**JAC444 - Lecture 2**

Interfaces

Segment 2

**Interfaces**

**In this segment you will be learning about:**

▪ Interfaces

▪ Default methods, Private methods

▪ Interface inheritance

▪ Annotations

▪ Functional interface

# Interface Definition

Interface is a data type in Java. It is a collection of abstract methods. An interface may also contain constants, default methods, static methods, and nested types.

**interface InterfaceName {**

**abstract method declaration(s)**

**constant(s) - final static fields**

**default method(s)**

**static method(s)**

**nested types**

**}**

An interface creates a new reference data type, just as class definition

**InterfaceName refVariable;**

# Interface Structure

* All methods in an interface are abstract and public

( a method without implementation is an abstract method)

* Variables declared in interface are public, static and final by default
* Java 8 allows *default method* - method with implementation
* Java 9 allows *private method -* improve code reusability

# Interface Example

**public interface Conversion { double INCH\_TO\_MM = 25.4; double inchToMM(double inches); }**

Conversion c; // c is a reference of an object of type Conversion

**public interface ConversionVersion2 { double INCH\_TO\_MM = 25.4; double inchToMM(double inches); default public void defaultMethod() {**

**System.out.println("Special implementation");**

**}**

**}**

# Implementing an Interface

An interface defines a protocol of behavior.

A class obeys the protocol defined by interface by using the Java keyword **implements**

**class MyConversion implements Conversion { double inchToMM(double inches) { //implementation**

**}**

**}**

**Conversion c = new MyConversion(); double mm = c.inchToMM(…);**

# Private Method in Java 9

* Java 7 has only: **public abstract methods**
* Java 8 has: **public static public default methods**
* Java 9 has:  **private method**
* The valid combinations:

**public static** - correct

**public abstract** - correct

**public default** - correct

**private static** - correct

**private abstract** - compile error **private default** - compile error

**private** - correct

# Multiple Inheritance

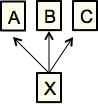
Implementing Interface

**interface Iable { void methodOne(); }**

**class First implements Iable { void methodOne() { … } }**

Extending Interface

**interface Jable extends Iable { String methodTwo(int i); } class Second implements Jable { void methodOne() { … } String methodTwo(int i) { … }**

**}**

Interface Multiple inheritance **interface X extends A, B, C { … }**

# Marker Interface

• **A marker interface** is an interface with **no methods** (empty body)

**interface Markable { }**

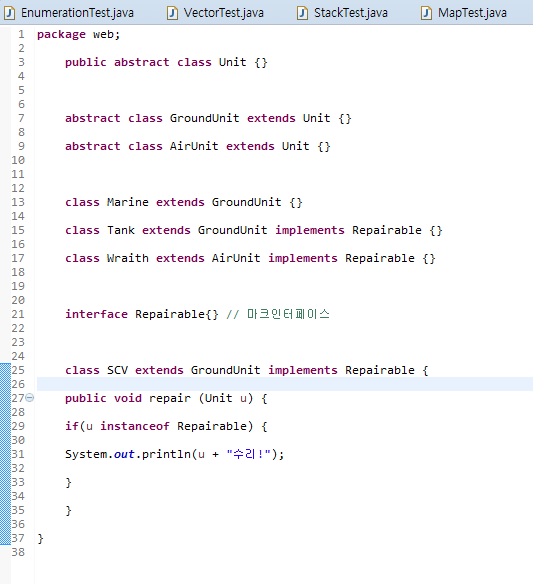
**class Special implements Markable { }**

**Markable obj = new Special();**

Example: **java.io.Serializable**

**Marker Interface , instanceof 객체지향적**

<http://blog.naver.com/jack716/221019436686>



# Annotations

* Data that provides information about other data is called metadata
* **Annotation is a language construct** that **provides metadata** to Java source elements.
* Classes, methods, variables, parameters and interfaces may be annotated

// Declares the annotation Important.

**public @interface Important { }**

// @Important is an annotation to method say().

**@Important public String say(char c) {**

**}**

# Functional Interface

* **A functional interface** is an interface with **an exactly one abstract method**

**interface Workable {**

**String work(int j);**

**}**

* To emphasize that an interface is a Functional interface one can use annotation

**@FunctionalInterface interface Workable {**

**String work(int j);**

**}**