클릭하세요 자바

4부 상속

– 16장 상속

최문환

16장 상속

1. 상속성이란
2. 슈퍼 클래스와 서브 클래스
3. 코드를 재활용하는 상속
4. protected 접근 지정자
5. 메서드 오버라이딩
6. 레퍼런스 변수 super
7. 상속에서의 생성자

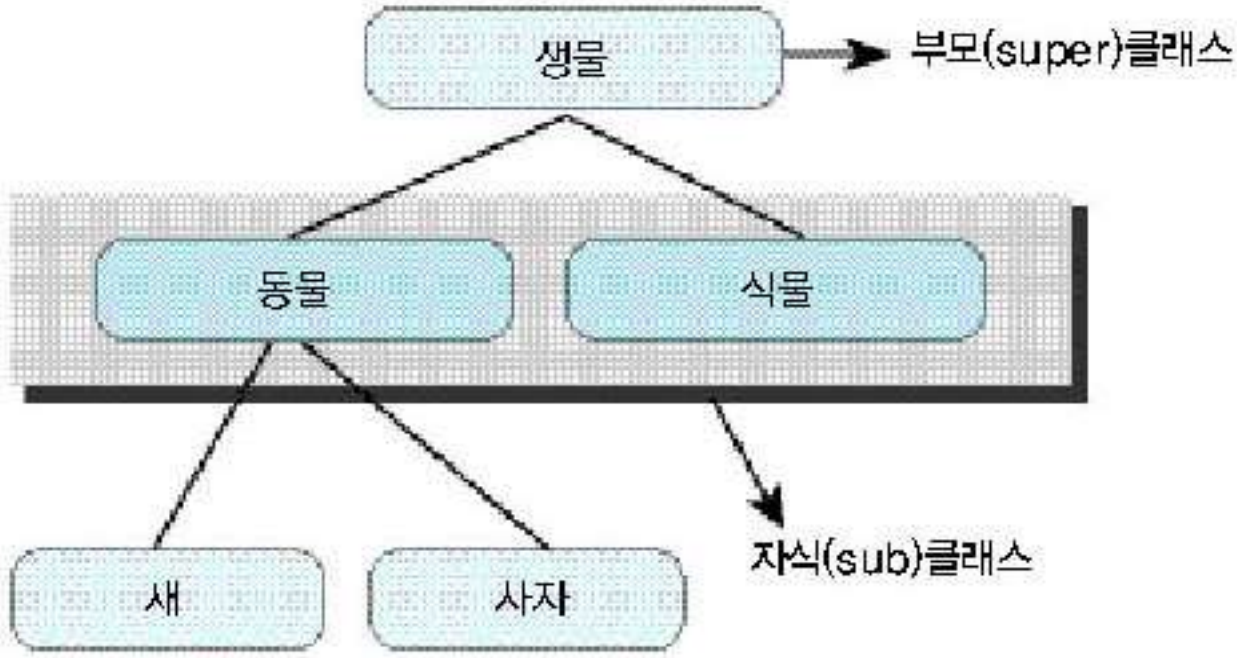
1. 상속성이란

상속이란 자식이 부모가 가지고 있는 재산이나 권력을 물려받는다는 의미이다.

특정(자식) 클래스는 다른(부모) 클래스가 가지고 있는 모든 멤버변수나 멤버함수를 사용할 수 있다.

슈퍼(super) 클래스, 서브(sub) 클래스

1. 상속성이란



일반화
-공통 속성 가짐
-속성이 간단

특수화
-부모 속성 상속
-개별 속성 추가
-속성이 많다

동물(자식) 클래스는 생물(부모) 클래스로부터 파생된 객체로서 생물 클래스가 가지고 있는 모든 속성들을 상속받아 사용할 수 있다.

2. 부모 클래스와 자식 클래스

```
class 자식_클래스 extends 부모_클래스 {


}
```

```
class Parent {
    public void parentPrn( ){


    }
}
```

```
class Child extends Parent {
    public  void childPrn( ){


    }
}
```

<예제> 슈퍼 클래스와 서브 클래스 만들기

```
001:class Parent{
002:  public void parentPrn( ){
003:    System.out.println("슈퍼 클래스 메서드는 상속된다.");
004:  }
005:}
006:
007://Parent를 슈퍼 클래스로 하는 서브 클래스 Child 정의
008:class Child extends Parent{
009:  public void childPrn( ){
010:    System.out.println("서브 클래스 메서드는 슈퍼가 사용 못한다.");
011: }
012:}
013:
014:class SuperSub01{
015:  public static void main(String[] args){
016:    Child c = new Child( );    //서브 클래스로 객체를 생성
017:    c.parentPrn( );            //슈퍼 클래스에서 상속 받은 메서드 호출
018:    c.childPrn( );             //서브 클래스 자기 자신의 메서드  호출
019:    System.out.println("------------------------------>> ");
020     Parent p = new Parent( ); //슈퍼 클래스로 객체 생성
021:    p.parentPrn( );            //슈퍼 클래스 자기 자신의 메서드 호출
022:    //p.childPrn( );           //서브 클래스 메서드는 가져다 사용 못함
023:  }
024:}
```

<예제> 코드의 재활용을 위한 상속

```
001:class Point2D{
002:  private int x;
003:  private int y;
004:  public int getX( ){
005:     return x;
006:  }
007:  public void setX(int new_X){
008:    x=new_X;
009:  }
010:  public int getY( ){
011:     return y;
012:  }
013:  public void setY(int new_Y){
014:    y=new_Y;
015:  }
016:}
```


<예제> 코드의 재활용을 위한 상속

```
017:class Point3D extends Point2D{
018:  private int z;
019:  public int getZ( ){
020:     return z;
021:  }
022:  public void setZ(int new_Z){
023:    z=new_Z;
024:  }
025:}
026:class SuperSub00{
027:  public static void main(String[] args){
028:
029:    Point3D  pt=new Point3D( );
030:    pt.setX(10); //상속받아 사용
031:    pt.setY(20); //상속받아 사용
032:    pt.setZ(30); //자신의 것 사용
033:    System.out.println( pt.getX( )//상속받아 사용
034:              +", "+ pt.getY( )//상속받아 사용
035:              +", "+ pt.getZ( ));//자신의 것 사용
036:  }
037:}
```

4. protected 접근 지정자

```
001:class Point2D{
002:  protected int x=10;     //  private int x=10;
003:  protected int y=20;
004:}
005:class Point3D extends Point2D{
006:  protected int z=30;
007:  public void print( ){
008:    System.out.println(  x  +", "+y+","+z);
009:  }
010:}
011:
012:class SuperSub04{
013:  public static void main(String[] args){
014:    Point3D pt=new Point3D( );
015:    pt.print( );
016:  }
017:}
```

4.1 접근 지정자(제어자)

접근 지정자	같은클래스	같은패키지	다른패키지 자식클래스	다른패키지 전체
private	O	X	X	X
생략(기본접근지정자)	O	O	X	X
protected	O	O	O	X
public	O	O	O	O

5. 메서드 오버라이딩

자손 클래스에서 부모 클래스의 기존 메서드이름, 전달인자의 자료형과 전달인자의 개수,반환타입을 동일하게 정의한다. 부모클래스로 부터 상속받은 메서드 내용을 자식 클래스에 맞게 변경하는 것을 오버라이딩 이라 한다.

자손 클래스에서 메서드는 부모 클래스의 접근 제어보다 더 좁아질 수 없다.

예외는 부모클래스의 메서드 보다 많이 선언할 수 없다.

<예제>-메서드 오버라이딩

```
001:class Parent{
002:  public void parentPrn( ){
003:    System.out.println("슈퍼 클래스 : ParentPrn 메서드");
004:  }
005:}
006://Parent를 슈퍼 클래스로 하는 서브 클래스 Child 정의
007:class Child extends Parent{
008:  //메서드 오버라이딩
010:  public void parentPrn( ){
011:    System.out.println("서브 클래스 : ParentPrn 메서드");
012:  }
013:  public void childPrn( ){
014:    System.out.println("서브 클래스 : ChildPrn 메서드");
015: }
016:}
017:
```

<예제>-메서드 오버라이딩

```
018:class SuperSub05{
019:  public static void main(String[] args){
020:    Child c = new Child( ); //서브 클래스로 객체를 생성
021:    c.parentPrn( );            //오버라이딩된 서브 클래스의 메서드 호출
022:    c.childPrn( );              //서브 클래스 자기 자신의 메서드 호출
023:    System.out.println("------------------------->> ");
024:    Parent p = new Parent( ); //슈퍼 클래스로 객체를 생성
025:    p.parentPrn( );               //슈퍼 클래스(자기 자신)의 메서드 호출
026:  }
027:}
```

6. 레퍼런스 변수 super

```
001:class Parent{
002:  public void parentPrn( ){
003:     System.out.println("슈퍼 클래스 : ParentPrn 메서드");
004:  }
005:}
006://Parent를 슈퍼 클래스로 하는 서브 클래스 Child 정의
007:class Child extends Parent{
008:  //슈퍼 클래스에 있는 ParentPrn 메서드를 오버라이딩
009:  public void parentPrn( ){
010:     super.parentPrn( );  //super로  슈퍼 클래스의 메서드 호출
011:     System.out.println("서브 클래스 : ParentPrn 메서드");
012:  }
013:  public void childPrn( ){
014:     System.out.println("서브 클래스 : ChildPrn 메서드");
015: }
016:}
017:
018:class SuperSub05{
019:  public static void main(String[] args){
020:    Child c = new Child( );  //서브 클래스로 객체를 생성
021:    c.parentPrn( );          //오버라이딩된 서브 클래스의 메서드 호출
022:  }
023:}
```

7. 상속에서의 생성자

```
001:class Point2D{
002:  protected int x=10;
003:  protected int y=20;
004:  public Point2D( ){
005:    System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
006:  }
007:}
008:class Point3D extends Point2D{
009:  protected int z=30;
010:  public void print( ){
011:  System.out.println(x+", "+y+","+z);
012:  }
013:  public Point3D( ){
014:    System.out.println("서브 클래스인 Point3D 생성자 호출");
015:  }
016:}
017:
018:class SuperTest05{
019:  public static void main(String[] args){
020:    Point3D pt=new Point3D( );
021:    pt.print( );
022:  }
023:}
```

7. 상속에서의 생성자

1. 생성자는 상속되지 않는 유일한 멤버함수이다.

2. 서브 클래스의 인스턴스가 생성될 때 자신의 생성자가 호출되면서 슈퍼 클래스의 생성자가 연속적으로 자동으로 호출된다. (이때, 자동 호출되는 생성자는 전달인자 없는 디폴트 생성자 형태이다.)

3. 슈퍼 클래스 생성자가 먼저 실행되고 서브 클래스의 생성자가 실행된다.

7.1 상속관계에서의 생성자 문제

```
001:class Point2D{
002:   protected int x=10;
003:   protected int y=20;
004:   /*
005:   public Point2D( ){
006:      System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
007:   }
008:   */
009:   public Point2D(int xx, int yy){
010:   x=xx; y=yy;
011:   }
012:}
013:class Point3D extends Point2D{
014:   protected int z=30;
015:   public void print( ){
016:   System.out.println(x+", "+y+","+z);
017:   }
018:   public Point3D( ){
019:      System.out.println("서브 클래스인 Point3D 생성자 호출");
020:   }
021:}
022:class SuperTest06{
023:   public static void main(String[] args){
024:      Point3D pt=new Point3D( );
025:      pt.print( );
026:   }
027:}
```

7.1 상속관계에서의 생성자 문제

<예제>-super( )로 슈퍼 클래스의 생성자를 명시적으로 호출

```
001:class Point2D{
002:  protected int x=10;
003:  protected int y=20;
004:  public Point2D( ){
005:    System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
006:  }
007:  public Point2D(int xx, int yy){
008:    x=xx; y=yy;
009:  }
010:}
011:class Point3D extends Point2D{
012:  protected int z=30;
013:  public void print( ){
014:  System.out.println(x+", "+y+", "+z);
015:  }
```

7.1 상속관계에서의 생성자 문제

<예제>-super( )로 슈퍼 클래스의 생성자를 명시적으로 호출

```
016:  public Point3D( ){
017:    super(123, 456);
018:    System.out.println("서브 클래스인 Point3D 생성자 호출");
019:  }
020:  public Point3D(int xx, int yy, int zz){
021:    x=xx; y=yy; z=zz;
022:  }
023:}
024:
025:class SuperTest07{
026:  public static void main(String[] args){
027:    Point3D pt=new Point3D( );
028:    pt.print( );
029:    Point3D pt02=new Point3D(1, 2, 3);
030:    pt02.print( );
031:  }
032:}
```

<문제>

1. 접근 권한이 다음과 같이 지정되어 있을 경우 문제가 발생하는 문장찾기.([ ] 안에 숫자가 기술되어 있다. [ ] 몇 번 문장이 잘못 된 것인지 번호를 기술.(Ex16_2.java)

```
class  Parent {
  private      int  a;
               int  b;
  protected   int  c;
  public       int  d;
}
class Child extends Parent {
  public Child(int a, int b, int c, int d){
     this.a=a;    //[1]
     this.b=b;    //[2]
     this.c=c;    //[3]
     this.d=d;    //[4]
  }
  void func( ){
    System.out.println(a); //[5]
    System.out.println(b); //[6]
    System.out.println(c); //[7]
    System.out.println(d); //[8]
  }
}
```

```
class Ex16_2 {
  public static void main(String[] args){

     Child one=new Child(1, 2, 3, 4);
     one.func( );

     System.out.println(one.a); //[9]
     System.out.println(one.b); //[10]
     System.out.println(one.c); //[11]
     System.out.println(one.d); //[12]
  }
}
```

<문제>

2. 다음과 같은 상속 구조에서 생성자의 호출 순서를 알아보기 위한 문제입니다. 실행 결과를 유추해봅시다.(Ex16_3.java)

```
class  Parent2 {
  protected int  a, b, c;
  public Parent2( ){
    System.out.println("Parnet 클래스의 디폴트 생성자 호출");
  }
  public Parent2(int a, int b, int c){
    System.out.println("Parnet 클래스의 전달인자 3개짜리 생성자 호출");
    this.a=a;  this.b=b;   this.c=c;
  }
}
```

<문제>

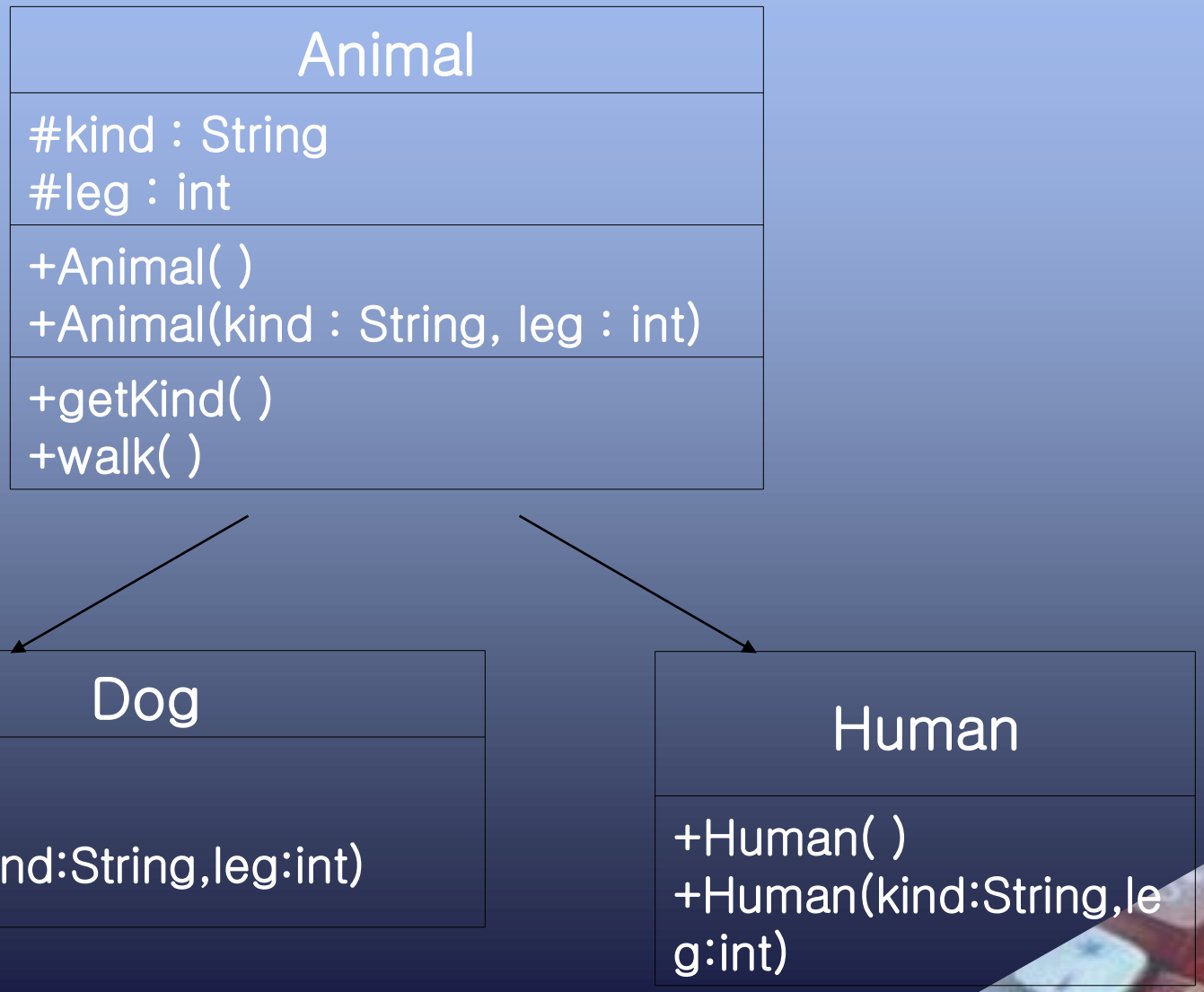
```
class Child2 extends Parent2 {
  public Child2( ){
    System.out.println("Child  클래스의 디폴트 생성자 호출");
  }
  public Child2(int a, int b, int c){
    super(a, b, c);
  System.out.println("Child  클래스의 전달인자 3개짜리 생성자 호출");
  }
}

class Ex16_3 {
  public static void main(String[] args) {
    Child2 one = new Child2( );
    Child2 two = new Child2(10, 20, 30);
  }
}
```

<문제>

3. 동물(Animal)과 개(Dog)와 사람(Human)이란 클래스를 서로 상속이란 개념을 도입해서 설계해 보도록 합시다. 슈퍼 클래스로는 동물(Animal)을 두고, 슈퍼 클래스에는 멤버변수로는 어떤 종인지의 구분을 위해서 kind와 다리의 개수를 저장하기 위한 leg를 둡니다. 또한 슈퍼 클래스의 멤버함수로는 getKind와 walk를 둡니다. getKind 메서드는 어떤 동물인지를 알려주는 메서드이고 walk 메서드로 어떻게 걷는지를 알려주는 메서드입니다.
Animal에서 생성자 오버로딩을 하시고,각 메서드에서 If~else분기문을 사용하세요. 각 자식클래스에서 생성자 오버로딩을 하시고 super();로 부모클래스 오버로딩 된 생성자를 호출하면 됩니다.
슈퍼 클래스 상속 관계로 개(Dog)와 사람(Human) 클래스를 설계해봅시다. (Ex16_04.java)

<문제>


Animal
#kind : String
#leg : int
+Animal( )
+Animal(kind : String, leg : int)
+getKind( )
+walk( )
Dog
+Dog( )
+Dog(kind:String,leg:int)
Human
+Human( )
+Human(kind:String,leg:int)

<문제>

```
class Ex16_04 {
  public static void main(String[] args) {
    Dog d=new Dog(“강아지",4);
   Human h=new Human(“소녀",2);


    d.getKind();
    d.walk();
    h.getKind();
    h.walk();
  }
}
```

<실행 결과>

강아지동물이다
강아지는 4발로 걷는다
소녀사람이다
사람은 2발로 걷는다

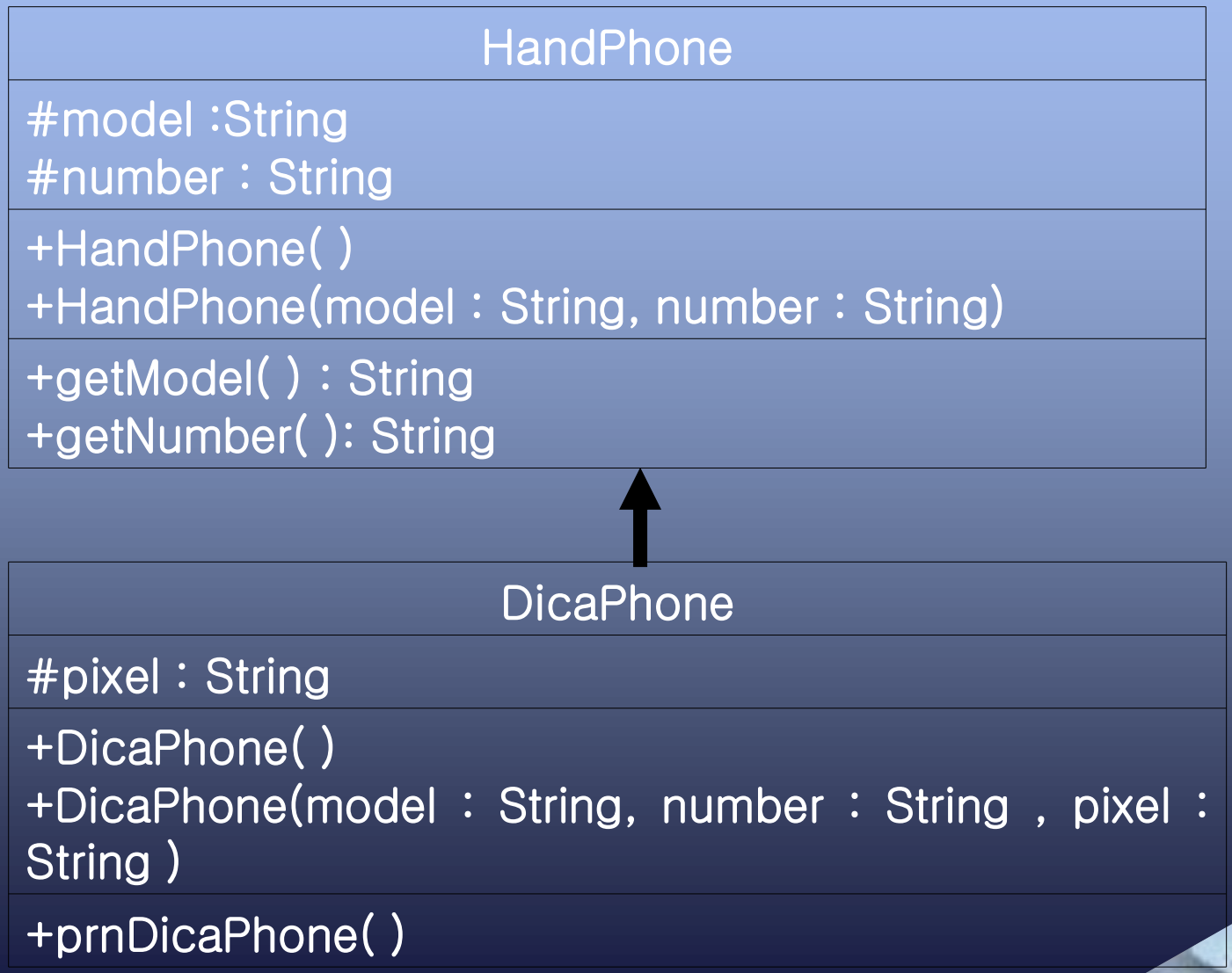
<문제>

4. 다음과 같은 상속 관계를 갖는 두개의 클래스를 설계하시오. 설계된 클래스로 다음과 같이 객체 생성하여 메서드를 호출하면 실행결과와 같이 출력되도록 합시다.

```
class  Ex16_05 {
  public static void main(String[] args){
    DicaPhone dp=new DicaPhone("갤럭시", "010", "1024");
    dp.prnDicaphone( );
  }
 }
```

<실행결과>
모델명 : 갤럭시 번호 : 010 화소수 : 1024

<문제>


HandPhone
#model :String
#number : String
+HandPhone( )
+HandPhone(model : String, number : String)
+getModel( ) : String
+getNumber( ): String
DicaPhone
#pixel : String
+DicaPhone( )
+DicaPhone(model : String, number : String , pixel : String)
+prnDicaPhone( )

<문제>

5. 다음 프로그램의 실행 결과를 유추하시오.

```
001: class TestSuper {
002:   TestSuper(int i) { }
003: }
004: class TestSub extends TestSuper{ }

005:   class Ex16_06 {
006:     public static void main (String [] args) {
007:       new TestSub( );
008:   }
009: }
```

<문제>

6. 다음 프로그램의 실행 결과를 유추하시오.

```
001.  class Base {
002.    Base( ) { System.out.print("Base"); }
003.  }
004.  class Alpha extends Base { }

      public class EX16_07{
005.    public static void main( String[] args ) {
006.      new Alpha( );
007.      new Base( );
008.    }
009.  }
```

<문제>

7. 다음 클래스 중에서 컴파일러로부터 디폴트 생성자를 묵시적으로 제공받는 클래스 2개를 고르시오.

(1)

```
class A {



}
```

(2)

```
class A {
  public A(int
x,int y ) {

 }

}
```

(3)

```
class A {
  public A(int x){

}


}
```

(4)

```
class Z {
}
class A extends Z {
  void a( ) {

  }
}
```

<문제>

8. 다음 프로그램의 실행 결과를 유추하시오.

```
class A {
  public A( ) {
    System.out.println("hello from a");
  }
}
class B extends A {
  public B ( ) {
    System.out.println("hello from b");
    super( );
  }
}
public class Ex16_09{
   public static void main(String args[]) {
   A a = new B( );
    }
 }
```


<문제>

9. 다음 프로그램의 실행 결과를 유추하시오.

```
class A2 {
  public String toString ( ) { return "4"; }
}
class B2 extends A2 {
  public String toString ( ) {
    return super.toString( ) + "3";
  }
}

public class SubEx09 {
  public static void main(String[]args) {
    System.out.println(new B2( ));
  }
}
```