14. 생성자

예제1)

```
class Point2D{
  private int x;
  private int y;
  public int getX( ){
     return x;
  }
  public void setX(int new_X){
    x=new_X;
  }
  public int getY( ){
     return y;
  }
  public void setY(int new_Y){
    y=new_Y;
  }
}
class Point3D extends Point2D{
  private int z;
  public int getZ( ){
     return z;
  }
  public void setZ(int new_Z){
    z=new_Z;
  }
}
class SuperSub00{
  public static void main(String[] args){

    Point3D  pt=new Point3D( );
    pt.setX(10); //상속받아 사용
    pt.setY(20); //상속받아 사용
    pt.setZ(30); //자신의 것 사용
    System.out.println( pt.getX( )     //상속받아 사용
                + ", "+  pt.getY( )     //상속받아 사용
                + ", "+  pt.getZ( ));    //자신의 것 사용
  }
}
```

예제2)

```
class Parent{
  public void parentPrn( ){
    System.out.println("슈퍼 클래스 메서드는 상속된다.");
  }
}

//Parent를 슈퍼 클래스로 하는 서브 클래스 Child 정의
class Child extends Parent{
  public void childPrn( ){
    System.out.println("서브 클래스 메서드는 슈퍼가 사용 못한다.");
 }
}

class SuperSub01{
  public static void main(String[] args){
    Child c = new Child( );     //서브 클래스로 객체를 생성
    c.parentPrn( );             //슈퍼 클래스에서 상속 받은 메서드 호출
    c.childPrn( );               //서브 클래스 자기 자신의 메서드  호출
    System.out.println("------------------------------------>> ");
    Parent p = new Parent( ); //슈퍼 클래스로 객체 생성
    p.parentPrn( );             //슈퍼 클래스 자기 자신의 메서드 호출
    //p.childPrn( );            //서브 클래스 메서드는 가져다 사용 못함
  }
}
```

예제3)

```
class Point2D{
  protected int x=10;      //  private int x=10;
}
class Point3D extends Point2D{
  protected int z=30;
  public void print( ){
    System.out.println(  x  + ", "+ y+ ","+ z);
  }
}

class SuperSub04{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );
  }
}
```

예제4)

```
class Point2D{
  protected int x=10;
  protected int y=20;
}
class Point3D extends Point2D{
  protected int z=30;
  public void print( ){
    System.out.println(  x  +", "+y+", "+z);   //x와 y는 상속 받아 사용하는 멤버변수
  }
}

class SuperTest02{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );
  }
}
```

예제5)

```
class Point2D{
  protected int x=10; //Point3D에서 다시 한번 정의되므로 은닉 변수가 됨
  protected int y=20; //은닉 변수는 쉐도우 변수라고도 함
}
class Point3D extends Point2D{
  protected int x=40; //슈퍼 클래스에 존재하는 멤버변수를
  protected int y=50; //서브 클래스에 다시 한 번 정의함

  protected int z=30;
  public void print( ){
    System.out.println(  x  +", "+y+", "+z);   //x와 y는 재 정의된 Point3D 클래스 소속
  }
}

class SuperTest03{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );
  }
}
```

예제6)

```
class Point2D{
  protected int x=10; //은닉 변수
  protected int y=20; //혹은 쉐도우 변수
}
class Point3D extends Point2D{
  protected int x=40; //슈퍼 클래스에 존재하는 멤버변수를
  protected int y=50; //서브 클래스에 다시 한 번 정의함

  protected int z=30;
  public void print( ){
    System.out.println(  x  + ", "+ y+ ", "+ z);    //x와 y는 재 정의된 Point3D 클래스 소속
  }
  public void print02( ){
    System.out.println(super.x+ ", "+ super.y+ ", "+ z); //Point2D 클래스 소속 멤버변수로 접근
  }
}
class SuperTest04{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );    //40, 50, 30 // Point3D의 x, y
    pt.print02( ); //10, 20, 30 // Point2D의 x, y
  }
}
```

예제7)

```
class Point2D{
  protected int x=10;
  protected int y=20;

  public Point2D( ){
    System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
  }
}

class Point3D extends Point2D{
  protected int z=30;
  public void print( ){
  System.out.println(x+ ",  "+ y+ ","+ z);
  }
  public Point3D( ){
    System.out.println("서브 클래스인 Point3D 생성자 호출");
  }
}

class SuperTest05{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );
  }
}
```

예제8)

```
class Point2D{
  protected int x=10;
  protected int y=20;

  /*
  public Point2D( ){
    System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
  }
  */
  public Point2D(int xx, int yy){
  x=xx; y=yy;
  }
}

class Point3D extends Point2D{
  protected int z=30;

  public void print( ){
  System.out.println(x+ ",  "+ y+ ","+ z);
  }
  public Point3D( ){
    System.out.println("서브 클래스인 Point3D 생성자 호출");
  }
}

class SuperTest06{
  public static void main(String[] args){
     Point3D pt=new Point3D( );
    pt.print( );
  }
}
```

예제9)

```
class Point2D{
  protected int x=10;
  protected int y=20;

  public Point2D( ){
    System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
  }
  public Point2D(int xx, int yy){
    x=xx; y=yy;
  }
}

class Point3D extends Point2D{
  protected int z=30;
  public void print( ){
  System.out.println(x+ ", "+ y+ ", "+ z);
  }
  public Point3D( ){
    super(123, 456);
    System.out.println("서브 클래스인 Point3D 생성자 호출");
  }
  public Point3D(int xx, int yy, int zz){
    x=xx; y=yy; z=zz;
  }
}

class SuperTest07{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );
    Point3D pt02=new Point3D(1, 2, 3);
    pt02.print( );
  }
}
```

예제10)

```
class Point2D{
  protected int x=10;
  protected int y=20;

  public Point2D( ){
    System.out.println("슈퍼 클래스인 Point2D 생성자 호출");
  }
  public Point2D(int xx, int yy){
    x=xx; y=yy;
  }
}

class Point3D extends Point2D{
  protected int z=30;

  public void print( ){
  System.out.println(x+ ", "+ y+ ", "+ z);
  }

  public Point3D( ){
    super(123, 456);
    System.out.println("서브 클래스인 Point3D 생성자 호출");
  }

  public Point3D(int xx, int yy, int zz){
    x=xx; y=yy; z=zz;
  }
}

class SuperTest07{
  public static void main(String[] args){
    Point3D pt=new Point3D( );
    pt.print( );
    Point3D pt02=new Point3D(1, 2, 3);
    pt02.print( );
  }
}
```

예제11)

```
package packTest.packOne;
public class  AccessTest {  //다른 패키지에서 가져다 사용할 것임으로 pubic으로
  private int     a=10;    //[1] private
  int             b=20;   //[2] 기본 접근 지정자
  protected  int c=30;   //[3] protected
  public      int d=40;   //[4] public

  public void print( ){
    System.out.println("AccessTest의 print");
    System.out.println(a);
    System.out.println(b);
    System.out.println(c);
    System.out.println(d);
  }
}
```

```
import  packTest.packOne.AccessTest;

//AccessTest의 서브 클래스로 SubOne을 설계
class SubOne extends AccessTest {
  void subPrn( ){
    System.out.println(a); //[1. Sub] private -X
    System.out.println(b); //[2. Sub] 기본 접근 지정자-X
    System.out.println(c); //[3. Sub] protected -O
    System.out.println(d); //[4. Sub] public -0
  }
}

//AccessTest랑 상속관계가 없는 클래스
class SuperSubA{
  public static void main(String[] args){
    AccessTest at=new AccessTest( );
    at.print( );
    System.out.println("main");
    System.out.println(at.a); //[1. main] private -X
    System.out.println(at.b); //[2. main] 기본 접근 지정자-X
    System.out.println(at.c); //[3. main] protected -X
    System.out.println(at.d); //[4. main] public -O
  }
}
```

15. 참조 (레퍼런스)

예제12)

```
class Parent{
  public void parentPrn( ){
     System.out.println("슈퍼 클래스 : ParentPrn 메서드");
  }
}
//Parent를 슈퍼 클래스로 하는 서브 클래스 정의
class Child extends Parent{
  public void childPrn( ){
     System.out.println("서브 클래스 : ChildPrn 메서드");
 }
}

class RefTest01{
  public static void main(String[] args){
    Child c = new Child();  //서브 클래스로 객체를 생성

    //Child 객체로 접근해서 호출할 수 있는 메서드는 2개
    c.parentPrn();    //부모로부터 상속 받은 메서드
    c.childPrn();      //자신의 메서드

    Parent p;         //슈퍼 클래스형 레퍼런스 변수 선언
    p=c;              //암시적으로 업 캐스팅이 일어남
    p.parentPrn(); //업 캐스팅 후에는 부모로부터 상속받은 메서드만 호출할 수 있다,
    //p.childPrn(); //컴파일 에러가 발생하게 된다.
  }
}
```

예제13)

```
class RefTest02{
  public static void main(String[] args){
  Parent  p = new  Parent();  //슈퍼 클래스로 인스턴스 선언
  Child  c;                   //서브 클래스로 레퍼런스 변수 선언

  //서브 클래스형 레퍼런스 변수에 슈퍼 클래스의 레퍼런스 값을 대입하면
  c= p;   //이를 DownCasting 이라하는데 컴파일러 에러가 발생한다.
 }
}
```

예제14)

```
 class RefTest03{
  public static void main(String[] args){
  Parent  p = new  Child( );  //서브 클래스로 인스턴스 선언

  p.parentPrn();

 //p.childPrn();//-컴파일 에러

  Child  c;                    //서브 클래스로 레퍼런스 변수 선언

  System.out.println("----------------------------->>");
  //서브 클래스 레퍼런스 변수에 슈퍼 클래스의 레퍼런스 값이 대입됨
  c = (Child) p;   //강제 형변환으로 다운 캐스팅
  c.parentPrn();
  c.childPrn();
 }
}
```

예제15)

```
class RefTest04{
  public static void main(String[] args){
  Parent  p = new  Parent( );  //슈퍼 클래스로 인스턴스 선언
  Child  c;                    //서브 클래스로 레퍼런스 변수 선언

  //서브 클래스 레퍼런스 변수에 슈퍼 클래스의 레퍼런스 값이 대입됨
  c = (Child) p;   //강제 형변환으로 다운 캐스팅
  c.parentPrn();
  c.childPrn();
 }
}
```

예제16)

```
class HandPhone {
  String model; //모델명
  String number;//전화번호

  public HandPhone(){
  }

  public HandPhone(String m, String n){
    model=m;
    number=n;
  }
}

class DicaPhone extends HandPhone{
  String pixel; //화소수

  public  DicaPhone( ){
  }
  public  DicaPhone(String m, String n, String p){
    super(m, n);
    pixel=p;
  }
  public void prnDicaPhone(){
      System.out.println(model+ ":"+ number+ ":"+ pixel);
  }
}

class RefTest05 {
  public static void main(String[] args){
    DicaPhone dp=new DicaPhone("에버-5500","010-3346-1980","256");
    // instanceof 는 좌변의 객체가 우변의 자손인지를 판별해 주는 연산자
    dp.prnDicaPhone();
    System.out.println("-----------------------");

    if (dp instanceof DicaPhone){ //dp는 DicaPhone입니까?
      System.out.println("dp는 DicaPhone이다.");
    }

    if (dp instanceof HandPhone){  //dp는 HandPhone입니까?
      System.out.println("dp는 HandPhone 이다.");
      HandPhone hh=dp;
      System.out.println("그러므로 HandPhone으로 형변환 가능함");
      System.out.println("dp는 HandPhone이 가진 기능을 모두 다 포함");
    }
```

```
    System.out.println("-----------------------");

    HandPhone hp = new HandPhone( );

    if (hp instanceof DicaPhone){ //p는 DicaPhone입니까?
      System.out.println("hp는 DicaPhone 이다.");
    }else{
      System.out.println("hp는 DicaPhone이 아니다.");
      System.out.println("그러므로 DicaPhone으로 형변환 불가능함.");
      System.out.println("hp는 DicaPhone이 가진 기능을 모두 다 포함하지 못함");
    }
  }
}
```

예제17)

```
class Parent{
  public void parentPrn( ){
     System.out.println("슈퍼 클래스 : ParentPrn 메서드");
  }
}
//Parent를 슈퍼 클래스로 하는 서브 클래스 정의
class Child extends Parent{
  public void parentPrn( ){
     System.out.println("서브 클래스 : 오버라이딩된 ParentPrn 메서드");
  }

  public void childPrn( ){
     System.out.println("서브 클래스 : ChildPrn 메서드");
 }
}

class RefTest06{
  public static void main(String[] args){
    Child c = new Child();  //서브 클래스로 객체를 생성
    //레퍼런스와 객체의 자료형이 동일하면
    c.parentPrn();    //자신의 기능들이 호출된다.
    Parent p;         //슈퍼 클래스형 레퍼런스 변수 선언

    //슈퍼 클래스형 레퍼런스 변수가 서브 클래스형 객체를 가리킴
    p=c;     //업 캐스팅
    //업 캐스팅 후에도 슈퍼 클래스의 parentPrn 메서드는 은닉되고
    p.parentPrn();  //서브 클래스에 오버라이딩된 메서드가 호출된다.
   }
}
```

16. 추상클래스(abstract)와 final

예제18)

```
class AbstractClass{//추상 클래스가 아닌 클래스에서
    abstract void Method01();//추상 메서드를 가질 경우 컴파일 에러
}

public class AbstractTest00 {

    public AbstractTest00() {
    }

    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

예제19)

```
interface IHello{
  void sayHello(String name);
}
class Hello implements IHello{
  //public void sayHello(String name){
            void sayHello(String name){
      System.out.println(name+ "씨 안녕하세요!");
   }
}
class AbstractTest01{
  public static void main(String[] args) {
    Hello obj= new Hello();
    obj.sayHello(args[0]);
  }
}
```

예제20)

```
abstract class Hello{
  public abstract void sayHello(String name);
}

abstract class GoodBye{
  public abstract void sayGoodBye(String name);
}

//class SubClass extends GoodBye{
class SubClass extends Hello{//추상 클래스 Hello를 상속
  public void sayHello(String name){//오버라이딩 한 것이
    System.out.println(name+"씨 안녕하세요!");
  }
  public void sayGoodBye(String name){      //추상 클래
    System.out.println(name+"씨 안녕히 가세요!");
  }
}

class  AbstractTest02{
   public static void main(String[] args) {
     SubClass test= new SubClass();
     test.sayHello(args[0]);
     test.sayGoodBye(args[0]);
   }
}
```

예제21)

```
abstract class Hello{
  public abstract void sayHello(String name);
}

abstract class GoodBye{
  public abstract void sayGoodBye(String name);
}

class SubClass extends GoodBye, Hello {
  public void sayHello(String name){
       System.out.println(name+ "씨 안녕하세요!");
  }
    public void sayGoodBye(String name){
       System.out.println(name+ "씨 안녕히 가세요!");
  }
}

class AbstractTest03{
   public static void main(String[] args) {
      SubClass test= new SubClass();
      test.sayHello(args[0]);
      test.sayGoodBye(args[0]);
   }
}
```

예제22)

```
interface IHello{
  public abstract void sayHello(String name);
}
interface IGoodBye{
  public abstract void sayGoodBye(String name);
}
interface ITotal extends IHello, IGoodBye{
  public abstract void greeting(String name);
}

class SubClass implements ITotal{
  public void sayHello(String name){
    System.out.println(name+"씨 안녕하세요!");
  }
  public void sayGoodBye(String name){
    System.out.println(name+"씨 안녕히 가세요!");
  }
  public void greeting(String name){
    System.out.println(name + ", 안녕!");
  }
}

class AbstractTest06{
  public static void main(String[] args) {
    SubClass test= new SubClass();
    test.sayHello(args[0]);
    test.sayGoodBye(args[0]);
    test.greeting(args[0]);
  }
}
```

예제23)

```
class AbstractClass{//추상 클래스가 아닌 클래스에서
    abstract void Method01();//추상 메서드를 가질 경우 컴파일 에러

}
class AbstractTestA{
  public static void main(String[] args){
    AbstractClass  obj= new AbstractClass();
    //추상클래스는 자생력이 없으므로 객체 생성을 하지 못함
  }
}
```

예제24)

```
public class AbstractTestB {

    public AbstractTestB() {
    }

    public static void main(String[] args) {
        // TODO code application logic here
       SubClass obj=new SubClass( );
       obj.Method01( );   //서브 클래스에서 추상 메서드를 오버라이딩하였기에 호출 가능
       obj.Method02( );   //추상 클래스에서 구현된 메서드를 상속 받아 호출
       obj.Method03( );   //추상 클래스에서 구현된 메서드를 상속 받아 호출
    }
}
```

예제25)

```
class FinalMember {
  final int a=10;

  public void setA(int a){
  //  this.a=a;
  }
}
public class FinalTest01{
  public static void main(String[] args) {
    FinalMember ft= new FinalMember();
    final int a=1000;

    ft.setA(100);
    System.out.println(a);
  }
}
```

예제26)

```
class FinalMethod{
  String str="Java ";

  //public void setStr(String s) {
  //final 붙이면 서브 클래스에서 오버라이딩이 불가.

  public final void setStr(String s) {
    str=s;
    System.out.println(str);
  }
}

class FinalEx extends FinalMethod{
  int a=10; // final 붙이면 밑에서 a값 대입 불가.

  public void setA(int a) {
    this.a=a;
   }

  public void setStr(String s) {
    str+=s;
    System.out.println(str);
  }
}

public class FinalTest02{
  public static void main(String[] args) {
    FinalEx  ft= new FinalEx( );

    ft.setA(100);
    ft.setStr("hi");// 슈퍼 클래스의 setStr을 실행.
    FinalMethod ft1=new FinalMethod( );
    ft1.setStr("hi");// 자신의 클래스의 setStr을 실행.
  }
}
```

예제27)

```
final class FinalClass{
  String str="Java ";
    public void setStr(String s){
      str=s;
    System.out.println(str);
  }
}

class FinalEx extends FinalClass{
   int a=10;
   public void setA(int a)  {
    this.a=a;
   }
   public void setStr(String s){
    str+=s;
    System.out.println(str);
   }
}

public class FinalTest03{
   public static void main(String[] args) {
    FinalEx fe= new FinalEx( );
   }
}
```

17. 인터페이스

예제28)

```
abstract class Hello{
  public abstract void sayHello(String name);
}

abstract class GoodBye{
  public abstract void sayGoodBye(String name);
}

//class SubClass extends GoodBye{
class SubClass extends Hello{//추상 클래스 Hello를 상속
  public void sayHello(String name){//오버라이딩 한 것이
    System.out.println(name+"씨 안녕하세요!");
  }
  public void sayGoodBye(String name){      //추상 클래
    System.out.println(name+"씨 안녕히 가세요!");
  }
}

class  AbstractTest02{
   public static void main(String[] args) {
     SubClass test= new SubClass();
     test.sayHello(args[0]);
     test.sayGoodBye(args[0]);
   }
}
```

예제29)

```
abstract class Hello{
  public abstract void sayHello(String name);
}

abstract class GoodBye{
  public abstract void sayGoodBye(String name);
}

class SubClass extends GoodBye, Hello {
  public void sayHello(String name){
       System.out.println(name+ "씨 안녕하세요!");
  }
    public void sayGoodBye(String name){
       System.out.println(name+ "씨 안녕히 가세요!");
  }
}

class AbstractTest03{
   public static void main(String[] args) {
      SubClass test= new SubClass();
      test.sayHello(args[0]);
      test.sayGoodBye(args[0]);
   }
}
```

예제30)

```
interface IHello{
  public abstract void sayHello(String name);
}

abstract class GoodBye{
  public abstract void sayGoodBye(String name);
}

class SubClass extends GoodBye implements IHello{
  public void sayHello(String name){
    System.out.println(name+ "씨 안녕하세요!");
  }
  public void sayGoodBye(String name){
    System.out.println(name+ "씨 안녕히 가세요!");
  }
}

class AbstractTest04{
  public static void main(String[] args) {
    SubClass test= new SubClass();
    test.sayHello(args[0]);
    test.sayGoodBye(args[0]);
  }
}
```

예제31)

```
 interface IHello{
   public abstract void sayHello(String name);
}

interface IGoodBye{
   public abstract void sayGoodBye(String name);
}

//두 인터페이스로부터 상속을 받는 클래스 설계
class SubClass implements IHello, IGoodBye{
   public void sayHello(String name){
       System.out.println(name+"씨 안녕하세요!");
   }
   public void sayGoodBye(String name){
       System.out.println(name+"씨 안녕히 가세요!");
   }
}

class AbstractTest05{
   public static void main(String[] args) {
      SubClass test= new SubClass();
      test.sayHello(args[0]);
      test.sayGoodBye(args[0]);
   }
}
```

예제32)

```
interface IHello{
  public abstract void sayHello(String name);
}

interface IGoodBye{
  public abstract void sayGoodBye(String name);
}

interface ITotal extends IHello, IGoodBye{
  public abstract void greeting(String name);
}

class SubClass implements ITotal{
  public void sayHello(String name){
    System.out.println(name+"씨 안녕하세요!");
  }
  public void sayGoodBye(String name){
    System.out.println(name+"씨 안녕가세요!");
  }
  public void greeting(String name){
    System.out.println(name + ", 안녕!");
  }
}

class AbstractTest06{
  public static void main(String[] args) {
    SubClass test= new SubClass();
    test.sayHello(args[0]);
    test.sayGoodBye(args[0]);
    test.greeting(args[0]);
  }
}
```

예제33)

```
 //인터페이스(IColor)므로 다중 상속 가능
interface IColor{
  int RED=1;                              //상수(public static final 로 인식)
  public static final int GREEN=2;        //상수
  public static final int BLUE=3;         //상수
  void setColor(int c);                   //추상메서드 (public abstract 로 인식)
  public abstract int getColor();         //추상메서드
}

//클래스(AbsColor)이므로 다중 상속 불가능 단일 상속만,
abstract class AbsColor implements IColor{
  int color=GREEN;                        //변수도 가질 수 있디.
  public void setColor(int c){            //구현된 메서드도 가질 수 있다.
    color=c;
  }
}

class SubClass extends AbsColor{
  public int getColor(){
    return color;
  }
}

class AbstractTest07{
  public static void main(String[] args) {
    SubClass test= new SubClass();
    test.setColor(IColor.RED);
    System.out.println(test.getColor());
  }
}
```

예제34)

```
interface IHello{
  void sayHello(String name);
}

class Hello implements IHello{
  public void sayHello(String name){
  //void sayHello(String name){
       System.out.println(name+"씨 안녕하세요!");
   }
}

class InterfaceTest01{
  public static void main(String[] args) {
    Hello obj= new Hello();
    obj.sayHello(args[0]);//윤정
  }
}
```

예제35)

```
 interface IHello{
  public abstract void sayHello(String name);
}

interface IGoodBye{
  public abstract void sayGoodBye(String name);
}

//두 인터페이스로부터 상속을 받는 클래스 설계
class SubClass implements IHello, IGoodBye{
  public void sayHello(String name){
     System.out.println(name+"씨 안녕하세요!");
  }
  public void sayGoodBye(String name){
     System.out.println(name+"씨 안녕히 가세요!");
  }
}
class AbstractTest05{
  public static void main(String[] args) {
    SubClass test= new SubClass();
    test.sayHello(args[0]);
    test.sayGoodBye(args[0]);
  }
}
```