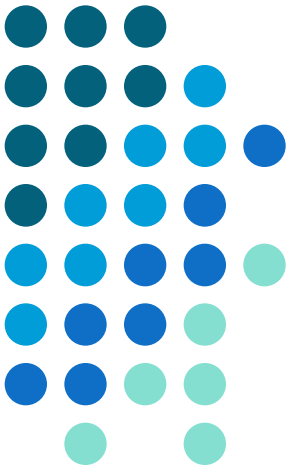
10. 상속



목차

- 01. 상속이란
- 02. 메소드의 오버라이딩
- 03. 상속에서의 생성자

01. 상속이란

❶ 상속은 코드의 재활용의 개념

❷ 상속은 부모에게 무엇인가를 물려받는 것을 의미한다.

❸ 새로 만드는 클래스를 이미 정의된 훌륭한 클래스의 상속을 받는다면 많은 기능을 모두 물려받아 사용할 수 있다.

❹ 부모가 되는 클래스를 수퍼 클래스, 자식 클래스에 해당되는 클래스를 서브 클래스

1.1 수퍼 클래스와 서브 클래스 만들기

형식

```
class 서브_클래스 extends 수퍼_클래스 {
}
```

```
1  package kame.ch07.ex01;
2
3  public class Point2D {
4      public int x;
5      public int y;
6  }
```

```
1  package kame.ch07.ex01;
2
3  public class Point3D extends Point2D {
4      public int z;
5  }
```

```
1  import kame.ch07.ex01.Point3D;
2
3  public class Ch07Ex01 {
4      public static void main(String[] args) {
5          Point3D pt=new Point3D();
6          pt.x=10; pt.y=20; pt.z=30;
7          System.out.println(pt.x+","+pt.y+","+pt.z);
8      }
9  }
```

1.2 protected 접근 지정자

- 상속이 적용되는 클래스
  - protected 접근 지정자
    - 서브 클래스가 수퍼클래스의 필드에 접근가능
    - 단, 클래스 외부에서는 접근 불가능

실 습 하 기 : protected 필드를 갖는 클래스 설계하기

```java
package kame.ch07.ex01;

public class Point2D {
    protected int x;
    protected int y;
}
```

```java
package kame.ch07.ex01;

public class Point3D extends Point2D {
    protected int z;
}
```

실 습 하 기 : protected 필드를 갖는 클래스 설계하기

```
1  import kame.ch07.ex01.Point3D;
2
3  public class Ch07Ex01 {
4      public static void main(String[] args) {
5          Point3D pt=new Point3D();
6          pt.x=10; pt.y=20; pt.z=30;
7          System.out.println(pt.x+","+pt.y+","+pt.z);
8      }
9  }
10
```

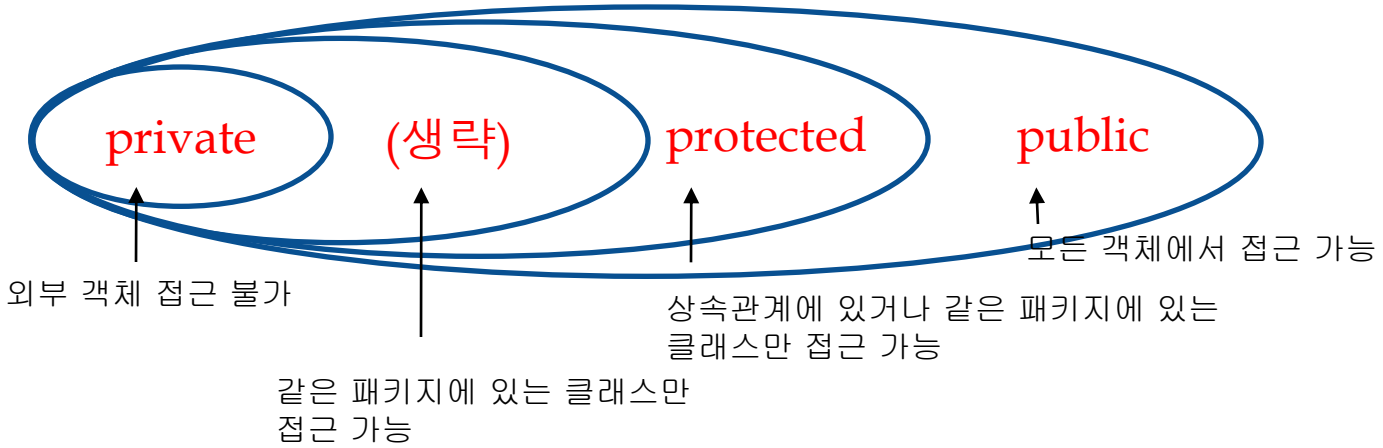
Problems | Javadoc | Declaration | Console

```
<terminated> Ch07Ex01 [Java Application] C:₩Program Files₩Java₩jre6₩bin₩javaw.exe (2012. 1. 3. 오후 3:03:47)
Exception in thread "main" java.lang.Error: Unresolved compilation prc
        The field Point2D.x is not visible
        The field Point2D.y is not visible
        The field Point3D.z is not visible
        The field Point2D.x is not visible
        The field Point2D.y is not visible
        The field Point3D.z is not visible

        at Ch07Ex01.main(Ch07Ex01.java:6)
```

1.2 접근 지정자의 차이점

수식어	자신의 클래스	같은 패키지	하위 클래스 (상속)	다른 패키지
private	O	X	X	X
(생략)friendly	O	O	X	X
protected	O	O	O	X
public	O	O	O	O



실 습 하 기 : 자바의 모든 형태의 접근 지정자의 접근 허용 범위 알아보기

패키지 kame.ch07.ex02

```
1  package kame.ch07.ex02;
2
3  public class AccessSuper {
4      private int m_pri=10;
5      int  m_def=20;
6      protected  int  m_pro=30;
7      public int m_pub=40;
8      public void showAccessSuper() {
9          System.out.println(m_pri);
10         System.out.println(m_def);
11         System.out.println(m_pro);
12         System.out.println(m_pub);
13     }
14 }
```

실 습 하 기 : 다른 패키지의 상속 관계에 있는 클래스에서 접근 허용 범위 알아보기

패키지 default

```
1  import kame.ch07.ex02.AccessSuper;
2
3  public class AccessSub extends AccessSuper {
4      void showAccessSub(){
5          System.out.println(m_pri);
6          System.out.println(m_def);
7          System.out.println(m_pro);
8          System.out.println(m_pub);
9
10     }
11 }
```

실 습 하 기 : AccessSuper와 전혀 상관없는 클래스에서의 접근 지정자의 접근 허용 범위 알아보기

패키지 default

```
1  public class Ch07Ex02 {
2
3      public static void main(String[] args) {
4          AccessSub obj=new AccessSub();
5          obj.showAccessSuper();
6          obj.showAccessSub();
7          System.out.println(obj.m_pri);
8          System.out.println(obj.m_def);
9          System.out.println(obj.m_pro);
10         System.out.println(obj.m_pub);
11     }
12 }
```

실습: 메소드 상속받아 사용

패키지 kame.ch07.ex03

```java
package kame.ch07.ex03;

public class Point2D {
    protected int x=10;
    protected int y=20;
    public void show() {
        System.out.println(x);
        System.out.println(y);
    }
}
```

패키지 kame.ch07.ex03

```java
package kame.ch07.ex03;

public class Point3D extends Point2D {
    protected int z=30;
}
```

패키지 default

```java
import kame.ch07.ex03.Point3D;

public class Ch07Ex03 {
    public static void main(String[] args) {
        Point3D pt=new Point3D();
        pt.show();
    }
}
```

02. 메소드 오버라이딩

- 오버라이딩(override)
  - 수퍼 클래스의 메소드를 서브 클래스에서 상속받아 사용
  - 서브 클래스만의 독특한 기능을 수행하도록 해야 할 경우
- 메소드 오버라이딩의 특징
  - 수퍼 클래스의 메소드를 서브 클래스에 다시 정의
  - 이름, 매개 변수의 자료형, 개수까지 동일
- 메소드 오버라이딩이란
  - 수퍼 클래스의 메소드를 서브 클래스에 다시 정의하는 것으로 이름뿐만 아니라 매개변수의 자료형이나 개수마저도 동일해야 한다
  - 수퍼 클래스의 메소드는 은닉, 서브 클래스의 메소드가 사용됨

실습: 메소드 오버라이딩

패키지 kame.ch07.ex03

```
1  package kame.ch07.ex03;
2
3  public class Point2D {
4      protected int x=10;
5      protected int y=20;
6      public void show() {
7          System.out.println(x);
8          System.out.println(y);
9      }
10 }
```

패키지 kame.ch07.ex03

```
1  package kame.ch07.ex03;
2
3  public class Point3D extends Point2D {
4      protected int z=30;
5
6      public void show() {
7          System.out.println(x);
8          System.out.println(y);
9          System.out.println(z);
10     }
11 }
```

패키지 default

```
1  import kame.ch07.ex03.Point3D;
2
3  public class Ch07Ex03 {
4      public static void main(String[] args) {
5          Point3D pt=new Point3D();
6          pt.show();
7      }
8  }
```

2.1 레퍼런스 변수 super

- 서브 클래스에서 오버라이딩하면, 수퍼 클래스의 해당 메소드가 은닉
- super.메소드( );
  - 은닉된 수퍼 클래스의 메소드 호출

```
1  package kame.ch07.ex03;
2
3  public class Point2D {
4      protected int x=10;
5      protected int y=20;
6      public void show() {
7          System.out.println(x);
8          System.out.println(y);
9      }
10 }
```

```
1  package kame.ch07.ex03;
2
3  public class Point3D extends Point2D {
4      protected int z=30;
5
6      public void show() {
7          super.show();
8          System.out.println(z);
9      }
10 }
```

3. 상속에서의 생성자

- 상속에서 생성자의 특징
  - 생성자는 상속되지 않는 유일한 멤버 함수
  - 서브 클래스의 인스턴스가 생성될 때 자신의 생성자가 호출되면서 수퍼 클래스의 생성자가 연속적으로 자동으로 호출 (이때, 자동 호출되는 생성자는 매개 변수 없는 디폴트 생성자형태이다.)
  - 수퍼 클래스 생성자가 먼저 실행되고 서브 클래스의 생성자가 실행

실 습 하 기 : 상속에서 생성자의 연속적인 호출

```java
package kame.ch07.ex03;
public class Point2D {
    protected int x;
    protected int y;
    public Point2D() {
        System.out.println("2D생성자");
    }
}
```

```java
package kame.ch07.ex03;
public class Point3D extends Point2D {
    protected int z=30;
    public Point3D() {
        System.out.println("3D생성자");
    }
}
```

```java
import kame.ch07.ex03.Point3D;
public class Ch07Ex04 {
    public static void main(String args[]){
        Point3D pt=new Point3D();
    }
}
```

실습:생성자의 오버로딩

```java
package kame.ch07.ex03;

public class Point2D {
    protected int x;
    protected int y;
    public Point2D() {
        System.out.println("2D생성자");
    }
    public Point2D(int x, int y) {
        this.x=x;
        this.y=y;
    }
}
```

```java
package kame.ch07.ex03;
public class Point3D extends Point2D {
    protected int z=30;
    public Point3D() {
        System.out.println("3D생성자");
    }
    public Point3D(int x, int y, int z){
        this.x=x;
        this.y=y;
        this.z=z;
    }
    public void showPoint(){
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);
    }
}
```

```java
import kame.ch07.ex03.Point3D;

public class ch07Ex05 {
    public static void main(String[] args) {
        Point3D pt=new Point3D(100,200,300);
        pt.showPoint();

    }
}
```

3.2 상속 관계에서의 생성자 문제와 해결책

- 디폴트 생성자
  - JVM이 제공해주지만, 클래스 내의 매개변수가 있는 생성자가 하나라도 존재하게 되면 JVM은 더 이상 디폴트 생성자를 제공해 주지 않게 된다.
- 수퍼 클래스에 매개 변수가 있는 생성자를 정의
  - 매개 변수 없는 디폴트 생성자를 정의하지 않으면 수퍼 클래스에는 매개 변수 없는 생성자가 존재하지 않게 되어 문제 발생

```
1  package kame.ch07.ex03;
2
3  public class Point2D {
4      protected int x;
5      protected int y;
6  /*  public Point2D() {
7          System.out.println("2D생성자");
8      }*/
9      public Point2D(int x, int y) {
10         this.x=x;
11         this.y=y;
12     }
13 }
```

```
1  package kame.ch07.ex03;
2  public class Point3D extends Point2D {
3      protected int z=30;
4      public Point3D() {
5          System.out.println("3D생성자");
6      }
7      public Point3D(int x, int y, int z){
8          this.x=x;
9          this.y=y;
10         this.z=z;
11     }
12     public void showPoint(){
13         System.out.println(x);
```

super( )

- 생성자 오버로딩
  - 생성자가 다양하게 제공되어 있을 경우
  - this( ) : 자신의 클래스에서 생성자를 호출
    - this(100,200); this(100,200,300);
  - super() : 수퍼 클래스의 생성자를 호출
    - super(100,200); super(100,200,300);
    - 만일 매개 변수가 있는 수퍼클래스의 생성자를 호출하고 싶은 경우에는 super(매개 변수)라고 호출
- 서브 클래스의 생성자에서, 무엇보다도 수퍼 클래스의 생성자를 제일 먼저 호출

실습:매개변수가 있는 생성자를 명시적으로 호출

```java
package kame.ch07.ex03;

public class Point2D {
    protected int x;
    protected int y;
    public Point2D() {
        System.out.println("2D생성자");
    }
    public Point2D(int x, int y) {
        this.x=x;
        this.y=y;
    }
}
```

```java
package kame.ch07.ex03;
public class Point3D extends Point2D {
    protected int z=30;
    public Point3D() {
        super(0,0);
        System.out.println("3D생성자");
    }
    public Point3D(int x, int y, int z){
        super(x,y);
        this.z=z;
    }
    public void showPoint(){
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);
    }
}
```

```java
import kame.ch07.ex03.Point3D;

public class ch07Ex05 {
    public static void main(String[] args) {
        Point3D pt3=new Point3D(100,200,300);
        pt3.showPoint();

        Point3D pt=new Point3D();
        pt.showPoint();
    }
}
```