CS 349 - User Interfaces **Winter 2018**

Lecture 6: January 15, 2018

Lecturer: Keiko Katsuragawa *Notes By: Harsh Mistry*

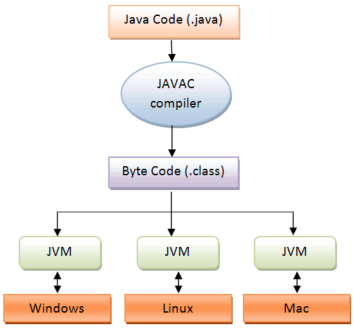
6.1 Java Basics

6.1.1 Background

- Designed by James Gosling
- Released by Sun in 1995
- Made open source in 2007
- Acquired by Oracle in 2010
- Portable through virtualization
- Class-based, object-oriented design

6.1.2 Java Portability through Virtualization

- Java compiles to bytecode (.class file)
- Bytecode is executed by a Java Virtual Machine (JVM)
- Just-in-Time (JIT) bytecode compilation can give near-native performance.



Taken from Lecture Notes

6.1.3 Java Class Library

- Classes are grouped into "packages"
- `package` keyword to assign source to a package
- Typically, a package is a subdirectory
- `import` keyword to include a class from a different package

6.1.4 Common Classes/Packages

Package	Classes (Examples)	Description
java.awt	Color, Graphics, Font, Graphics2D, event.	Contains all of the classes for creating user interfaces and for painting graphics and images.
javax.swing	JFrame, JButton, JList, JToolbar	Provides a set of "lightweight" (all-Java language) components that works the same on all platforms.
java.io	File, FileReader, FileWriter, InputStream	Provides for system input and output through data streams, serialization and the file system.
java.lang	Boolean, Integer, String, System, Thread, Math	Provides classes that are fundamental to the design of the Java programming language.
java.util	ArrayList, HashMap, Observable	Contains the collections framework, legacy collection classes, event model,...

Taken from Lecture Notes

6.1.5 Java Class Hierarchy

- All classes (implicitly) derive from Object class (in java.lang) and has methods like clone(), toString(), finalize()
- Classes you write inherit these basic behaviours

Taken from Lecture Notes

6.1.6 Instantiating Objects

- Primitive types (int, float, etc) are located on the stack and always passed by value

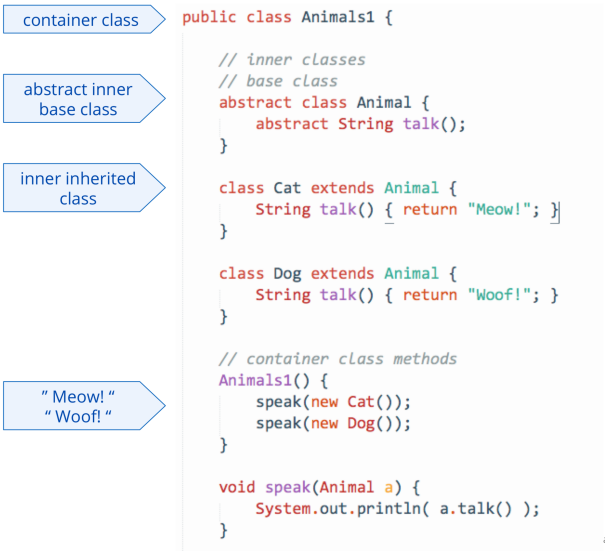
- Objects are allocated on the heap and can be thought of as always passed by reference, but in truth object address is passed by value

- There are no "pointer semantics" in Java

6.1.7 Inheritance

- Inherit some methods or fields from a base class ("is a")

- Its very common in Java to inherit and override other classes



Taken from Lecture Notes

6.1.8 Interfaces

- An `interface` represents a set of methods a class must have. Its basically a pure abstract class

- A class `implements` "all" methods in the interface

- A class can implement multiple interfaces

- Interfaces are often used to enforce an API, not functionality

interface

implementations

The interface Pet is like a type

```
// interface
interface Pet {
    String talk();
}

// inner class
class Cat implements Pet {
    public String talk() { return "Meow!"; }
}

class Dog implements Pet {
    public String talk() { return "Woof!"; }
}

void speak(Pet a) {
    System.out.println( a.talk() );
}
```

Java Basics 17

base class

interface

derived class

```
// base class
abstract class Bike {
    int wheels = 0;
    int speed = 0;

    void setWheels(int val) { wheels = val; }
    void setSpeed(int val) { speed = val; }
    void show() {
        System.out.println("wheels  = " + wheels);
        System.out.println("speed  = " + speed);
    }
}

// interface for ANYTHING driveable
// could be applied to car, scooter etc.
interface Driveable {
    void accelerate();
    void brake();
}

// derived two-wheel bike
class Bicycle extends Bike implements Driveable {
```

Taken from Lecture Notes