBook-Pro:~ peterhardeel$ 
```

- The directory of *java.exe* must be in the PATH environment variable
- The current directory (.) must be in the CLASSPATH environment variable
- Use the class name WITHOUT file extension!
- .class files are platform independant

Compiler Options

- Displaying options
  - Start *javac.exe* without parameters:



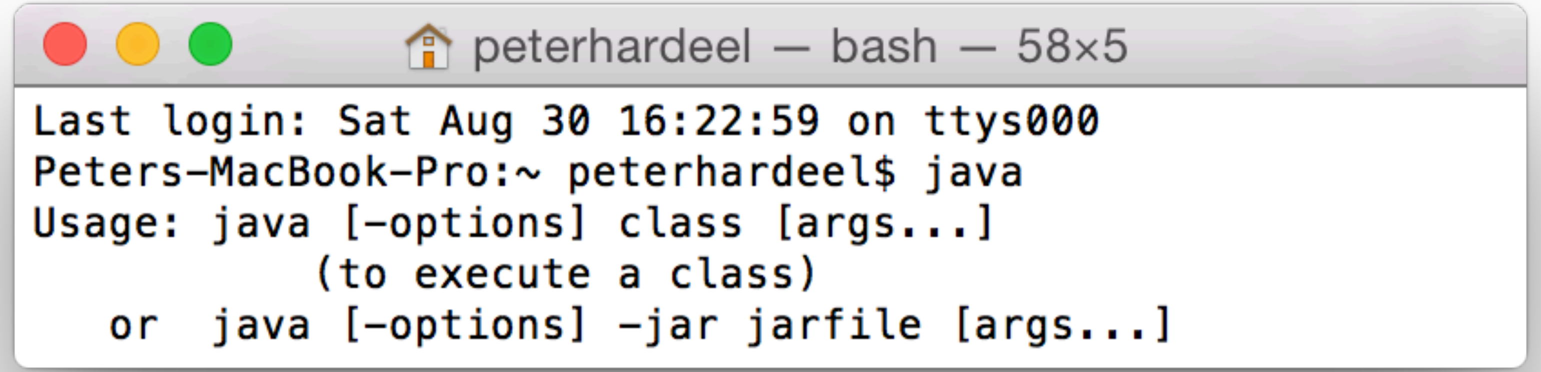
```
Last login: Sat Aug 30 15:42:50 on ttys000
Peters-MacBook-Pro:~ peterhardeel$ javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are u
sed
  -classpath <path>          Specify where to find user class files and annotati
on processors
  -cp <path>                 Specify where to find user class files and annotati
on processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compil
ation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors
to run; bypasses default discovery process
```

Compiler Options

- Frequently used options
  - Option `-cp` (or `-classpath`) indicates where the compiler should look for already compiled classes imported by the class to be compiled
  - Option `-d` specifies the directory where to put the *.class* files (and instructs the compiler to create the directory structure associated with the package structure of the class – see Objective 4)

Program Arguments

- Displaying syntax
  - Start *java.exe* without parameters

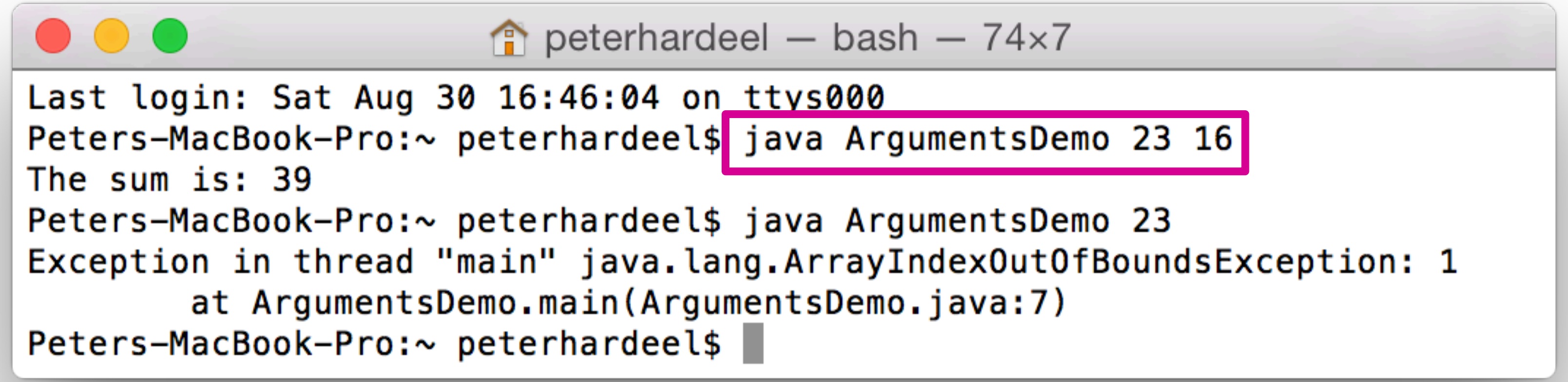


```
Last login: Sat Aug 30 16:22:59 on ttys000
Peters-MacBook-Pro:~ peterhardeel$ java
Usage: java [-options] class [args...]
           (to execute a class)
   or  java [-options] -jar jarfile [args...]
```

  - Arguments allows you to parametrize your program at start-up

Program Arguments

- Usage
  - Start *java.exe* with classname plus expected arguments:

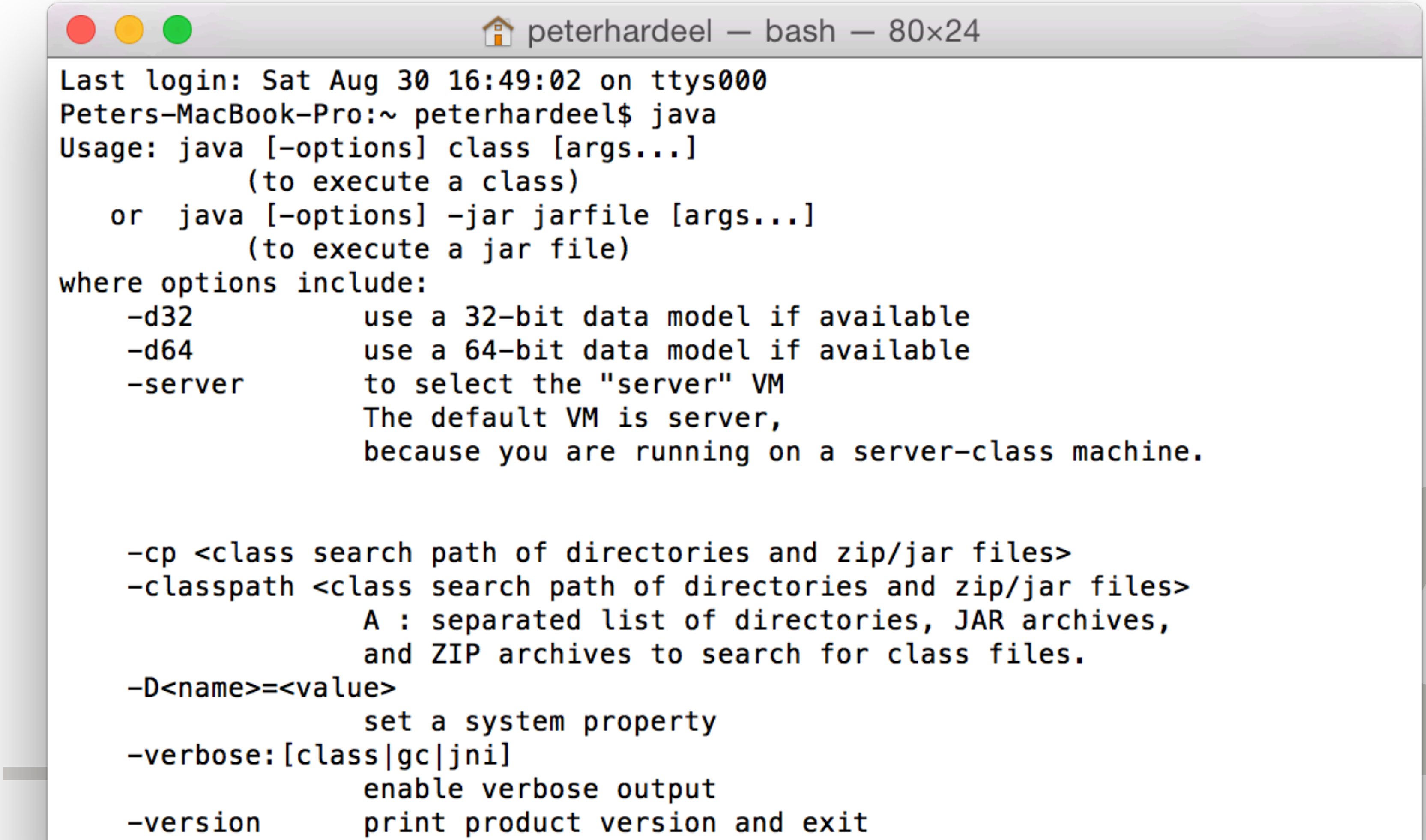


```
Last login: Sat Aug 30 16:46:04 on ttys000
Peters-MacBook-Pro:~ peterhardeel$ java ArgumentsDemo 23 16
The sum is: 39
Peters-MacBook-Pro:~ peterhardeel$ java ArgumentsDemo 23
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at ArgumentsDemo.main(ArgumentsDemo.java:7)
Peters-MacBook-Pro:~ peterhardeel$ 
```

  - You can program defensively if too few parameters are provided

JVM Parameters

- Displaying syntax
  - Start *java.exe* without parameters



```
Last login: Sat Aug 30 16:49:02 on ttys000
Peters-MacBook-Pro:~ peterhardeel$ java
Usage: java [-options] class [args...]
           (to execute a class)
   or  java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -server       to select the "server" VM
                  The default VM is server,
                  because you are running on a server-class machine.


    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A : separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D<name>=<value>
                  set a system property
    -verbose:[class|gc|jni]
                  enable verbose output
    -version      print product version and exit
```

JVM Parameters

- Frequently used option
  - Option `-cp` (or `-classpath`) indicates where the JVM should look for java classes (and libraries other than those from the Java API)

```
java -cp /Users/peterhardeel/code;
       /Users/peterhardeel/logging-log4j-1.2.14/dist/lib/log4j.jar
       ClassPathDemo

15:35:44,000  INFO ClassPathDemo:6 - *** Starting a demo ***
```

- We start the application from C:\
- We indicate the JVM:
  - where to find the main class
  - where to find a dependent library (a .jar file)

JVM Parameters

- Frequently used option
  - Option `-D` specifies system properties
    - necessary for the JVM in some circumstances (using a Security Manager, using a Web proxy, ...)
    - can also be read by the program

Java Virtual Machine(s)

- Oracle's JDK provides one or more implementations of the Java virtual machine (VM):
  - Java HotSpot Client VM is tuned for reducing start-up time and memory footprint
  - Java HotSpot Server VM (*server VM*) is designed for maximum program execution speed
- Key differences: adaptative interpreter (bottlenecks), memory allocation and garbage collection, thread synchronization

Java Developer's Kit

- The table summarizes some of the basic tools of the JDK:

program	description
javac.exe	The compiler for the Java programming language
java.exe	The launcher for Java applications
javadoc.exe	API documentation generator
appletviewer.exe	Run and debug applets without a web browser
javaw.exe	Identical to java.exe, except that with javaw.exe there is no associated console window
HtmlConverter.exe	Converts an HTML page (file) containing applets to the OBJECT / EMBED tag format for Java Plug-in
jar.exe	Create and manage Java Archive (JAR) files
keytool.exe	Manage keystores and certificates
rmic.exe	Generate stubs and skeletons for remote objects in distributed computing

Java Developer's Kit

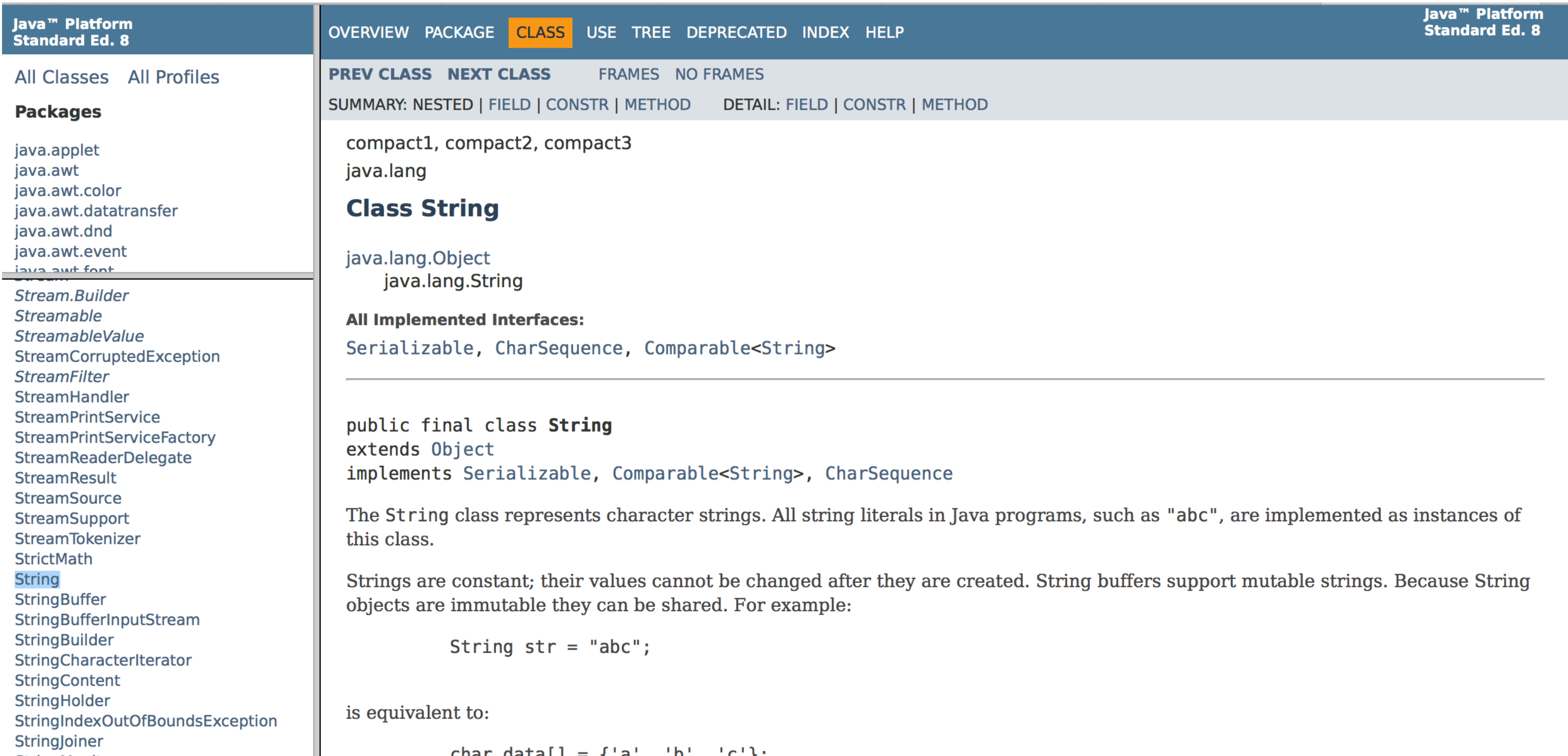
- Default libraries:
  - Regular expressions, collections, logging, reflection, ZIP files, ...
  - Input/output, math, networking, security, internationalization, XML, native interface, ...
  - Database connectivity, Remote Method Invocation (RMI), ...
  - Windowing toolkit, sound, print, drag&drop, ...

Java Developer's Kit

- Bonus:
  - Source code of the JDK API components
  - Demos

Java Developer's Kit

- JDK documentation:
  - Must be downloaded separately



compact1, compact2, compact3

java.lang

Class String

java.lang.Object
    java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

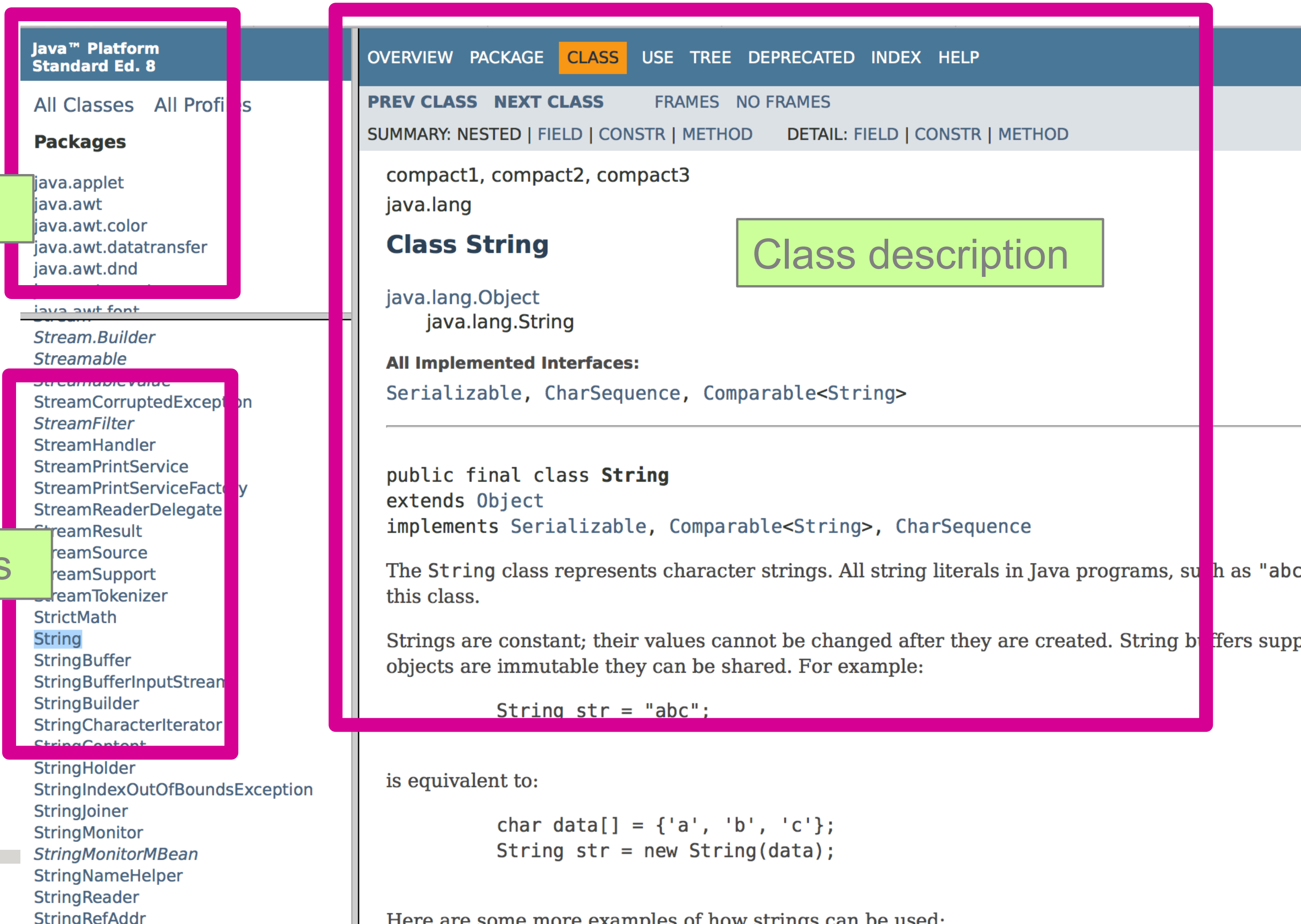
```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
```

Java Developer's Kit

- API documentation:



List of packages

List of classes

Class description

Java™ Platform Standard Ed. 8

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.font

Stream.Builder
Streamable
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamSupport
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder
StringCharacterIterator
StringContent
StringHolder
StringIndexOutOfBoundsException
StringJoiner
StringMonitor
StringMonitorMBean
StringNameHelper
StringReader
StringRefAddr

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.lang

Class String

java.lang.Object
 java.lang.String

All Implemented Interfaces:

`Serializable, CharSequence, Comparable<String>`

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as "abc
this class.

Strings are constant; their values cannot be changed after they are created. String buffers supp
objects are immutable they can be shared. For example:

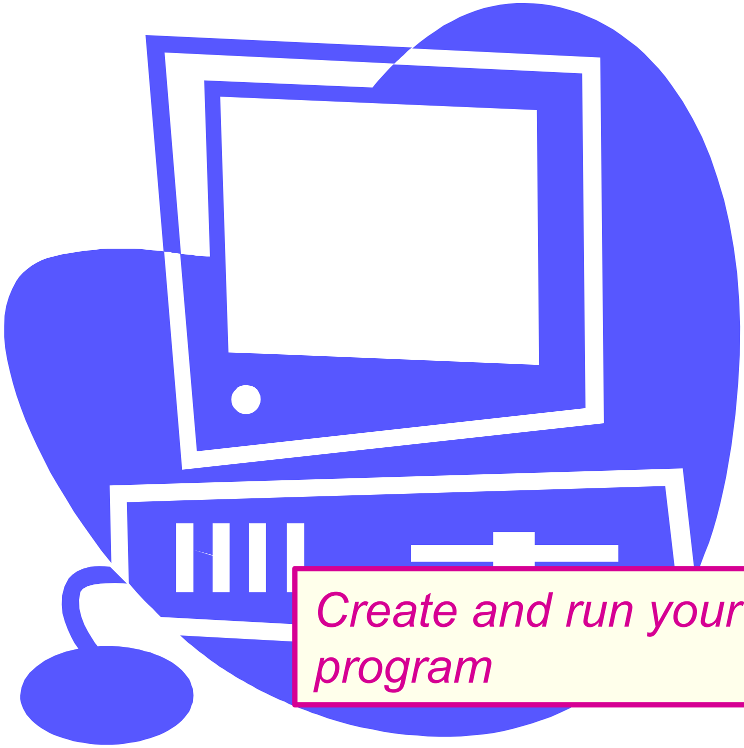
```
     String str = "abc";
```

is equivalent to:

```
     char data[] = {'a', 'b', 'c'};
     String str = new String(data);
```

Here are some more examples of how strings can be used:

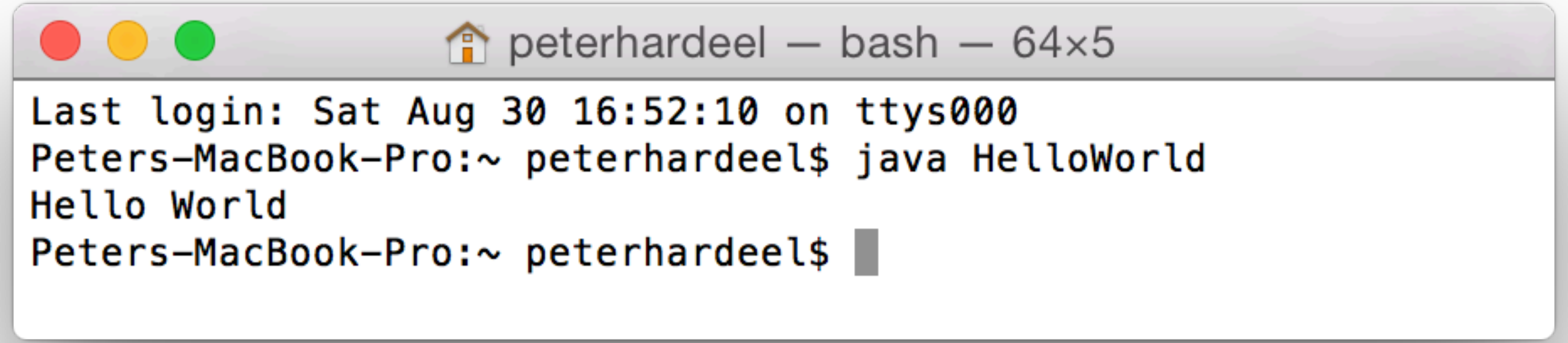
Exercise



Create and run your first Java program

Exercise

- Expected output:



```
Last login: Sat Aug 30 16:52:10 on ttys000
Peters-MacBook-Pro:~ peterhardeel$ java HelloWorld
Hello World
Peters-MacBook-Pro:~ peterhardeel$ 
```

- Key concepts:
  - Path and classpath environment variables
  - Text editor
  - Compiler, virtual machine
  - API documentation

Integrated Development Environments

- Provide comprehensive facilities to computer programmers for software development. They normally consist of:
  - Source code editor
  - Compiler and/or interpreter
  - Build automation tools
  - Debugger

Integrated Development Environments

- Examples
  - **Eclipse**-based IDEs (Eclipse EE, EasyEclipse, Eclipse IDE, Genuitech's MyEclipse, Borland's JBuilder, ...)
  - Sun Microsystem's **Netbeans**
  - IDEA's **IntelliJ**
  - Oracle's **JDevelopper**

Summary

- A program written in Java programming language is first translated into Java's intermediate language (compiling).
- The program is then executed on a Java virtual machine which interprets the intermediate language on some target computer (Windows, Linux, Mac, Unix, ...)
- Every platform has its own JVM