Concepts of Java

TA: Jimin Hsieh

Agenda

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Encoding

- Please use English as your variables.
- Do not use Chinese or Japanese character as variables.
- Try to use English to comment your source code.
- Remember always use UTF-8 as your encoding or your code couldn't cross platform.

Agenda

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Class, Instance, and Object

- Class: A class represents a template for several objects and describes how these objects are structured internally.
- Instance: An instance is an object created from a class.
- Object: An object is characterized by a number of operation and a state which remembers the effect of these operations.

Agenda

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Static

- It means that particular attribute or method is not tied to any particular object instance of that class.
- When you can see the static?
  - java.lang.Math
  - java.lang.Array
- When you have not to use static?
  - When your class will create individual object.

Static

Static variable

- A static variable is associated with its class rather than its ~~object~~.
- It is also known as class variable.

Static method

- A static method can only access static variables and call only static methods.
- You can not use this in static method.
  - Static method is associated with a class and not an instance.

Static Variables

class StaticTest {static int i = 47;}

StaticTest st1 = new Static StaticTest();

StaticTest st2 = new Static StaticTest();

- st1.i == st2.i == StaticTest.i will be true;
- StaticTest.i++;
  - st1.i and st2i will be 48

Count How much object you create

```
public class Counter{
    private static int count;
    public Counter(){count++;}
    public static void printCount(){
        System.out.println("Number of instances created so far is: " + count);
    }
```

Static Method

- Static methods are not object-oriented.
  - No ***this*** and ***super***.
  - Don't send a message to an object.(Constructor)
- If you find yourself using a lot of static methods, you should probably rethink your strategy.
- If a method is static, it doesn't behave polymorphically.(No ***super***)

Static

Do you ever think about why we should use main with static?

Static

Do you ever think about why we should use main with static?

Ans: ***JVM do not need to instantiate a object.***

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Constructor

- A special method is invoked automatically. The constructor initializes the newly created object.
- The main rule of constructors is that they should have the same name as the class and no return type.
- If you do not give a constructor for a class, the Java compiler provide a default constructor without parameter.

Constructor

- A class can have more than one constructor. (Overloaded)

Let's see the source code on ftp server.

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Encapsulation

- Define: Wrapping data and methods within classes in combination with implementation hiding.
- Why?

- Define: Wrapping data and methods within classes in combination with implementation hiding.
- Why?
  - Access Control/Information Hiding

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Overloading

- Definition: Different types of arguments, different number of arguments, or both. Return type should be the same.

Overriding

- Definition: A method in subclass with the same signature(method name, number of parameters, parameter types and order of parameter types)
- **@Override**
  - Avoid some mistakes.
  - Let your code more readable.

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

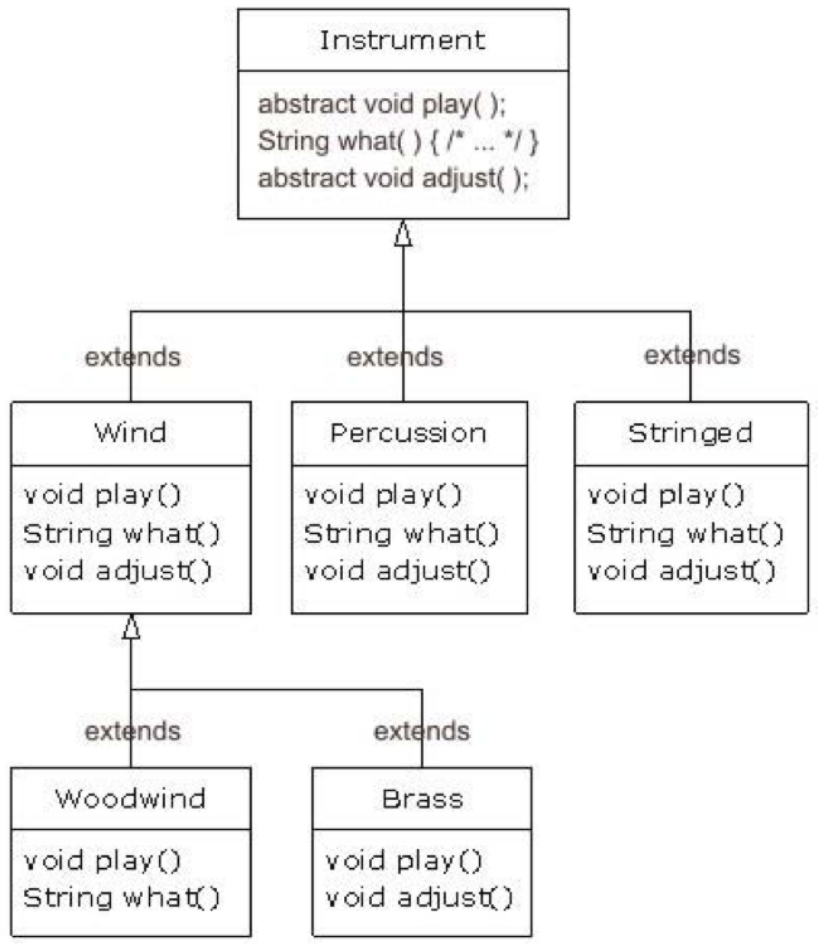
Abstract Class

- Abstract Method
  - It is declared without an implementation.(Method Body)
- Abstract Class = Abstract Base Class
  - It may not include abstract method.
  - If it include abstract method, the class itself must abstract class.
  - It cannot be instantiated, but they can be subclassed.

Sample Code

Music4.java

UML


Instrument
abstract void play( );
String what( ) { /* ... */ }
abstract void adjust( );
extends
extends
extends
Wind
void play()
String what()
void adjust()
Percussion
void play()
String what()
void adjust()
Stringed
void play()
String what()
void adjust()
extends
extends
Woodwind
void play()
String what()
Brass
void play()
void adjust()

- Encoding
- Class, Instance, and Object
- Static
- Constructor
- Encapsulation
- Overloading & Overriding
- Abstract
- Interface

Interface

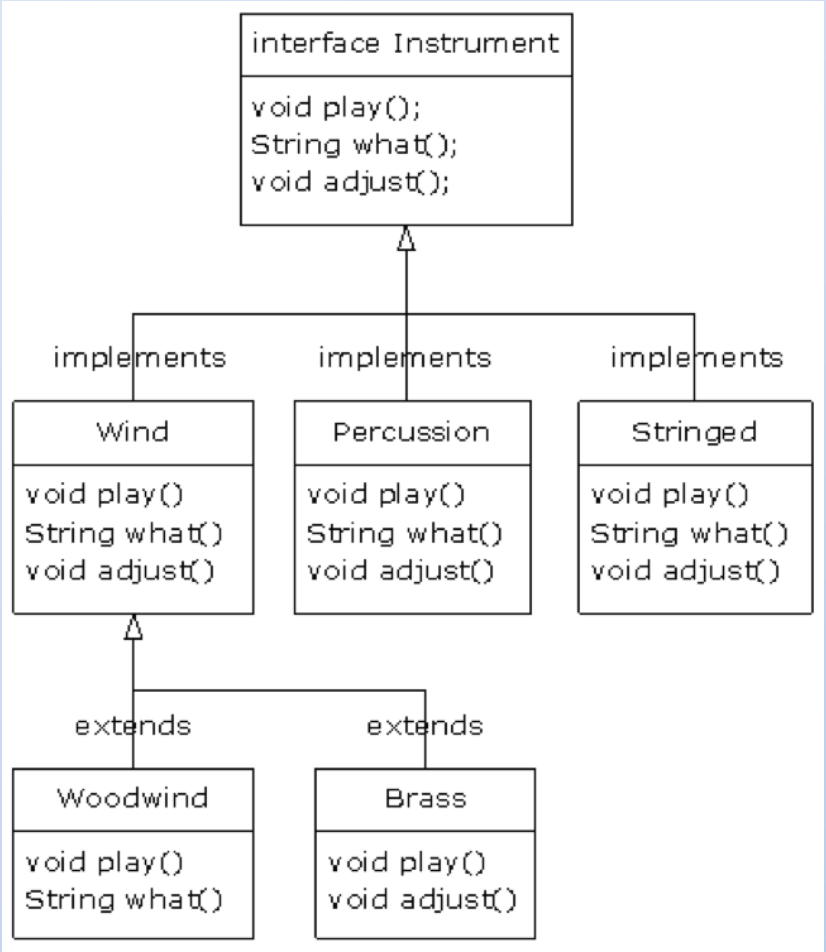
- The **interface** keyword produces a completely abstract class, one that provides no implementation at all.
- An interface provides only a form, but no implementation.
- An interface can also contain fields, but these are implicitly **static** and **final.**
  - This cannot be changed.

Sample Code

- You need to use **implements** this keyword.

Music5.java

UML


interface Instrument
void play();
String what();
void adjust();
implements
implements
implements
Wind
void play()
String what()
void adjust()
Percussion
void play()
String what()
void adjust()
Stringed
void play()
String what()
void adjust()
extends
extends
Woodwind
void play()
String what()
Brass
void play()
void adjust()

Abstract class or Interface?

- If it's possible to create your base class without any method definitions or member variables, you should always prefer interfaces to abstract classes.
- Abstract class
  - What object is.
- Interface
  - What object can do.
  - It can used for multiple inheritance.

Reference

1. Thinking in Java
2. Oracle Certified Professional Java SE 7 Programmer Exams 1Z0-804 and 1Z0-805: A Comprehensive OCPJP 7 Certification Guide
3. Java: How to Program, 9th Edition
4. Stack Overflow

Q&A

Reference:

Thanks for you attentions!

Casting

- Casting means is taking an Object of one particular type and turn it into another Object type.
- By casting, it changes the way the compiler sees an object.