Java Class

*. Class name ⇒ first character Capital !!

1. Mircophone.java

```
public class Microphone {

    String name;
    String color;
    int model;

    public Microphone(String name, String color, int model){

        this.name = name;
        this.color = color;
        this.model = model;
    }

    public void turnoff ( ) {

        System.out.println("Turn off" + this.name);

    }

}
```

class name

class attribute

Constructor

class method

for object attribute

注意!!! 不用static!!!

(若加上static 無法用this, why?)

2. MyClass.java

```
public class MyClass {
    public static void main ( String[ ] arg){
        Microphone mic = new Microphone();
        mic.name ="logan";
        mic.color = "red";
        mic.model = 10;
        mic.turnoff();
        Microphone mic = new Microphone("A","Red",10)
    }
}
```

Constructor in Inheritance

1. Person.java

Base class

```
public class Person{
    String name ;
```

```
    int age;
```

Attribute

```
    public Person(String name, int age){
```

Base Class Constructor

```
        this.name = name;

        this.age = age;

    }

}
```

2. Hero.java

Child Class

繼承用"extend"

```
public class Hero extends Person{
```

Base class

```
    int power;
```

Child class attribute

```
    public Hero(String name, int age, int power){

        super(name, age);

        this.power = power;

    }

}
```

child class constructor

*** Java中, Constructor不会被繼承!!!

∴ 用 "Super" call base class constructor

(same as python)

Getters and Setters

1. Microphone.java

```
public class Microphone {
    private String name;
    private String color;
    private int model;

    public String getName(String name) {
        return this.name;
    }
}
```

class attribute

getter

```
    public void setName(String name) {
        this.name = name;
    }
}
```

setter

*. Once we set attributes to private
we cannot access through "."
Thus, we need getter & setter

(Alt + Insert : Auto complete)

Overloading Constructor

1. Mircophone.java

```
public class Microphone {

    String name;
    String color;
    int model;
                                            Constructor 1
    public Microphone(){ }
                                              Constructor 2
    public Microphone(String name, String color, int model){
        this.name = name;
        this.color = color;
        this.model = model;
    }
}
```

*. Java Class Doc:

docs.oracle.com

Java Inheritance

*. Remember to add "package com.logan.inheritance;" on top of the class file!

Overriding Super Class Method

Child Base

```
public class Manager extends Employee {
```

override function in Base

```
    @Override
    public double getAnnualSalary() {
            return super.getAnnualSalary() + 1000 ;

    }
```

Override "toString" method in "Object" Base Class!

```
    @Override
    public String toString() {
            return this.getName() + ", "
                   this.getId() + ", "
                   this.getAnnualSalary() ;

    }
}
```

*. In main() ↓

```
public static void main(String[] arg) {
```

```
Manager logan = new Manager();

System.out.println(logan);

}
```

If we didn't override toString method The address of logan will be printed out. Instead, this will print out a string of logan's information.

*. Note: Every obj in Java are sub-classes of "Object" class.

Check out Object Class method,

Type: Object obj = new Object().