Week 11-1 : Class Hierarchy

Part 1. Inheritance
Inheritance between classes. When a child class is inherited from parent class, a child class has parent class' member.
Parent class is a superclass, and derived class is a subclass.
Inheritance keyword : extends

- Needs of Inheritance
Don't need to declare member repeatedly.
simplifying a class by reusing fields and methods
managing and classifying classes hierarchically

- Example of Inheritance

```java
class Point {
    int x, y;
    void set(int x, int y) {
        this.x = x; this.y = y;
    }
    void showPoint() {
        System.out.println("x : "+ x + "/ y : " + y);
    }
}

class ColorPoint extends Point {
    String color;
    void setColor(String color) {
        this.color = color;
    }
    void showColorPoint() {
        System.out.print(color);
        showPoint();
    }
    public static void main(String[] args) {
```

```java
```
ColorPoint cp = new ColorPoint();
    cp.set(3,4);
    cp.setColor("red");
    cp.showColorPoint();
}

ColorPoint class inherit Point class. ColorPoint type cp has ColorPoint's member and Point's member. Don't need to implement another for coordinates. Only reuse the method implemented in the Point. ColorPoint has set method about color information and show method about color information and coordinates.

- Access modifier

<table>
<thead>
<tr>
<th>Class accessing member</th>
<th>Member's access modifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>default</td>
</tr>
<tr>
<td>Same package class</td>
<td></td>
</tr>
<tr>
<td>Different package class</td>
<td>X</td>
</tr>
<tr>
<td>Same package's subclass</td>
<td>O</td>
</tr>
<tr>
<td>Different package's subclass</td>
<td>X</td>
</tr>
</tbody>
</table>

class Person {
    public String name;
    protected int height;
    private int weight;
    int age;

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public void getWeight() {
        return weight;
    }
}
public class Student extends Person {
    void set() {
        name = "Gildong Hong";
        height = 180;
        setWeight(70);
        age = 20;
    }

    public static void main(String[] args) {
        Student s = new Student();
        s.set();
    }
}

public String name : Possibly accessing a Student class
protected int height : Possibly accessing a Student class (inheritance)
private int weight : Not possibly accessing a Student class (using methods)
int age : Possibly accessing a Student class
* this Constructor : using this constructor when parameter name and member name are same.

Part 2. Constructor
- Inheritance and Constructor
Superclass' constructor executes first, after that subclass' constructor executes.
```java
public class Student extends Person {
    public Student() {
        System.out.println("Student Constructor");
    }
}
```

```java
public class Constructor {
    public static void main(String[] args) {
        Student s = new Student();
    }
}
```

Person constructor executes first, after that student constructor executes.
[Exercise]
Make Person, Student, Constructor Classes. A Student class inherit a Person class.

1. Person Class
String name variable
no parameter constructor.
one parameter constructor : print out "person : a parameter value"

2. Student Class
no parameter constructor.
one parameter constructor :
   first line super(parameter name);
   print out “student : a parameter value”

3. Constructor Class
main method
generate Student type object ; parameter ("Minsu");

Question
1. Run the program. What is super() method?
2. In the Student class, change the two line in the Student(String x) Constructor. What happen?