B15

The Java Programming Language

Christopher Holtz

IMS Technical Conference

Miami Beach, FL October 22-25, 2001

RETURN TO INDEX

Outline

- Object-Oriented Programming
- Java Programming
- Java Runtime
- J2EE / JDBC
- Java's Future in IMS

What is Java?

Sun Defines Java as:

A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language

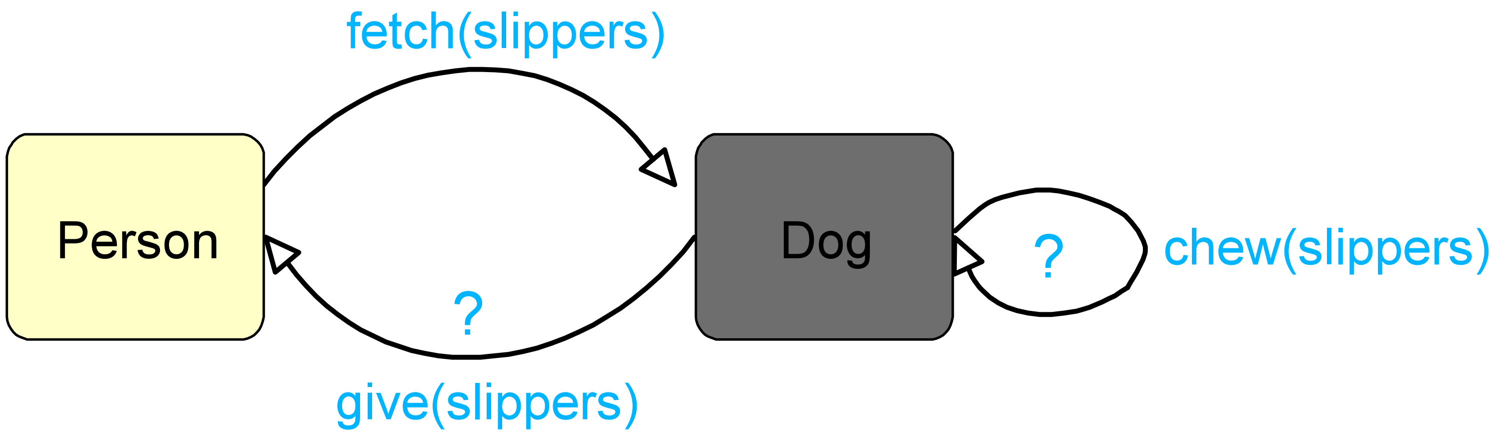
- **Object Oriented**
- **Platform Independent**
  - virtual machine
- **Rapid Application Development (RAD)**
  - no pointers
  - no storage managment
  - type strong

Object Oriented Programming

- **A type of programming that bundles data structures with functions to create re-useable objects.**
- **The three main principles of O-O design are:**
  - Encapsulation
  - Inheritance
  - Polymorphism

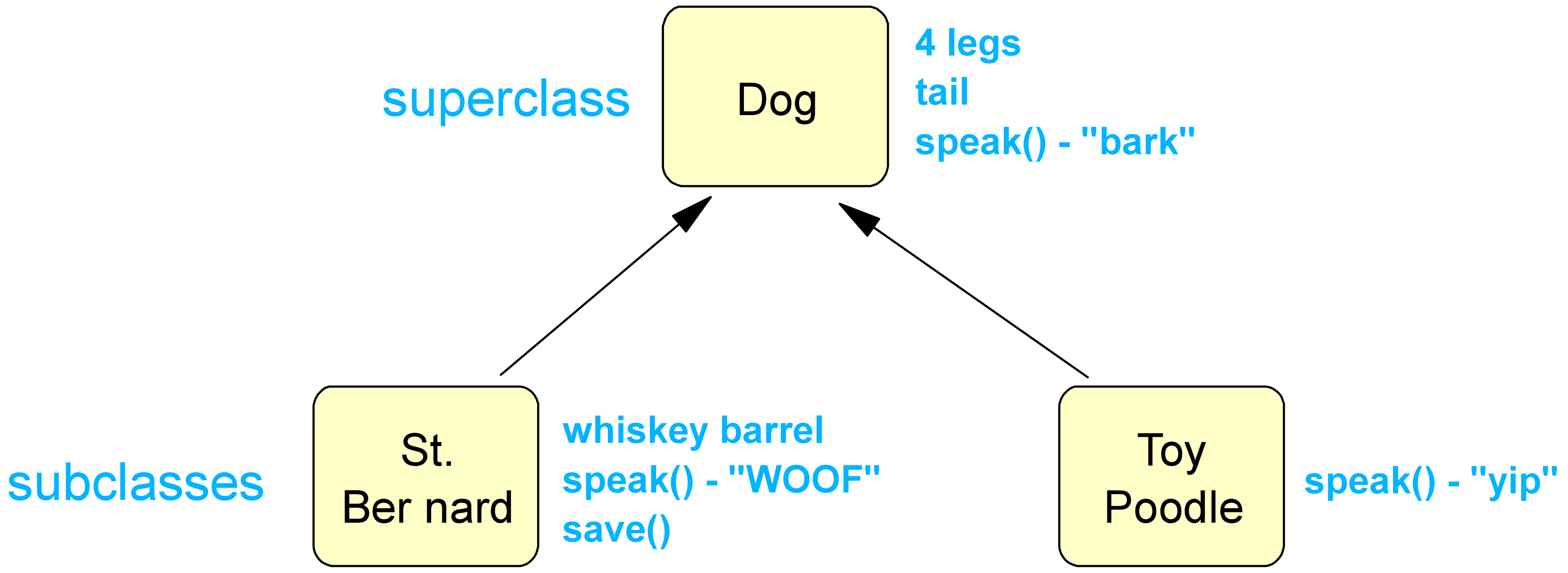
Encapsulation

- ***Encapsulation*** **or** ***Information Hiding*** **is the concealing of the implementation details of a data object from the outside world.**



Inheritance

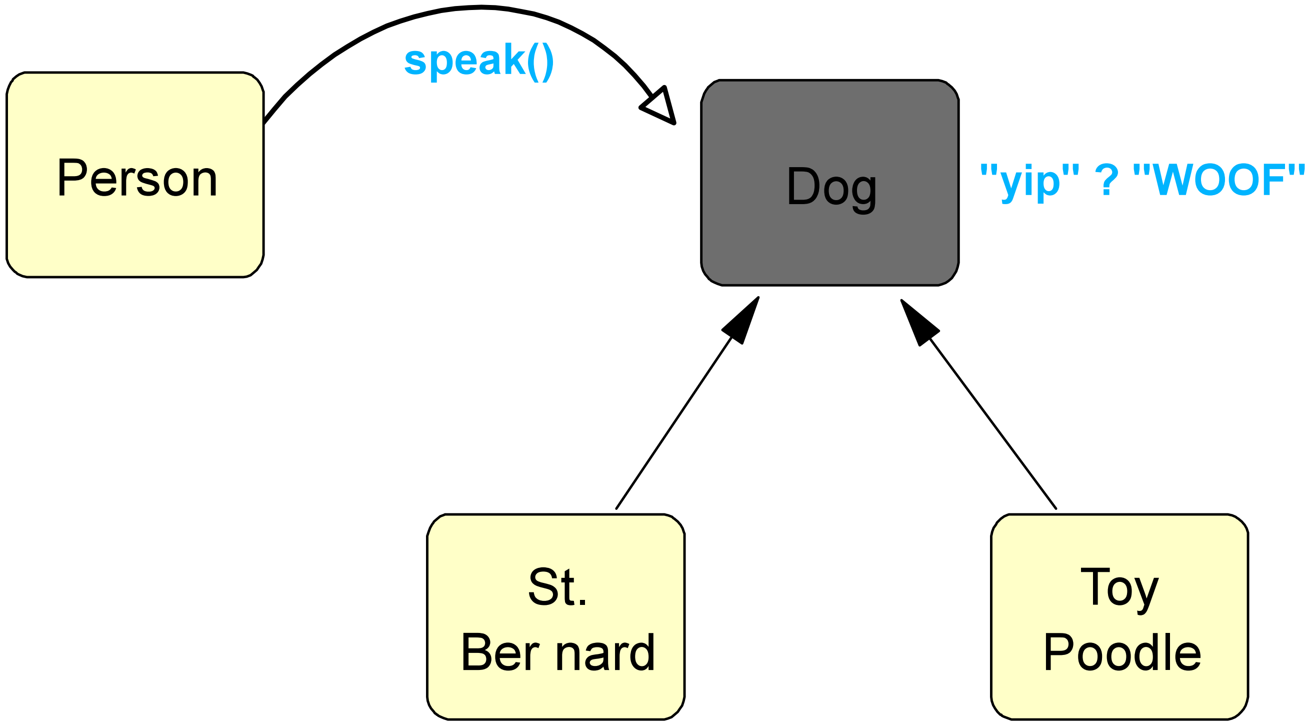
- Provides a means for a class to inherit variables and methods from its superclass, and all of its ancestors
- A subclass can then leave these members as is, *add* variables and methods or *override* the methods



IBM Software

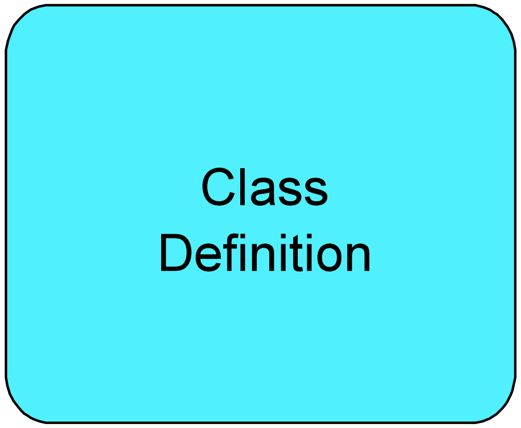
Polymorphism

- "*The ability to take many forms*"
- Ability to process objects differently depending on their data type or class unbeknownst to the caller



Classes

- **A blueprint or prototype that defines the variables and the methods common to all the objects of a certain kind.**

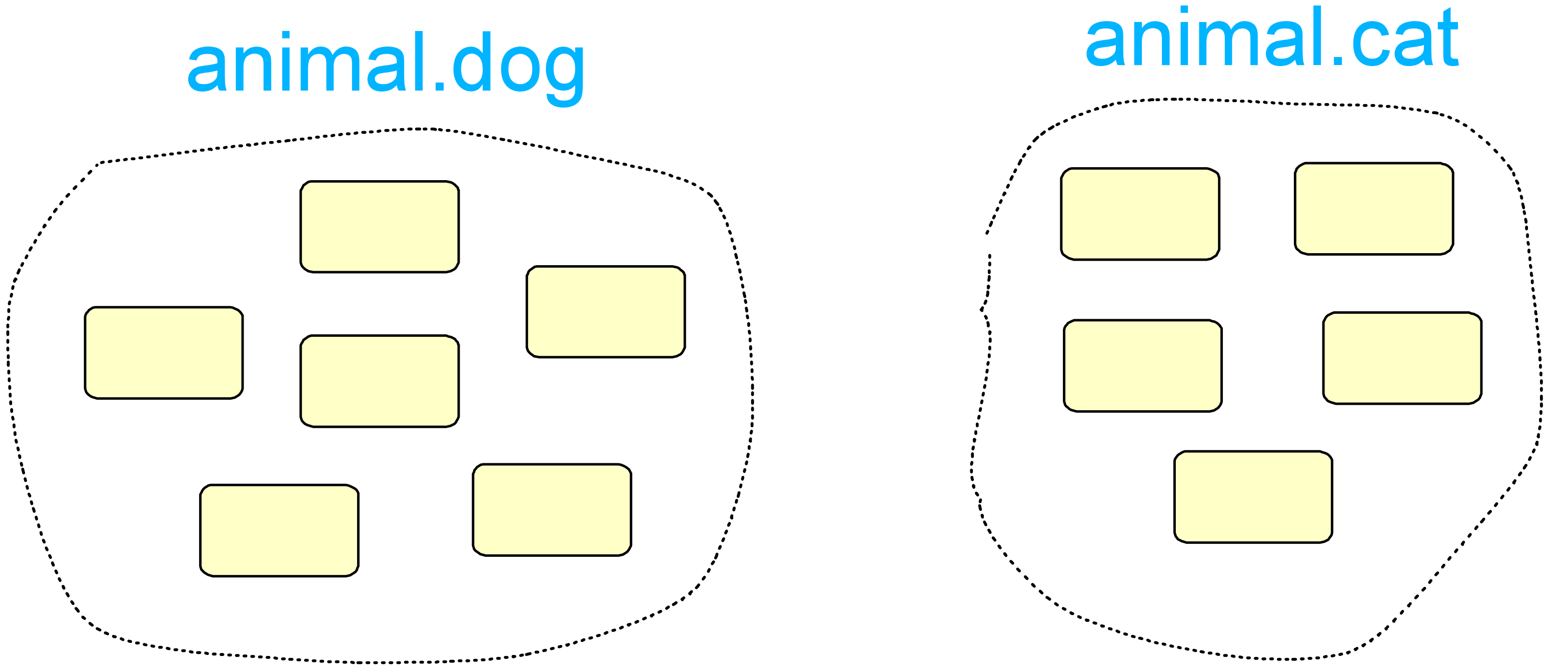


```
public class Dog {

    Leg[] legs;
    Tail tail;

    public void speak() {
        makeSound("bark");
    }
}
```

IBM Software

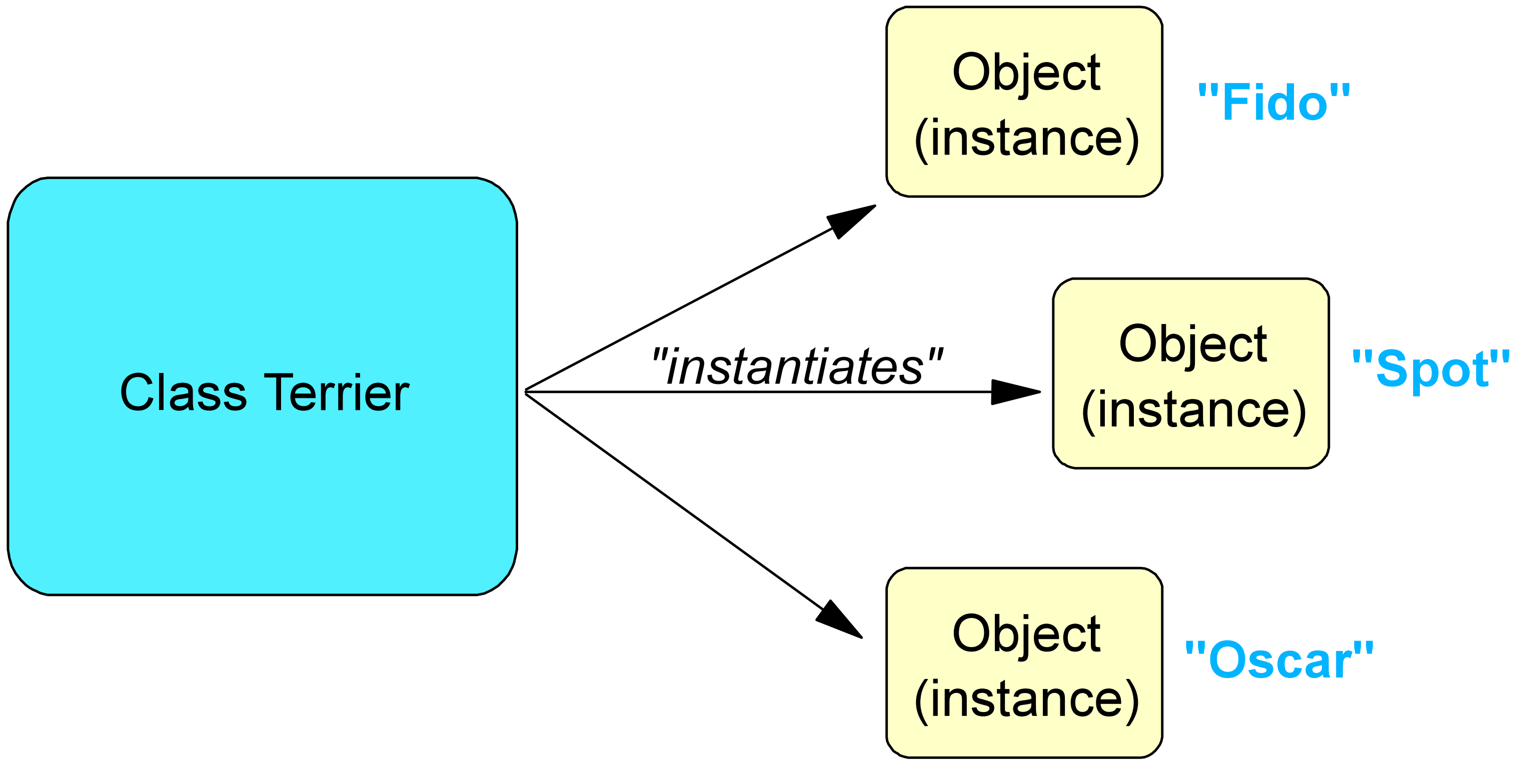
Packaging

- **Provides a way to bundle related classes into logically distinct groups**



Objects

- **An instance of a class**

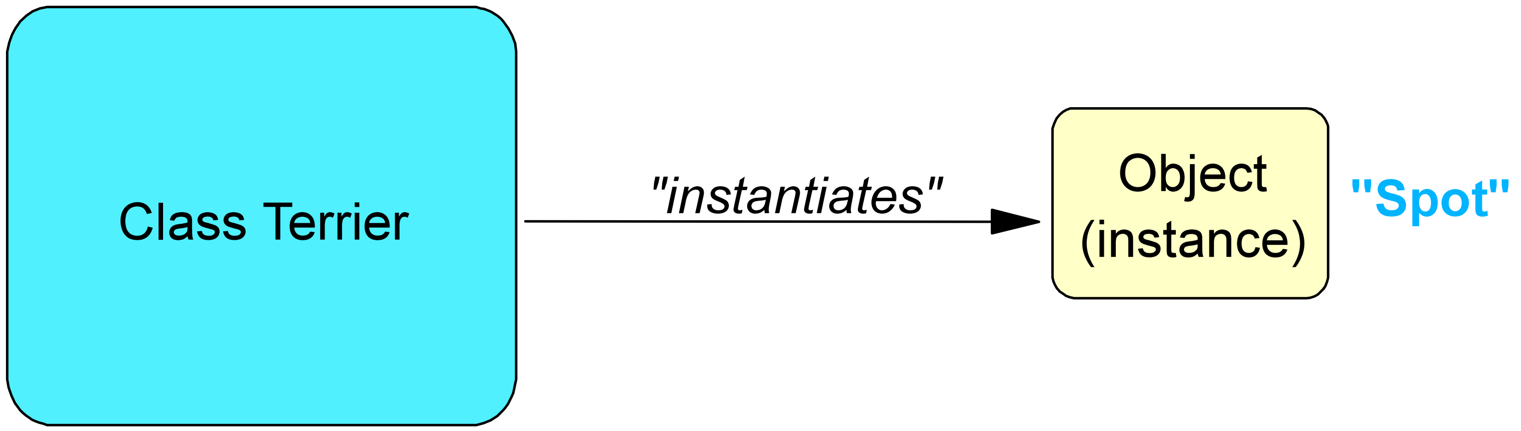


IBM Software

Constructors

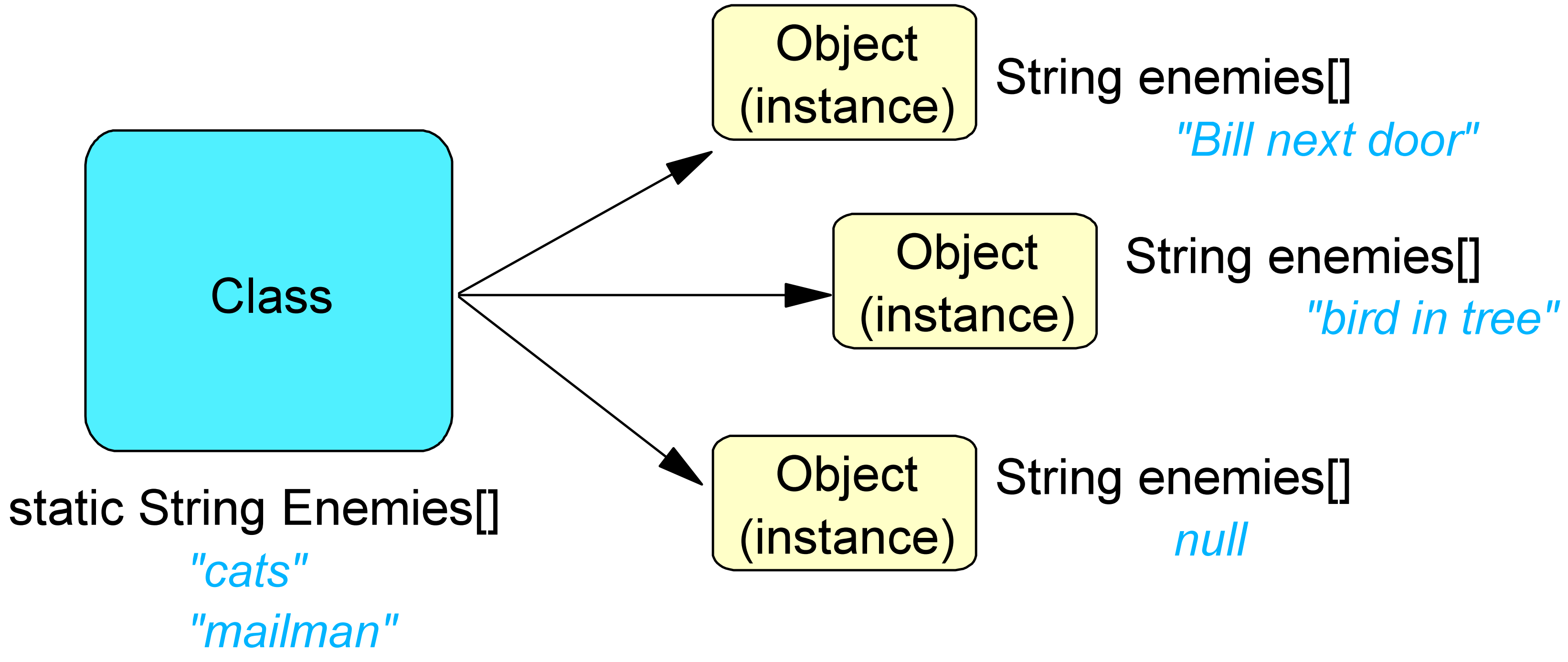
- Upon *instantiation* of a new object the *constructor* is automatically called by the JVM
- If not specified a default constructor is used

```
public class Terrier {
        Terrier() {
                super();
                initialize();
        }
}
```



Static Variables and Methods

- **The *static* keyword identifies that the variable or method belongs to the entire class rather than the individual instance**

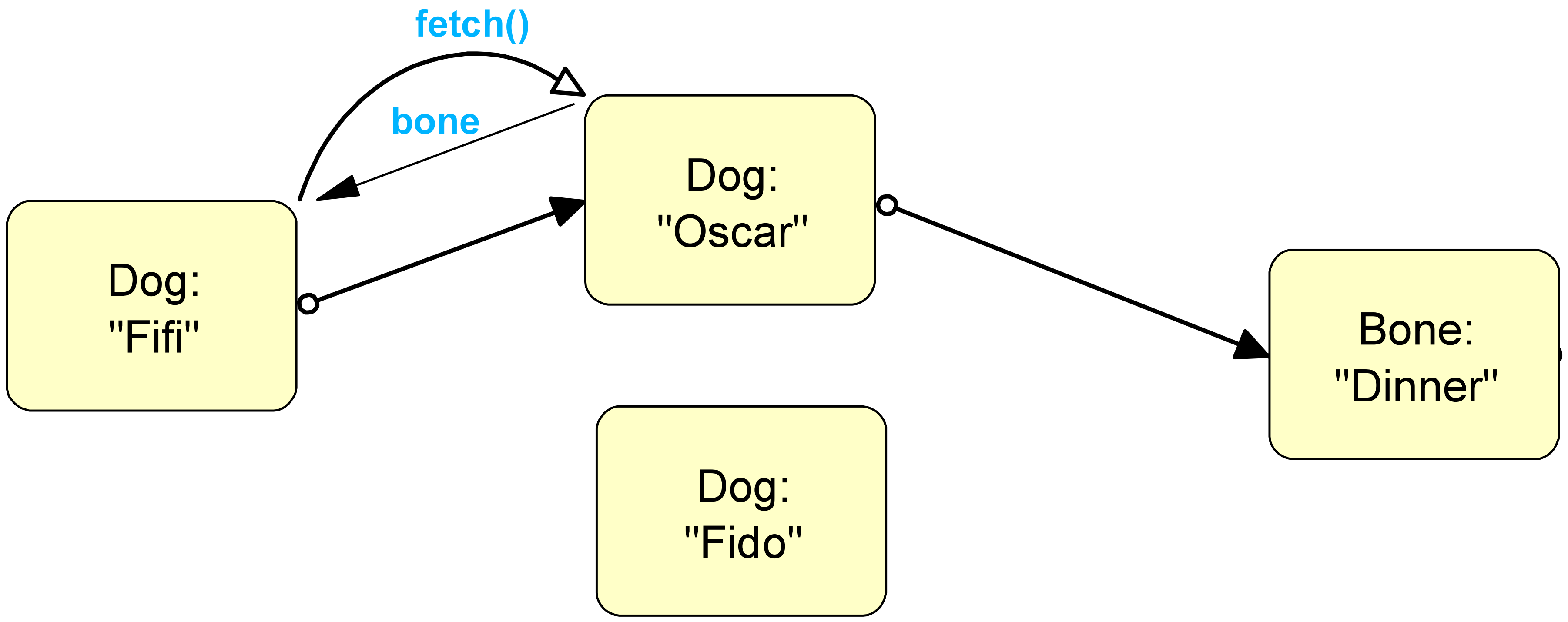


Abstraction and Interfaces

- **An abstract class cannot be instantiated, yet provides a default implentation for its subclass**
- **An interface is an abstract class consisting of a collection of method definitions (without implementation)**
  - Any decendant of an interface remains abstract until it implements or inherits an implementation of every method defined in the interface

Object Communication

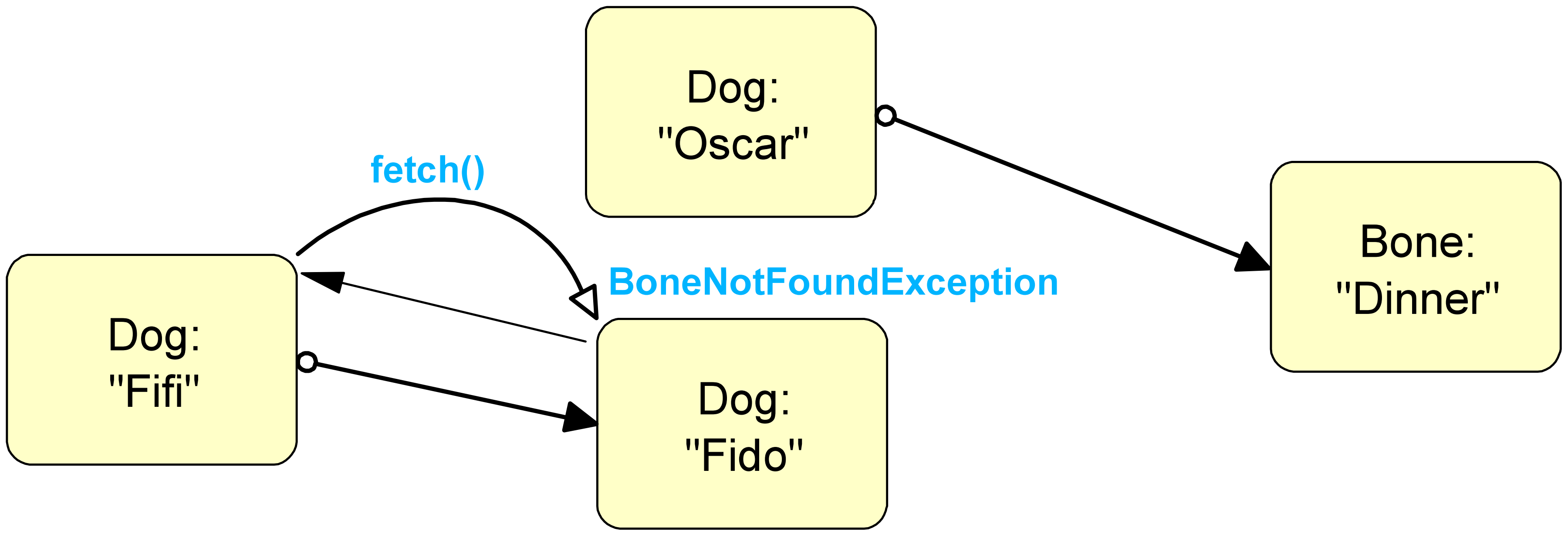
- Objects communicate through method calls
- Objects must possess a reference to another object in order to call its methods or access its data



IBM Software

Exceptions

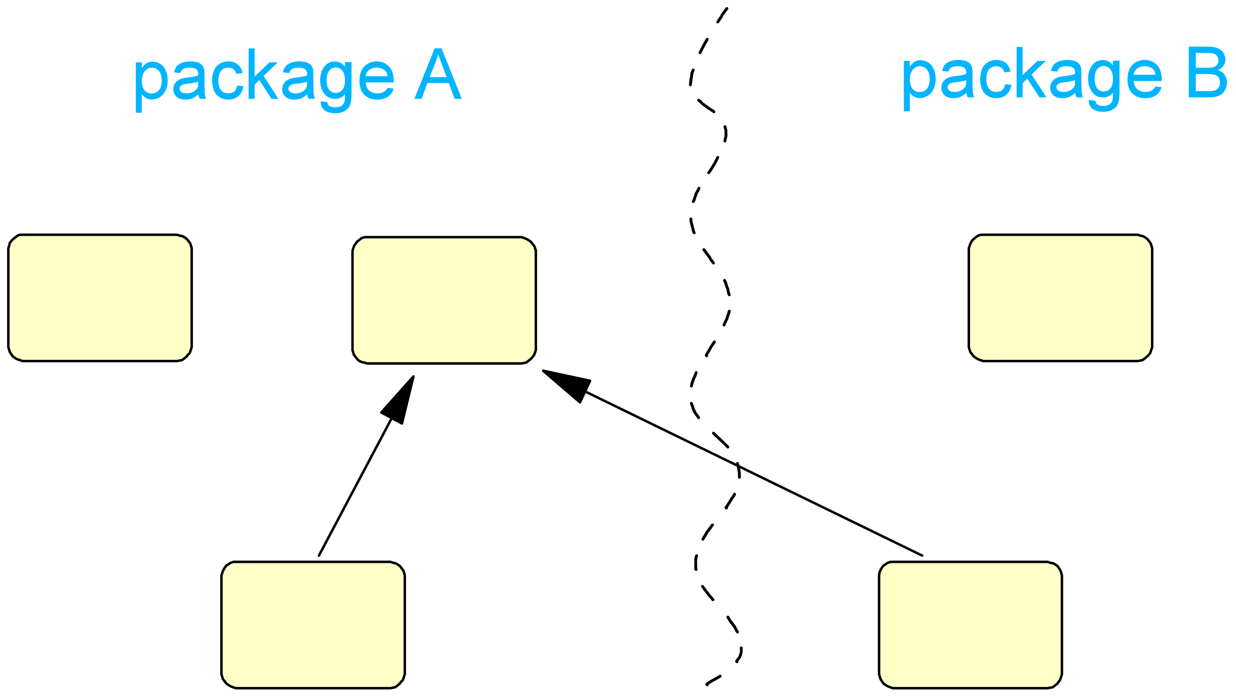
- An event during program execution that prevents the program from continuing normally; generally, an error.
- Methods are referred to as *throwing* an Exception, which can be *caught* by a block of code capable of handling the exception; typically, gracefully.



IBM Software

Access Privileges

- **Private** - member is only accessible by *self*
- **Package** - member is only accessible by objects in *same package*
- **Protected** - member is only accessible by *self, subclasses, and* objects in *same package*
- **Public** - member is accessible by *everyone*



Code Example

```
package animal.dog;
public abstract class Dog {
      public void speak() {
            makeSound("bark");
      }
}
```

```
package animal.dog;
public class ToyPoodle extends Dog {
      public void speak() {
            makeSound("yip");
      }
}
```

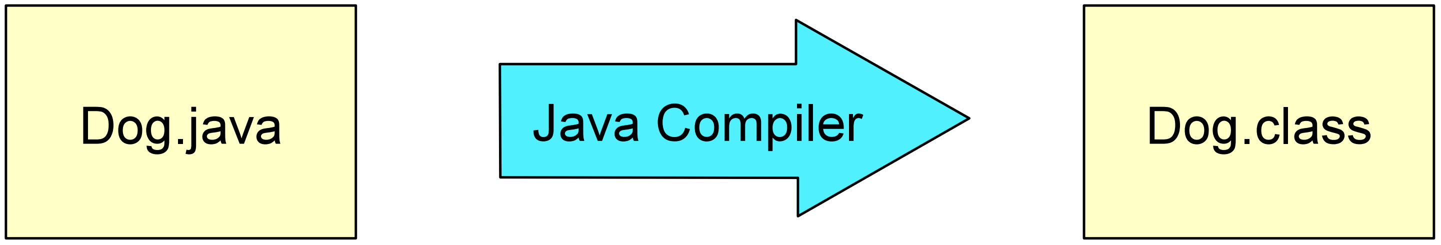
```
package animal.dog;
public class St_Bernard extends Dog {
      public void speak() {
            makeSound("WOOF");
      }
}
```

```
Dog someDog = new ToyPoodle();
someDog.speak();
```

IBM Software

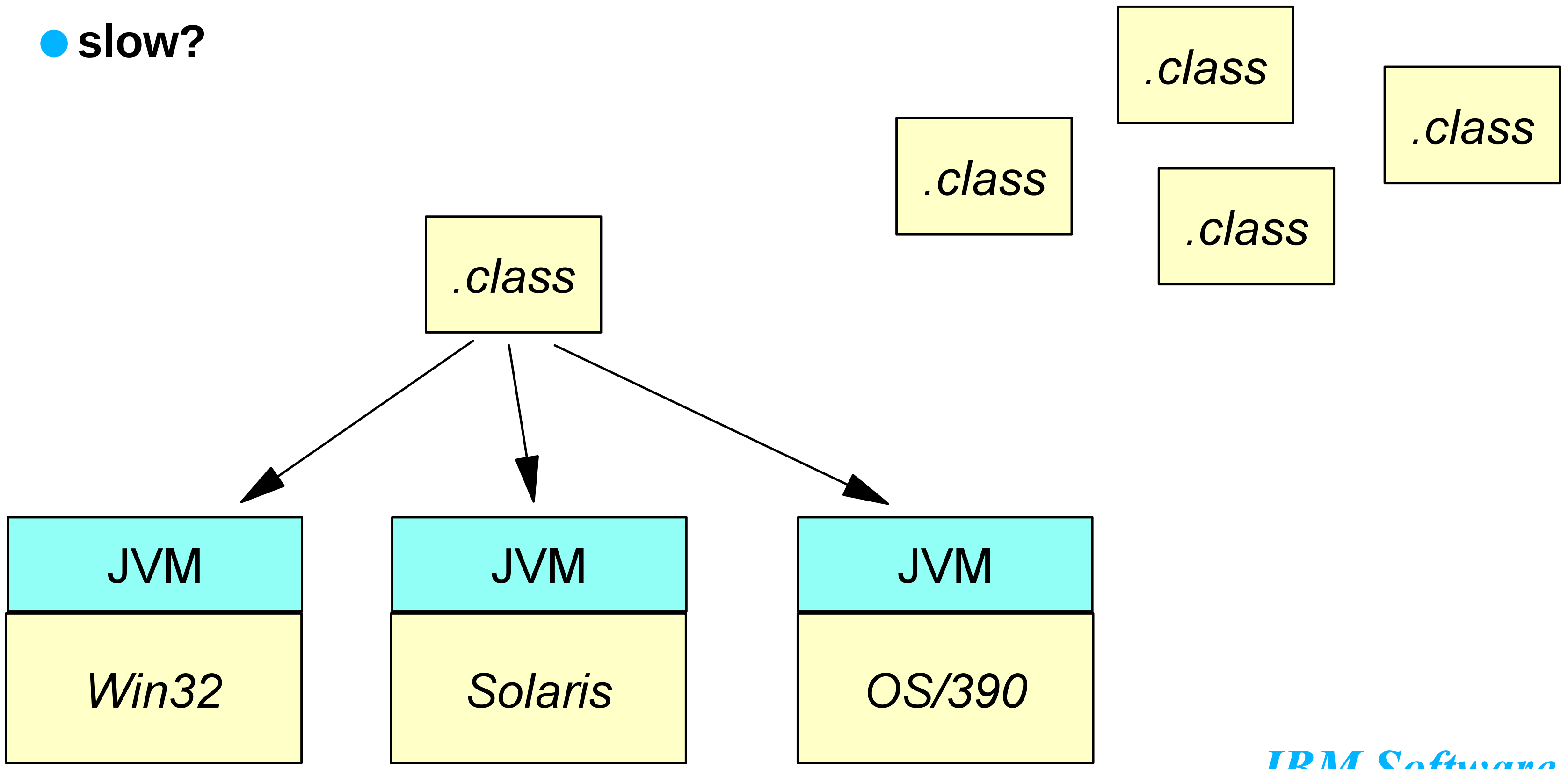
Java Compiler

- **The Java Compiler compiles Java programs down to an intermediate platform independent set of binary instructions, called *bytecode***
- **Every Java object is compiled into a corresponding .class file of the same name containing:**
  - Java Virtual Machine instructions (bytecode)
  - symbol table
  - other information



Java Virtual Machine (JVM)

- Bring up Java Virtual Machine (java ToyPoodle)
- Compiles bytecode into platform-specific machine code on the fly at runtime (interprets)
- Dynamic Loading
- slow?



IBM Software

Just In Time Compiler (JIT)

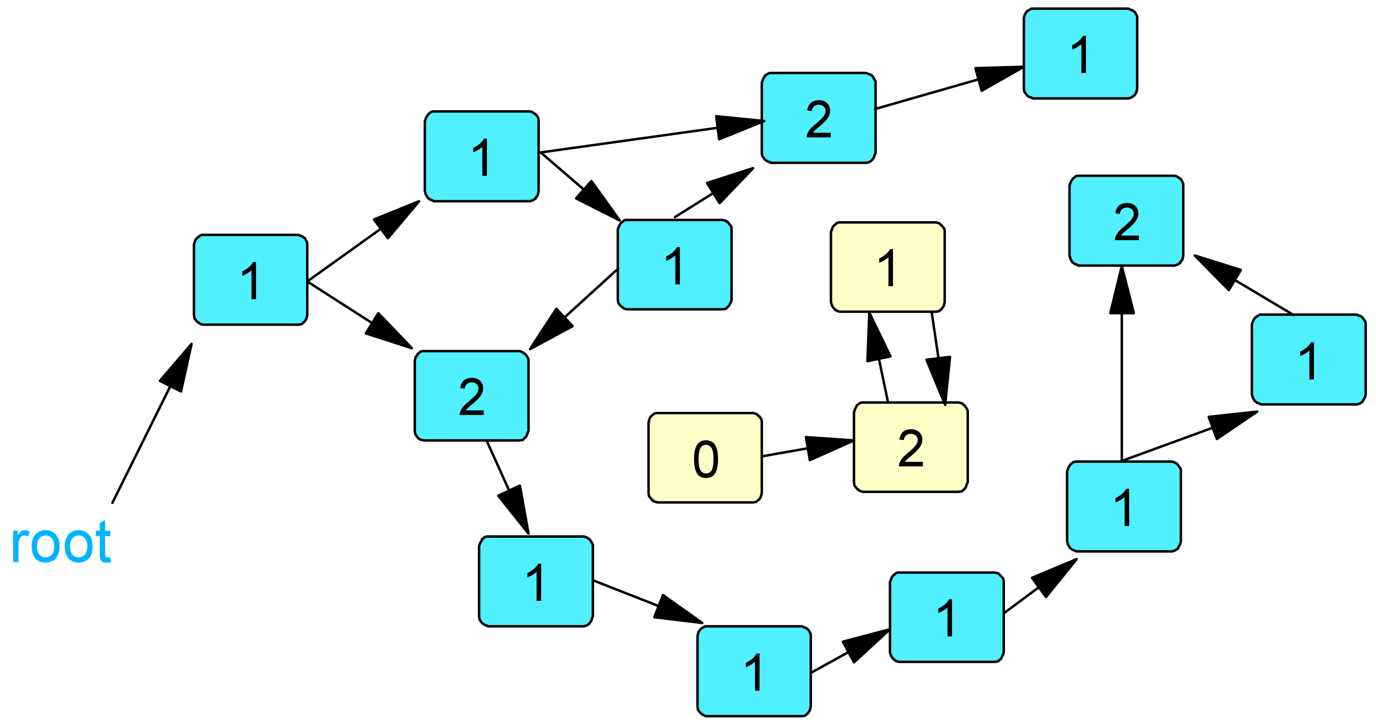
- Compiles bytecode on a method by method basis
- Stores JIT’d methods in memory for later re-use
- Depending on program can drastically improve performance

Java Native Interface (JNI)

- **JNI is a native programming interface**
- **Allows code running in a JVM to interoperate with applications and libraries written in other programming languages such as**
  - C
  - C++
  - Assembly
  - Depends on JVM implementation
- **Possible uses include**
  - Code reuse
  - Performance
  - Platform Dependent code segments

Garbage Collection

- **Remove storage management responsibilities from programmer**
- **JVM implementation specific**
  - reference counting
  - reference tracing (preferred)



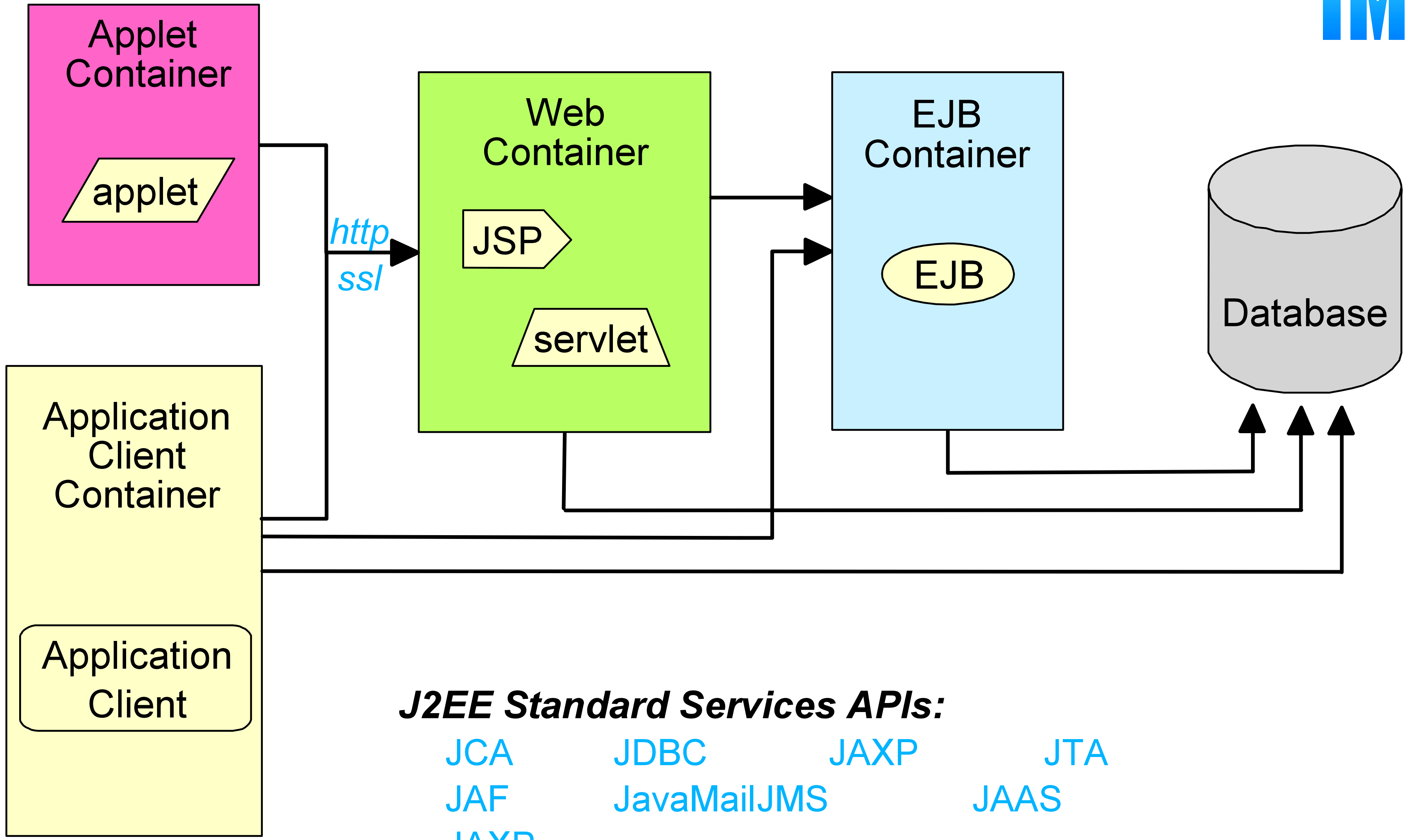
Object Life Cycle

- **Code / Compile**
- **Load / Instantiation**
- **Use**
- **Garbage Collection**

J2EE

- Java 2 Platform, Enterprise Edition (J2EE)
- Defines a standard component-based approach to the design, development, assembly, and deployment of enterprise applications

J2EE Architecture

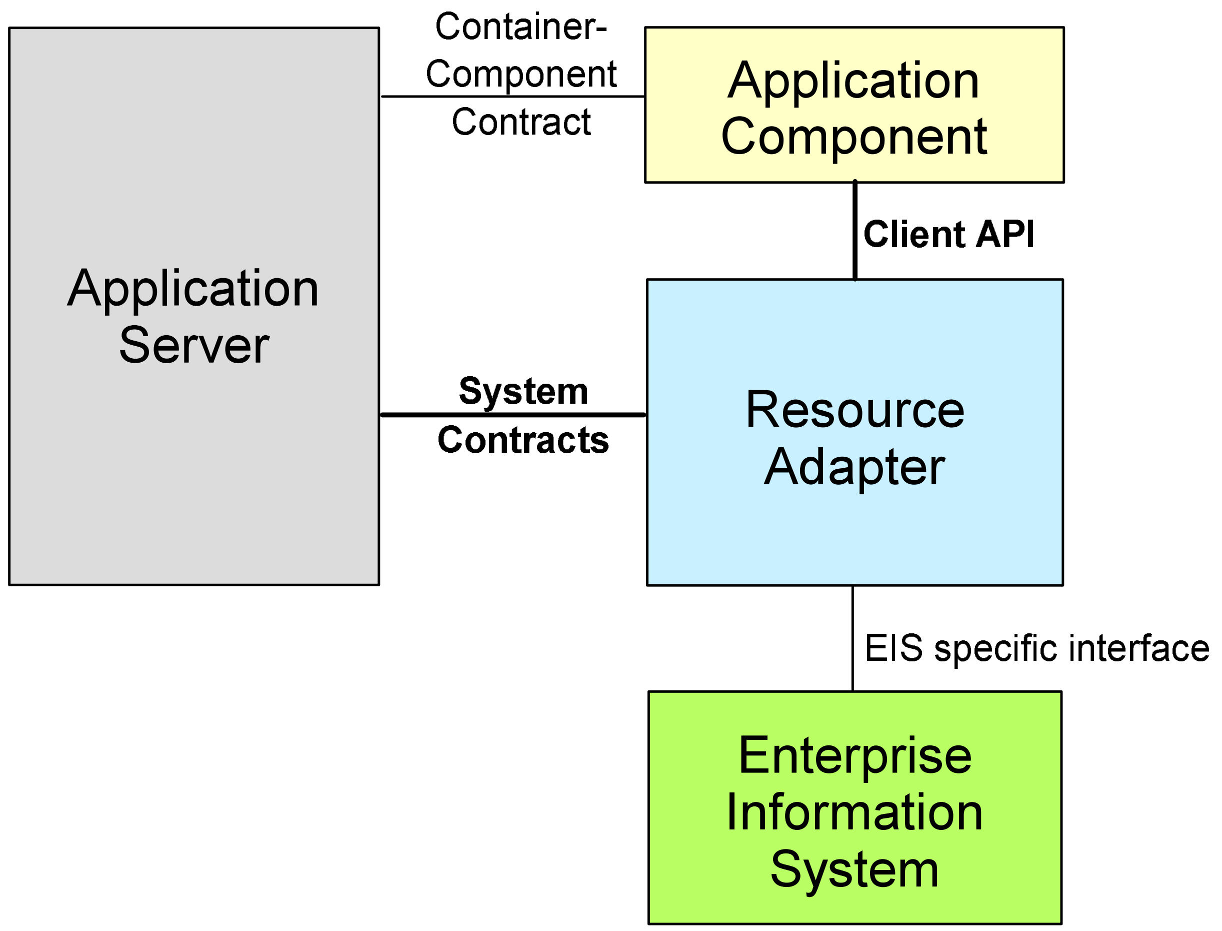


J2EE Standard Services APIs:

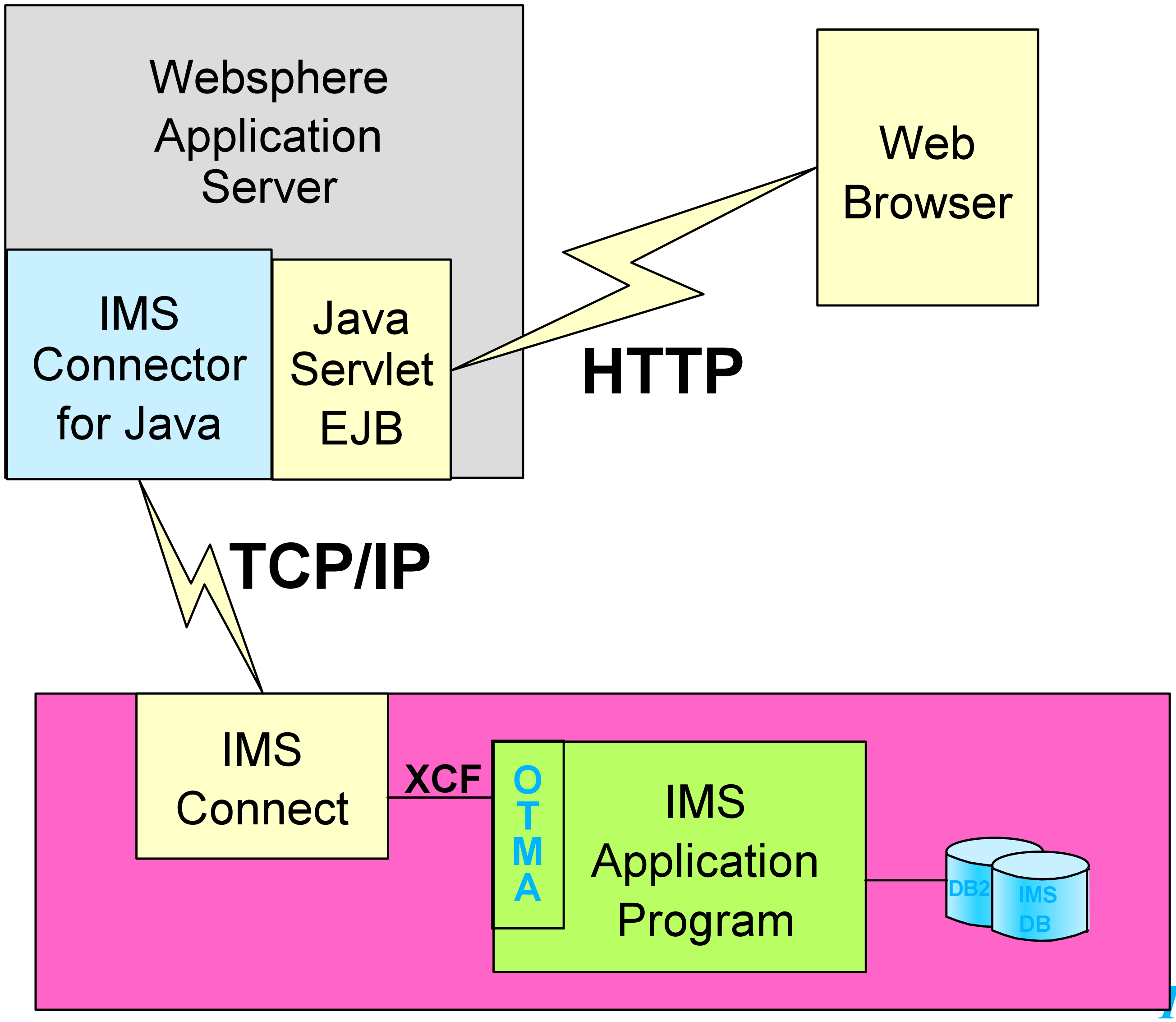
JCA	JDBC	JAXP	JTA
JAF	JavaMailJMS		JAAS
JAXP			

IBM Software

J2EE Connection Architecture (JCA)



IMS Connector for Java

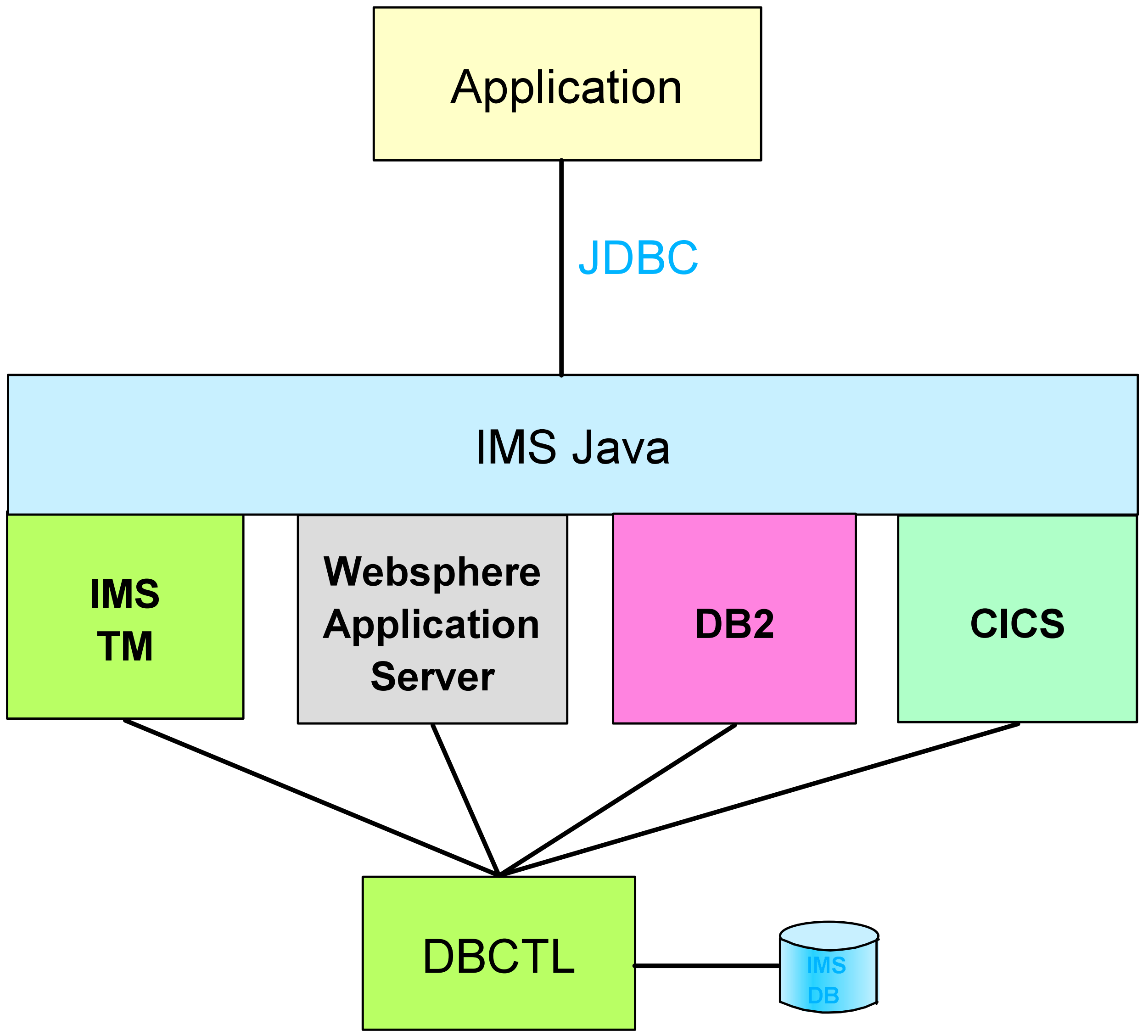


IBM Software

Java Database Connectivity (JDBC)

- **Defines a standard Java API for accessing relational databases**
- **Provides an API for sending SQL statements to a database and processing the tabular data returned by the database**

IMS Java



IBM Software

Conclusions

- Java is fun