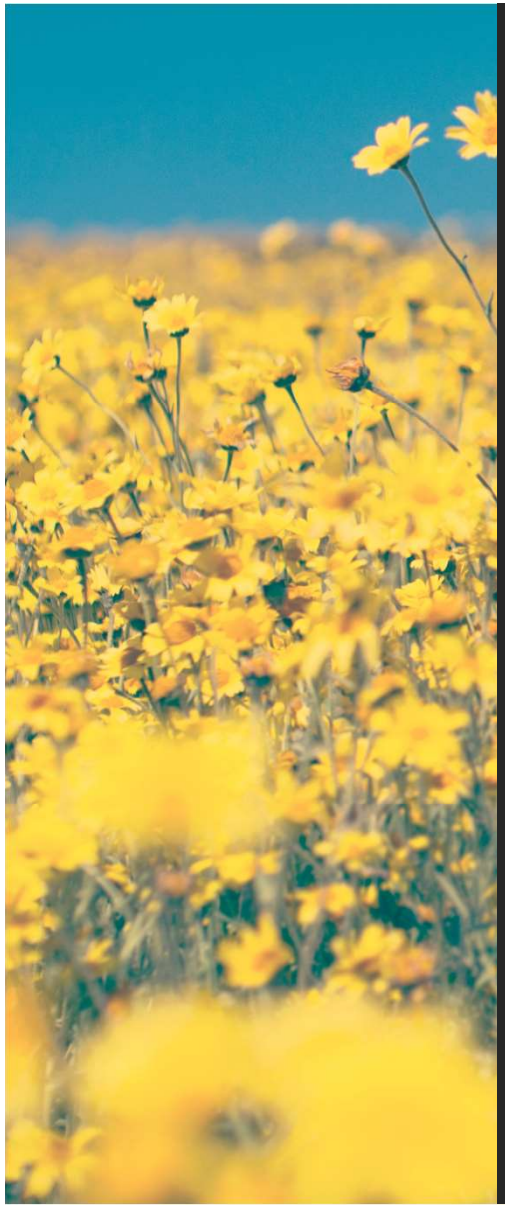

JAVA

GU, DA HAE


상 속
생 성 자
protected
오 버 라 이 딩
instanceof
abstract
interface
inner class
JAVA
객
체

JAVA

Object 클래스

자바의 **모든 클래스가 상속받는 최상위 클래스**다. java.lang 패키지에 선언되어 있다.

```
class 식별자(extends Object) {}
```

클래스를 정의할 때 아무런 클래스도 상속받지 않으면 자동으로 extends Object 클래스를 상속받는다.

JAVA

Object 주요 메서드

Modifier and Type	Method	Description
protected Object	clone()	현 인스턴스와 똑같은 인스턴스를 생성하여 반환.
boolean	equals(Object obj)	obj가 참조하는 인스턴스와 현 인스턴스를 비교하여 같으면 true를 반환.
protected void	finalize()	**Deprecated.** 인스턴스가 소멸되기 직전에 JVM에 의해 호출되어 메모리 해제.
Class<?>	getClass()	현 인스턴스의 runtime 클래스를 반환.
int	hashCode()	현 인스턴스의 해시 코드를 반환.
void	notify()	현 인스턴스의 대기중인 쓰레드 하나를 깨움.
void	notifyAll()	현 인스턴스의 대기중인 쓰레드 전부를 깨움.
String	toString()	현 인스턴스의 문자열 표현을 반환.
void	wait()	현 인스턴스의 다른 쓰레드가 notify() 또는 notifyAll 메서드를 호출할 때까지 현 쓰레드를 대기시킴.
void	wait(long timeout)	현 인스턴스의 다른 쓰레드가 notify() 또는 notifyAll() 메서드를 호출할 때까지, 또는 지정한 현실 시간동안 현 쓰레드를 대기시킴.
void	wait(long timeout, int nanos)	현 인스턴스의 다른 쓰레드가 notify() 또는 notifyAll() 메서드를 호출할 때까지, 또는 지정한 현실 시간동안 현 쓰레드를 대기시킴.

JAVA

final 클래스

final 키워드는 자바에서 상수를 선언할 때 사용한다. 클래스를 정의 할 때 final을 사용하면 해당 클래스는 상속할 수 없다는 뜻으로 사용된다.

```
final class 식별자{
    ...
}
```

JAVA

final 메서드

final 키워드는 자바에서 상수를 선언할 때 사용한다. 메서드를 정의 할 때 final을 사용하면 해당 메서드는 오버라이딩이 불가능하다.

```
class 식별자{
    final void test(){…}
}
```

JAVA

abstract 메서드

추상(abstract) 메서드는 선언만 되어있는 메서드를 말한다.

```
public abstract void test();
```

메서드의 정의가 필요 없을 경우, abstract 키워드를 적고 선언만 한다.

JAVA

abstract 클래스

추상(abstract) 클래스는 **인스턴스 생성이 불가능**한 클래스다. 추상 클래스는 인스턴스 생성이 불가능하다는 점을 제외하면 모든 부분이 일반 클래스와 같다.

```
abstract class 식별자{
    ...
    public abstract void test();
}
```

인스턴스 생성이 아닌, **상속을 정의하기 위해서만** 사용하는 상위 클래스다.

추상 메서드가 포함되어 있다면 반드시 **클래스를 추상화** 해야 한다.

추상 클래스를 상속할 경우 반드시 추상 메서드를 재정의해야 한다.

JAVA

예제

```java
// 상속을 정의하기 위해서만 사용
// 인스턴스를 생성할 일이 없음
class Worker{
    private String name;
    public Worker(String name){
        this.name = name;
    }
    public void showName(){
        System.out.println("Name : " + name);
    }
    // 메서드 오버라이딩을 위해 정의한 메서드
     // Worker에서 딱히 할 일도 없고, 필요한 메서드가 아니다.
     public int getPay(){return 0;}
     public void showSalaryInfo(){}
}
```



```java
// 추상화 메서드가 포함되어 있다면 반드시 추상화
abstract class Worker{
    private String name;

    public Worker(String name){
        this.name = name;
    }
    public void showName(){
        System.out.println("Name : " + name);
    }
    // 추상화
    abstract public int getPay();
    abstract public void showSalaryInfo();
}
```

JAVA

예제

```
abstract class A{
     public void m1(){System.out.println("m1!!");}
     public abstract void m2();
}

// 추상 클래스를 상속하면 반드시 하위 클래스에서 추상 클래스의 추상 메서드를 재정의해야 한다.
class B extends A{
     public void m2(){System.out.println("m2!!");}
}
```

JAVA

interface

자바는 다중상속(multiple inheritance)를 지원하지 않는다. 인터페이스는 다중상속과 비슷한 효과를 주기위해서 사용한다.

```
interface 식별자{
    ...
}
```

인터페이스는 모든 멤버가 추상화된 추상 클래스의 한 종류다.

인터페이스의 멤버 변수는 무조건 public static final로 선언된다.

인터페이스의 메서드는 무조건 public abstract로 선언된다.

인터페이스를 상속할 때는 extends가 아닌 implements를 사용한다.

인터페이스를 상속할 때는 상속이 아닌 '구현한다'라고 표현한다.

JAVA

예제

```
interface TestInterface{
     void method(); // 자동으로 public abstract 선언한다.
}

class TestClass implements TestInterface{
     public void method(){System.out.println("HI");}
}

public class Ex01{
     public static void main(String[] args){
          TestClass c = new TestClass();
          c.method();
     }
}
```

JAVA

예제

```
interface TestInterface1{void method1();}

interface TestInterface2{void method2();}


class TestClass implements TestInterface1, TestInterface2{// 다중 구현이 가능하다.
     public void method1(){System.out.println("HI");}
     public void method2(){System.out.println("Hello");}
}

public class Ex01{
     public static void main(String[] args){
          TestClass c = new TestClass();
          c.method1();
          c.method2();
     }
}
```

JAVA

예제

```
interface Week{ // 상수 선언에 활용
    // 자동으로 public static final 선언한다.
    int MON = 1, TUE = 2, WED = 3, THU = 4;
    int FRI = 5, SAT = 6, SUN = 7;
}
public class Ex01{
      public static void main(String[] args){
          System.out.println("일정");
          System.out.println("1.월 | 2.화 | 3.수 | 4.목 | 5.금
      | 6.토 | 7.일");
          System.out.print("선택 : ");

          Scanner sc = new Scanner(System.in);
          int menu = sc.nextInt();

          switch(menu) {
          case Week.MON :
              System.out.println("자료구조 스터디");
              break;
          case Week.TUE :
              System.out.println("공학수학 레포드 마감");
              break;
          case Week.WED :
              System.out.println("동아리 회의");
              break;
          case Week.THU :
              System.out.println("알고리즘 스터디");
              break;
          case Week.FRI :
              System.out.println("불금");
              break;
          case Week.SAT:
          case Week.SUN:
              System.out.println("휴일");
          }
      }
}
```

JAVA

해보기

도형을 표현한 Shape 클래스를 기반으로 Circle, Rectangle, Triangle 클래스를 정의해보자.
Circle은 반지름, Rectangle은 가로와 세로, Triangle은 높이와 밑변 길이를 저장할 수 있어야한다.
상속과 오버라이딩, interface를 활용하는 방향으로 해결해보자.

모든 도형은 반드시 넓이를 출력하는 기능, 저장하고 있는 데이터를 출력하는 기능이 필요하다.

JAVA

해보기

Printer 클래스를 기반으로 InkjetPrinter와 RaserPrinter 클래스를 정의해보자. 모든 프린터는 모델명, 제조사, 인터페이스 종류(USB, 블루투스, 지그비 등), 인쇄 매수, 인쇄 종이 잔량을 나타내는 정보를 가지며 printInfo()메서드를 갖는다. 잉크젯 프린터는 잉크 잔량을 표기해야 하며, 레이저 프린터는 토너 잔량을 표기해야 한다.