Java Technology

Portability

Security

Complete APIs support

What Is Java ?

- Java is much more than just a new programming language.
- Java provides a framework for applications development.
- Java provides code portability. Java executables could run on all operating systems.
- Java provides a very rich libraries support.
- Java increases both security and code integrity by its runtime mechanism.

Elements of The Java Platform

- Java programming language.
- Software libraries accompanying the system.
- Java Virtual Machine (interpreter).
- Compatibility to the Web.

The Java Programming Language

- Java is an easy to use pure object oriented language.
- Java has an overwhelming set of rich libraries that makes java a suitable language for a very large scale of programming fields.
- Java supplies an exception handling mechanism that improves bugs handling.

Java Portability

- All java code is cross platforms.
- The Java rich set of libraries classes are identical on all java implementation no matter which platform you use.
- A java program is compiled once on any OS and can run anywhere.
- The way this portability is achieved is explained later in this module.

Java Programming Language - Characteristics

- Syntax - similar to C++.
- Strongly typed.
- Pure object oriented.
- Size of primitives is laid down in the language and is platform **in**dependent.
- Has garbage collector.
- Multi threaded language.
- Support exceptions.

Java Programming Language – cont’d

- *“Within C++, there is a much smaller and cleaner language struggling to get out”* [Stroustrup, Bjjarne]
- Sometimes, it is not what that was added to the language but what was kept out…
- Java has left out many problematic design issues of C and C++.

Java does NOT have:

- Memory address (pointers) arithmetic.
- Preprocessor.
- Automatic type conversion.
- Global functions and variables.
- Typedefs.
- Operator overloading.
- Multiple inheritance.

Java Libraries

- Huge amount of cross-platform APIs.
- Easy to use APIs.
- GUI handling.
- Network handling.
- Database handling.
- I/O handling.

Java APIs

- Java runtime – Standard runtime libraries for I/O, networking, applets, basic windowing, data structures, internationalization, math operations and so on.
- JFC - Java Foundation Classes. The main support for graphics is done by the “Swing” package.
- Security – Digital signatures, certificates, message digests, etc.
- JDBC – Java Database Connectivity. Access to databases.
- JavaBeans – Software component library.
- Java RMI – Java Remote Method Invocation. Used for distributed programming and invoking methods of other processes, possibly on other systems.

Java APIs – cont'd

- Java Communications – Support parallel I/O . Enables faxing, voice mail, smart cards, etc.
- JavaMail – Protocol independent framework for building java based mail and messaging applications.
- Java media - Packages for 3D imaging, MIDI sound, telephony, video and audio streaming.
- JNDI – Java Naming and Directory Interface. A standard to get information from LDAP, NIS or other enterprise directory service.

Write Once – Run Everywhere

- A java program may be compiled on any computer that has a JVM installed.
- A java executable is a binary file that runs on every processor that has a JVM installed.
- Recompilation is never required when replacing a platform.

Java Virtual Machine

- A Java compiler creates *bytecode* instead of native binary code.
- *Bytecode* is an architecture-neutral machine code.
- *Bytecode* can quickly be interpreted to run on any specific computer.
- The JVM translates *bytecode* into native instruction set.
- A java *bytecode* written on one platform can be translated into any native instruction set of any possible platform that has a JVM installed on it.
- The JVM is actually an interpreter that interprets java *bytecode* to native code.

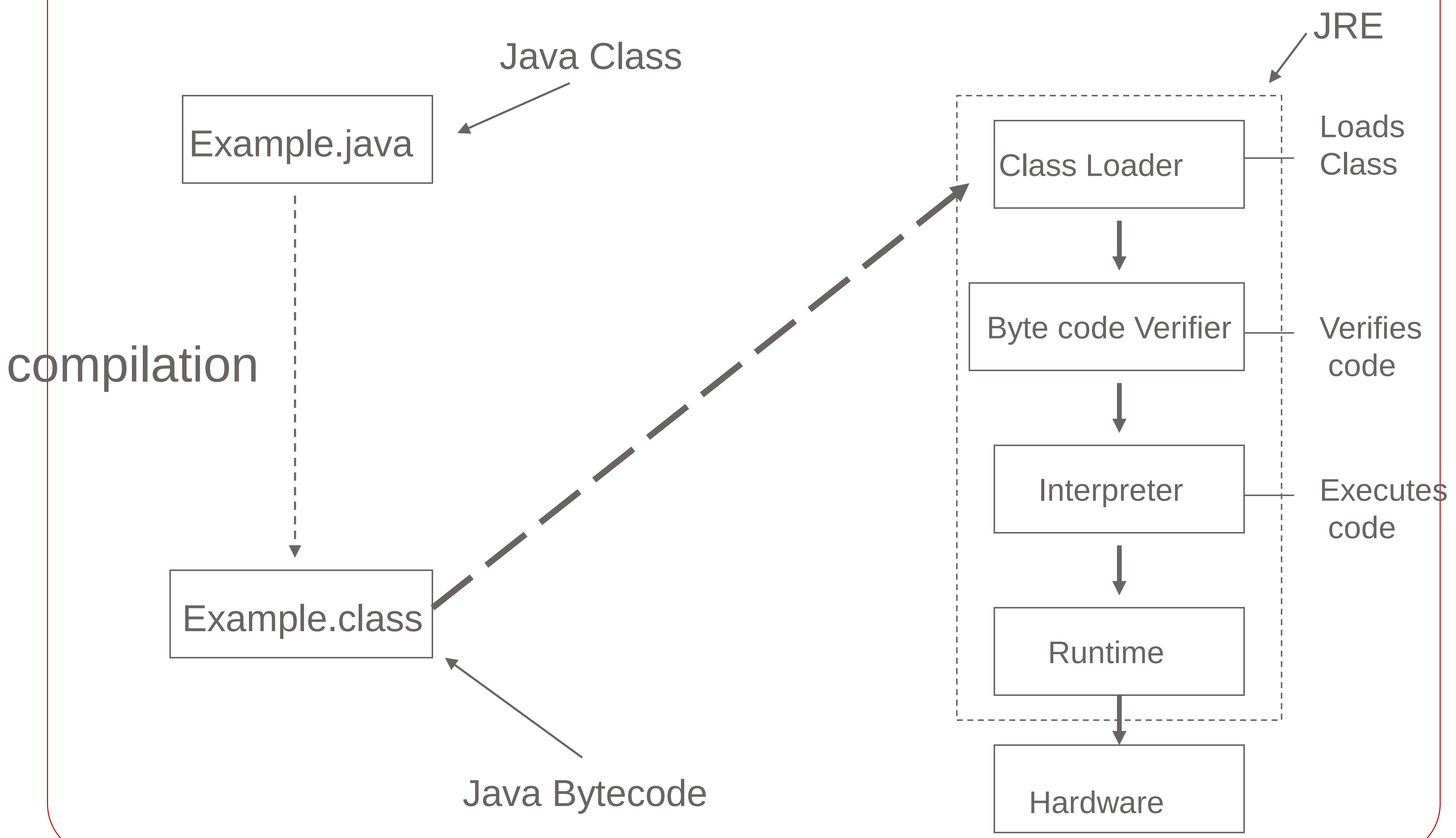
Java Runtime Environment

- Java source files are compiled into bytecode.
- The bytecode is stored in a *.class* file.
- At runtime, the bytecode is loaded, verified and ran by the interpreter.

Java Runtime Environment

- Is what needed to run Java programs
- Free download from www.java.sun.com
- Can be installed as a browser plug-in
- Contains:
  - JVM
  - Runtime classes (Matching the specific OS)
  - Java Application Launcher

Java Runtime Environment



Java Runtime Environment – cont’d

- A Java public class will be written in a file named after the class and has a “*.java”* extension.

  e.g. *String.java*.
- The java compiler (in the JDK, invoked by “*javac className.java”*) compiles the Java source code into bytecode. This will result in a file that has the class name and the extension *“.class”*.
- The Java Runtime Environment loads the class, verifies that its bytecode is compliment to the JVM specifications and then runs the bytecode.

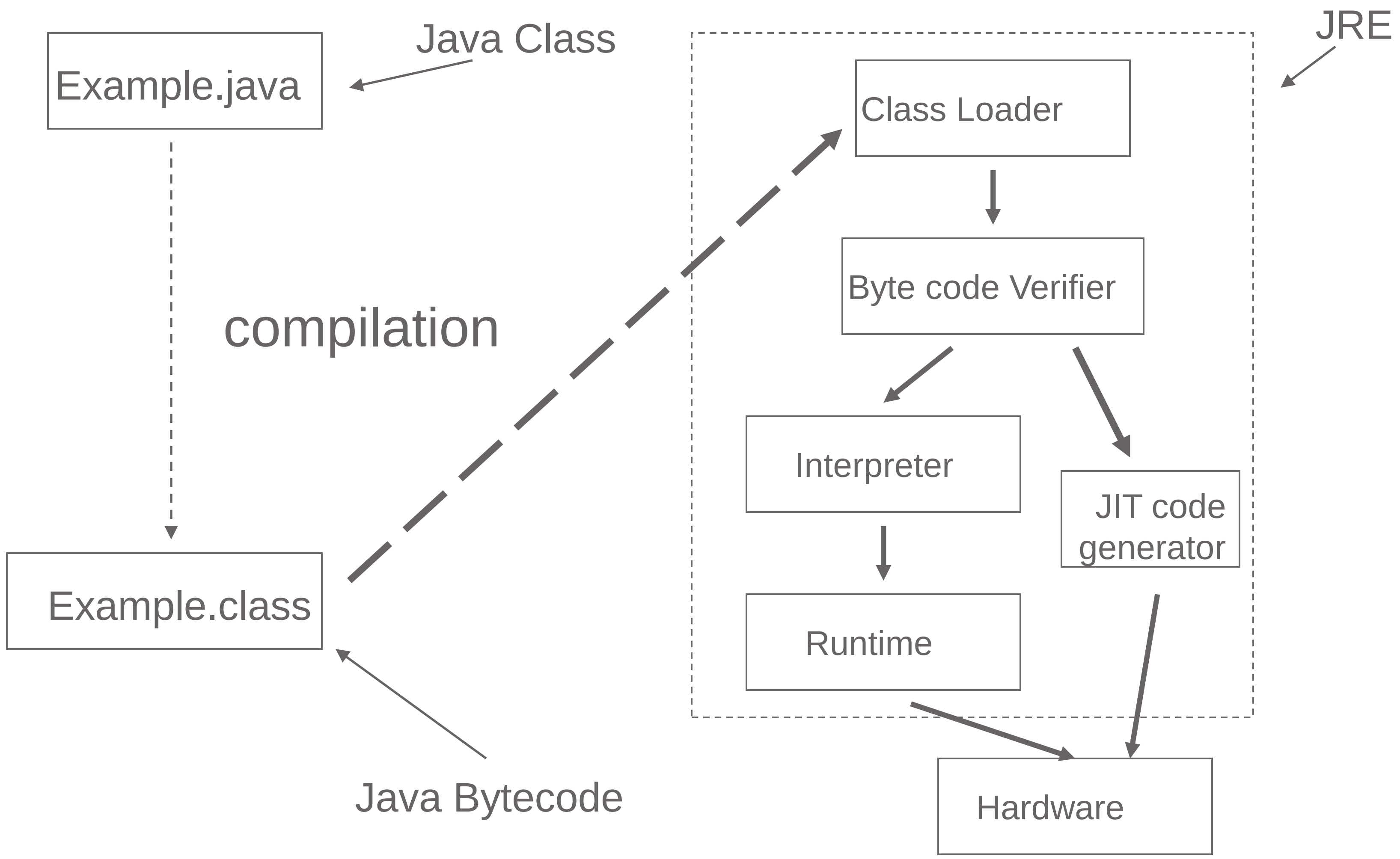
Java Runtime Environment Elements

- Class Loader - Loads all classes which are necessary for the execution of a program.
- Bytecode Verifier – verifies that class bytecode is legal and does not violate system integrity.
- Interpreter/JIT Compiler – Executes the code.

Just In Time Compiler

- If binary code is not compiled down to a specific machine code, extra work is needed at runtime to finish the job.
- The first java programs were about 10 to 20 times slower then equivalent C++ code (today differences are much smaller).
- Some JVMs can compile the bytecode as they run it.
- In the first run, the code is run in interpreted speed but it is also compiled in a separate thread to native machine code.
- From now on the binary native code is executed and the runtime extra work is saved.
- The technique is called “Just In Time” compilation.
- Some compilers translate directly to native code (executables files).

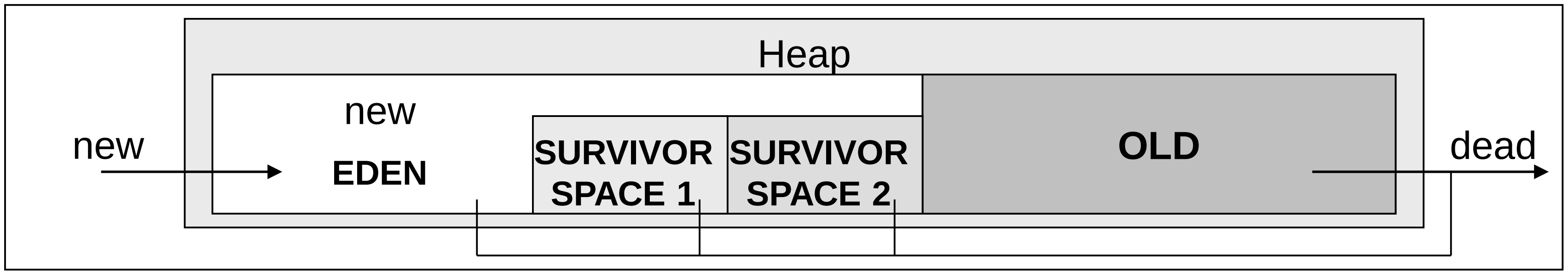
JIT Compiler-cont'd



Hot-Spot

- Java is a tool that:
- Improves caching of objects & methods in a distributed environments
- Has a fast and fully accurate Garbage Collection
- Offers more GC algorithms (see next slide)
- Has method inlining
- Has compiler that does not use the interpreter – much faster
- Has an advanced logging when handling native crashes
- Has smart thread queue management (all gets about 5% CPU time)
- Has fast thread synchronization
- Produces statistics between program executions to improve performance
- Supports new I/O
    - Better buffering management
    - Scalable network (not one thread per connection)
    - Character-set support

- Java Hotspot
    - Fast and fully accurate garbage collection (per generation cleanup)



Portability in Java

- Portability in Java is even more than just the use of bytecode. Portability is also achieved by the following Java language characteristics:
  - Unlike C/C++, Java operands are evaluated in a strict left-to-right order.
  - Primitives have definite size which make them platform **in**dependent.
  - Java introduces the same API on all platforms.
  - Java uses the Unicode character set, which is a 16-bit superset of ASCII and therefore could represent most alphabets symbols in the world.

Web Compatibility

- Applets - An application program that runs inside a browser.
- Servlets - An application program that runs on demand by a web server.

Memory Management

- In Java, objects are created on the heap using the *new* keyword.
- The heap tends to be fragmented from time to time.
- When the heap is too fragmented and not enough continues heap memory is available (although free memory exist in the heap) the heap should be enlarged.
- On systems that use virtual memory, this will result in extra paging which has its immediate affect on performance.

Garbage Collection

- In java, memory management is not in the hands of the programmer.
- A special garbage collector thread is responsible for memory management.
- The garbage collector frees all dynamically allocated memory that is no longer referenced.
- An object located on the heap that is no longer referenced should be regarded as “garbage” and has to be removed.
- Garbage collection is actually memory recycling.
- The garbage collector must somehow determine which objects are no longer referenced and free the heap space they occupy.
- In the process of freeing unreferenced objects, the garbage collector must run any finalizers of objects being freed before the memory is recycled.

Garbage Collection – cont’d

It is important to understand that the garbage collector thread will not evacuate any unreferenced object the moment this object become unreferenced. In fact the object may still reside at the heap until the completion of the entire run of the program.

The garbage collector thread will only start executes when there is not enough continues free memory on the heap. This might not happen in the entire run of a program and there for a java developer can not assume any thing regarding the timing of objects recycling.

This issue creates a major difference between Java finalizers and C++ destructors. The outcome of this difference will be reviewed later in this module.

The Pros of Having GC

- It reduces the memory management headache from the programmer and consequently reduces the amount of time devoted for development. Programmers can now go home earlier and have a life…
- It helps insure program integrity. In fact this is part of the Java security policy. Programmers are unable to accidentally (or purposely) crash the JVM by incorrectly freeing memory.

The Cons of Having GC

- It adds an overhead that could hurt performances.
- Programmers have less control over the scheduling of objects disposal and finalizing.
- How could these problems be coped?
- Use very affective garbage collector algorithms that will hardly hurt performances.
- Java programmers, should avoid writing code for which program correctness depends upon the timely finalization of objects.

Programming With Garbage Collectors

Suppose an object is using a resource that is required by another object. When the first object is done and becomes unreferenced, the programmer might be tempted to claim the resource for the second object, believing that the first object was finalized and has released the resource.

The first object might be laying safely at the heap and still holding the resource that the second object is now trying to use. This might result in hurting program integrity.

It is crucial that program correctness will not depend on the timing of objects recycling.

How Does the Garbage Collector Work?

- There is NO standard GC implementation.
- The JVM specification says only that the heap of the Java virtual machine must be garbage collected, not how should it be done.
- One approach is to use references counting algorithms. Disadvantages are:
  - It does not detect cycles.
  - The overhead of incrementing and decrementing the reference count each time.
  - The memory space required for each object reference counter.

hi-tech college

JOHN BRYCE
Leading in IT Education
a matrix company

How Does the Garbage Collector Work ? – cont’d

- Another method is tracing algorithms. Tracing have two phases:
  - Create a graph of all objects and mark the referenced objects.
  - Free all unmarked objects.

The Java Development Kit

- The JDK is the software and tools required to compile, debug and run applications written in Java.
- The major releases are:
  - JDK 1.0.2 – May 1996.
  - JDK 1.1 – February 1997.
  - JDK 1.2 – December 1998. Its main contribution is the improved GUI support achieved by introducing the “*swing*” package and the support for CORBA.
  - Java 2 SDK v. 1.3 – 1.4 – Software Development Kit.
  - Java 2 SDK – v. 5.0 – Tiger – Syntax enhancements
  - Java 2 SDK – v. 6.0 – Mustang – More APIs & utilities
  - Java 2 SDK – v 7.0 – Dolphin - ?

The Java Development Kit

- Full kit for developing java applications
- Contains:
  - J2SE
  - JVM
  - Compiler, RMI-Compiler, IDL converter (bin directory)
  - API Documentation generating tool
  - Code examples
  - J2SE code sources
- Is extensible

Environment Variables

- PATH
  - OS variable to determine where to run the script from
  - Should point to the bin directory:

```
path=c:\jdk1.4\bin
```

- CLASSPATH
  - JVM variable used to locate the compiled and executed classes
  - Should point to the same directory or to added classes

- JAVA_HOME
  - Java Servers variable, specify the Java main directory location
  - Should point to the JDK directory:

```
java_home=c:\jdk1.4
```