Unit 1: Java

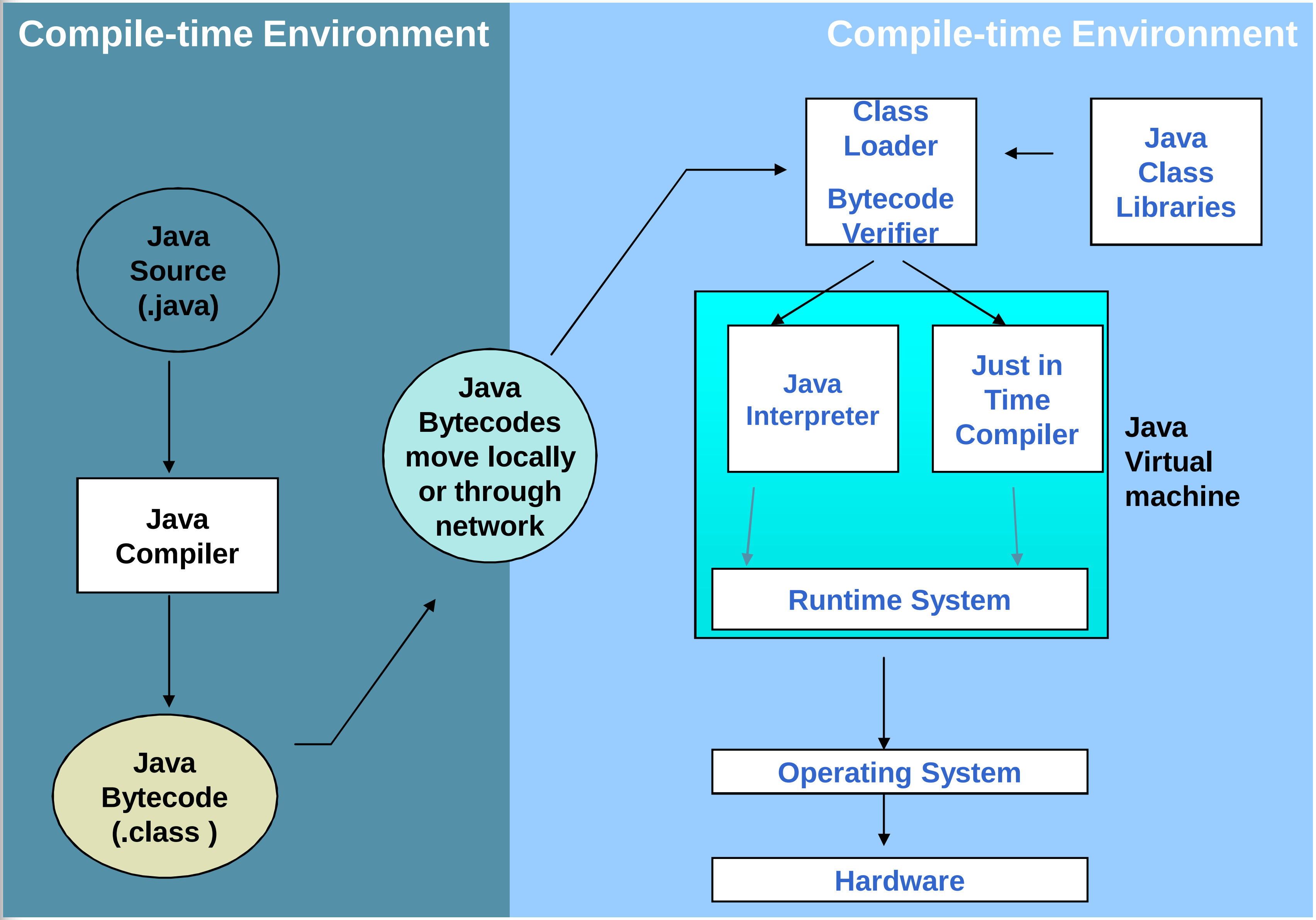
Java - Introduction

- Java is:
  - platform independent programming language
  - similar to C++ in syntax
  - similar to Smalltalk in mental paradigm
- Pros: also ubiquitous to net
- Cons: interpreted, and still under development (moving target)

Java - Application

- Java has some interesting features:
  - automatic type checking,
  - automatic garbage collection,
  - simplifies pointers; no directly accessible pointer to memory,
  - simplified network access,
  - multi-threading!

How it works…!


Compile-time Environment
Compile-time Environment
Java Source (.java)
Java Compiler
Java Bytecode (.class)
Java Bytecodes move locally or through network
Class Loader
Bytecode Verifier
Java Class Libraries
Java Interpreter
Just in Time Compiler
Runtime System
Java Virtual machine
Operating System
Hardware

How it works…!

- Java is independent only for one reason:
  - Only depends on the Java Virtual Machine (JVM),
  - code is compiled to *bytecode*, which is interpreted by the resident JVM,
  - JIT (just in time) compilers attempt to increase speed.

Java - Security

- Pointer denial - reduces chances of virulent programs corrupting host,
- Applets even more restricted -
  - May not
    - run local executables,
    - Read or write to local file system,
    - Communicate with any server other than the originating server.

Object-Oriented

- Java supports OOD
  - Polymorphism
  - Inheritance
  - Encapsulation
- Java programs contain nothing but definitions and instantiations of classes
  - Everything is encapsulated in a class!

Java Advantages

- Portable - Write Once, Run Anywhere
- Security has been well thought through
- Robust memory management
- Designed for network programming
- Multi-threaded (multiple simultaneous tasks)
- Dynamic & extensible (loads of libraries)
  - Classes stored in separate files
  - Loaded only when needed

Basic Java Syntax

Primitive Types and Variables

- boolean, char, byte, short, int, long, float, double etc.
- These basic (or primitive) types are the only types that are not objects (due to performance issues).
- This means that you don’t use the new operator to create a primitive variable.
- Declaring primitive variables:

```
float initVal;
int retVal, index = 2;
double gamma = 1.2, brightness
boolean valueOk = false;
```

Initialisation

- If no value is assigned prior to use, then the compiler will give an error
- Java sets primitive variables to zero or false in the case of a boolean variable
- All object references are initially set to null
- An array of anything is an object
  - Set to null on declaration
  - Elements to zero false or null on creation

Declarations

```
int index = 1.2;                   // compiler error
boolean retOk = 1;                 // compiler error
double fiveFourths = 5 / 4;   // no error!
float ratio = 5.8f;                // correct
double fiveFourths = 5.0 / 4.0;        // correct
```

- 1.2f is a float value accurate to 7 decimal places.
- 1.2 is a double value accurate to 15 decimal places.