

# 프로그래밍 언어 응용

chapter11

## 기본클래스

제공된 자료는 훈련생의 수업을 돕기 위한 것으로, 타인과 공유하시면 안됩니다.

# Contents

part.1

Object 클래스

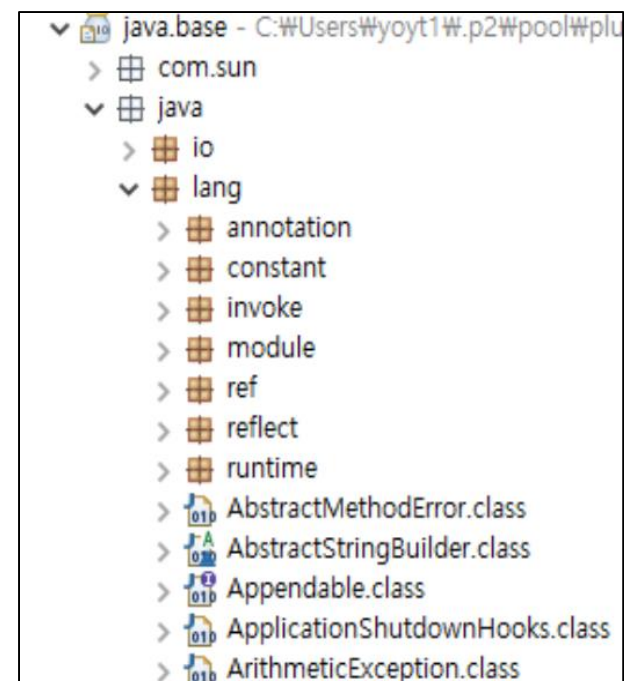
part.2

String 클래스

part.3

Wrapper 클래스

자바프로그램에서 자주 사용되는 클래스로는 String, Object, System 등이 있다.  
이 클래스들은 모두 java.lang 패키지에 포함이 되어있다.  
java.lang 패키지는 자바의 핵심 라이브러리로, 프로그램에서 자동으로 import된다.



자바 라이브러리

```
//import java.lang.*;  
  
public class Ex1 {  
    String str;
```

자바 프로그램

# Object 클래스

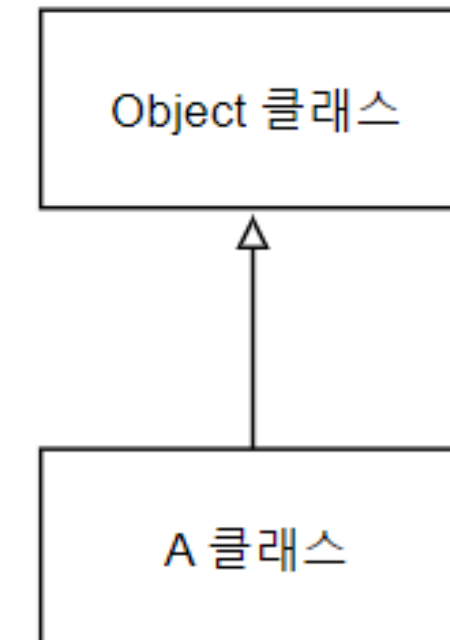
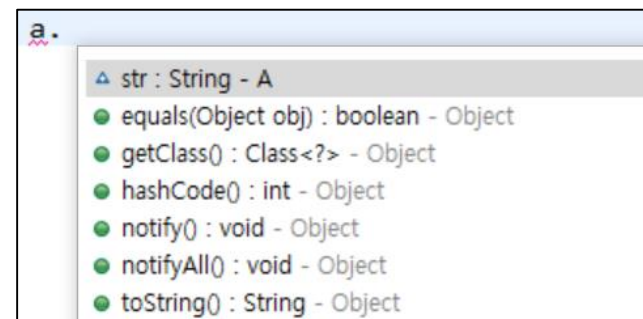
## 최상위 클래스 Object

### Object 클래스란?

- Object 클래스는 모든 클래스의 최상위 클래스이다.
- 모든 자바 클래스는 자동으로 Object 클래스를 상속받는다.

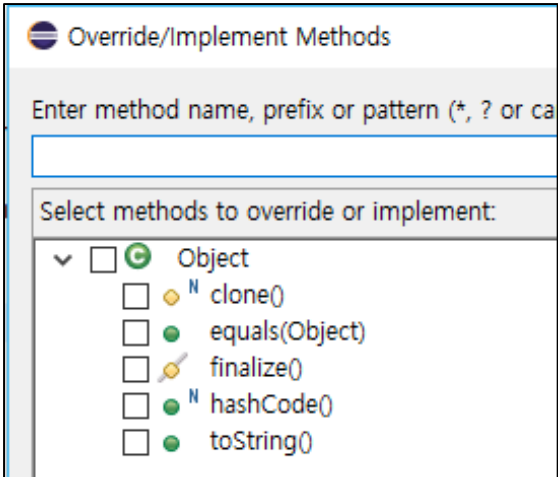
public class A { }    ➡    public class A **extends Object** { }

### A 클래스의 메소드



Object 클래스에서 자주 사용하는 메소드는 아래와 같다.  
equals()와 toString() 메소드는 주로 재정의하여 사용한다.

메소드	설명
Object clone( )	객체를 복사하여 새로운 인스턴스를 생성한다
boolean equals(Object obj)	객체 자신과 obj객체가 같은 객체인지 비교하여 결과를 반환한다
Class getClass()	클래스 설계 정보를 반환한다
int hashCode( )	메모리에 저장된 인스턴스 주소값을 반환한다
String toString( )	객체의 정보를 문자열로 반환한다



재정의 가능한 메소드

# Object 클래스

## equals 메소드

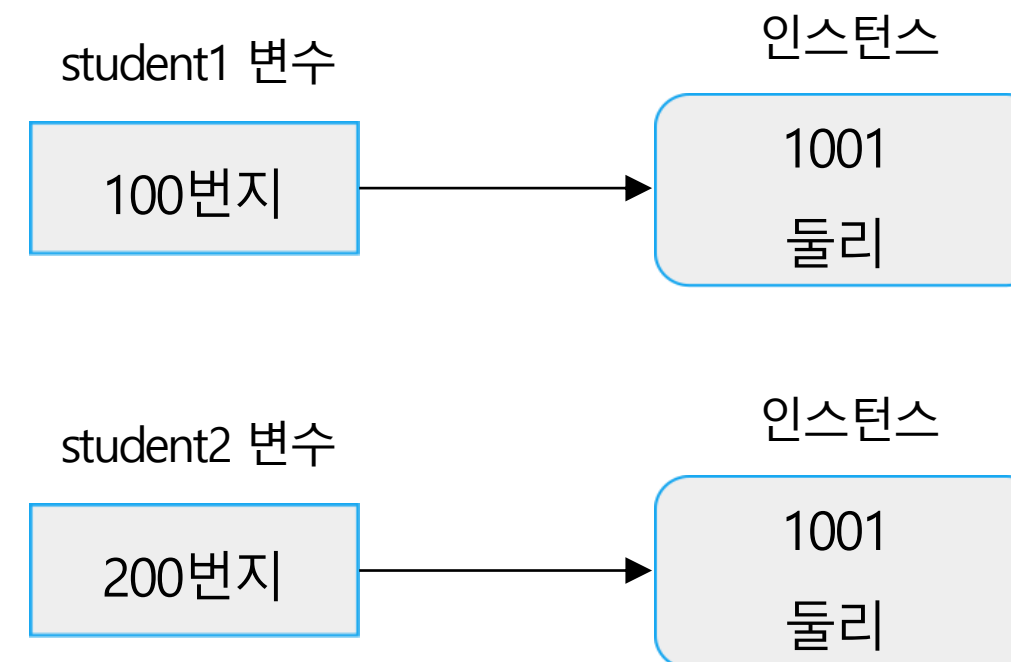
### equals 메소드

- equals 메소드는 두 객체가 같은지 비교한다.
- equals 메소드의 원형은 같은 객체인지 확인하기 위해 메모리 주소를 비교한다.
- 객체의 내용이 같더라도 메모리 주소가 다르면 두 객체는 다르다고 판단한다.

Object클래스의 equals 메소드

```
public boolean equals(Object obj) {  
    return (this == obj);  
}
```

```
Student student1 = new Student(1001, "둘리");  
Student student2 = new Student(1001, "둘리");  
student1.equals(student2); → false
```



대부분 equals 메소드를 객체의 내용으로 비교하도록 재정의한다.  
equals 메소드를 재정의한 경우, 두 객체의 내용이 같다면 true를 반환한다.

toString 메소드 재정의하기

```
class Student {  
  
    int id;  
    String name;  
  
    @Override  
    public boolean equals(Object obj) {  
        if(obj instanceof Student) {  
            Student student = (Student)obj;  
            if(this.id == student.id) {  
                return true;  
            }  
        }  
        return false;  
    }  
}
```

```
Student student1 = new Student(1001, "둘리");  
Student student2 = new Student(1001, "둘리");  
student1.equals(student2); → true
```

# Object 클래스

## toString 메소드

### toString 메소드

- toString 메소드는 객체의 정보를 문자열로 반환한다.
- toString 메소드 원형은 인스턴스 주소값을 반환한다.

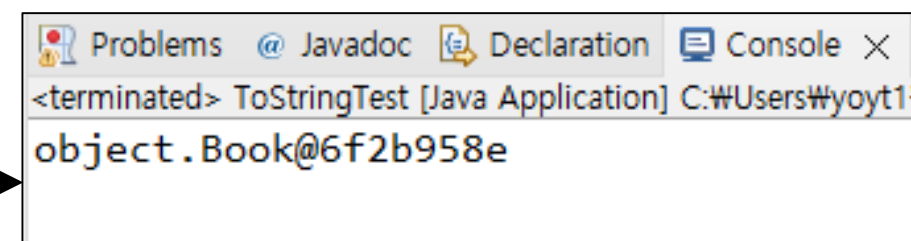
Object클래스의 toString 메소드

```
    * @return a string representation of the object.
    */
    public String toString() {
        return getClass().getName() + "@" + Integer.toHexString(hashCode());
    }
```

→ 클래스이름 + @ + 객체주소

```
Book book = new Book(200, "개미");
```

```
System.out.println( book.toString() );
```



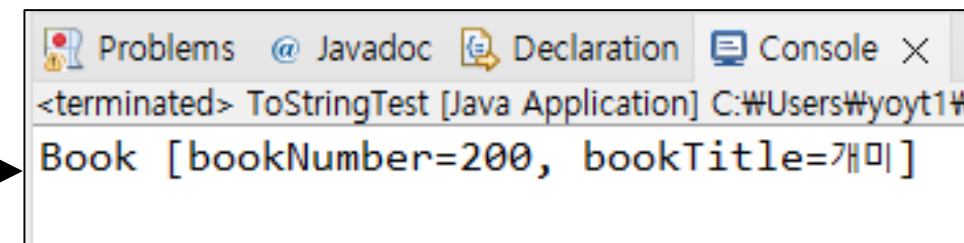


대부분 toString 메소드를 객체의 내용을 반환하도록 메소드를 재정의한다.  
toString 메소드를 재정의한 경우, 객체의 내용을 반환한다.

toString 메소드 재정의 하기

```
public class Book {  
    int bookNumber;  
    String bookTitle;  
  
    @Override  
    public String toString() {  
        return "Book [bookNumber=" + bookNumber + ", bookTitle=" + bookTitle + "];"  
    }  
}
```

```
Book book = new Book(200, "개미");  
System.out.println(book.toString());
```



The screenshot shows an IDE's console window with the following content:

```
Problems  @ Javadoc  Declaration  Console ×  
<terminated> ToStringTest [Java Application] C:\Users\wyoyt1\W...  
Book [bookNumber=200, bookTitle=개미]
```

An arrow points from the `System.out.println(book.toString());` line in the code block above to the console output.

# Object 클래스

## String 클래스의 toString 메소드

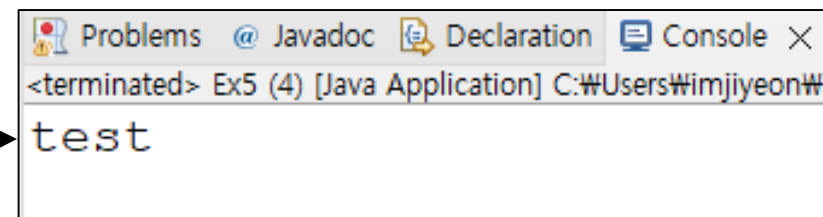
String 클래스는 Object 클래스로부터 상속받은 toString 메소드를 재정의해서 사용한다.  
String 타입의 변수로 toString() 메소드를 호출하면 문자열값이 반환된다.

String 클래스의 toString 메소드

```
public String toString() {  
    return this;  
}
```

```
String str = "test";
```

```
System.out.println( str.toString() );
```



String 클래스는 Object 클래스로부터 상속받은 equals 메소드를 재정의해서 사용한다.  
두 문자열을 비교할 때, 같은 값을 가지고 있으면 true를 반환한다.

String 클래스의 equals 메소드

```
public boolean equals(Object anObject) {  
    if (this == anObject) {  
        return true;  
    }  
    return (anObject instanceof String aString)  
        && (!COMPACT_STRINGS || this.coder == aString.coder)  
        && StringLatin1.equals(value, aString.value);  
}
```

```
String str1 = new String("test");
```

```
String str2 = new String("test");
```

```
str1.equals(str2);    → true
```

### String 클래스란?

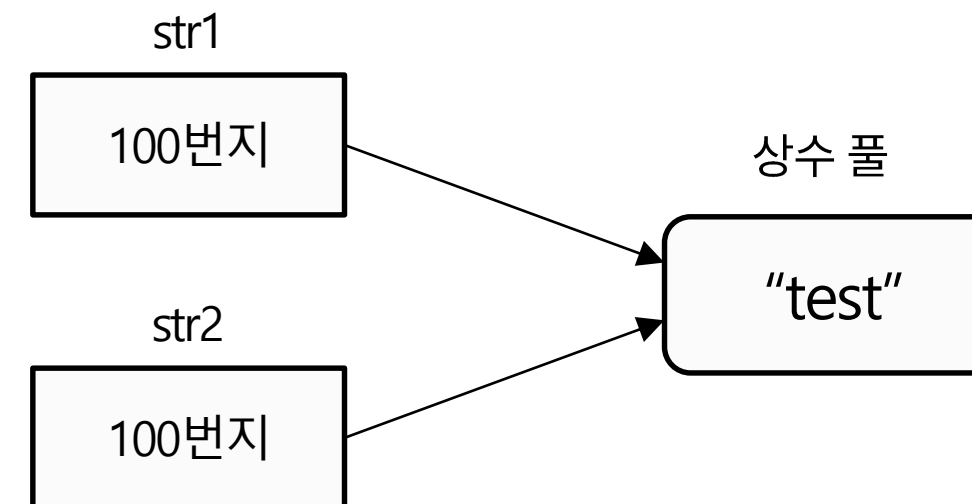
- String 클래스는 문자열을 저장하기 위한 클래스이다.

### 문자열을 생성하는 두가지 방법

#### 1. 리터럴로 생성하기

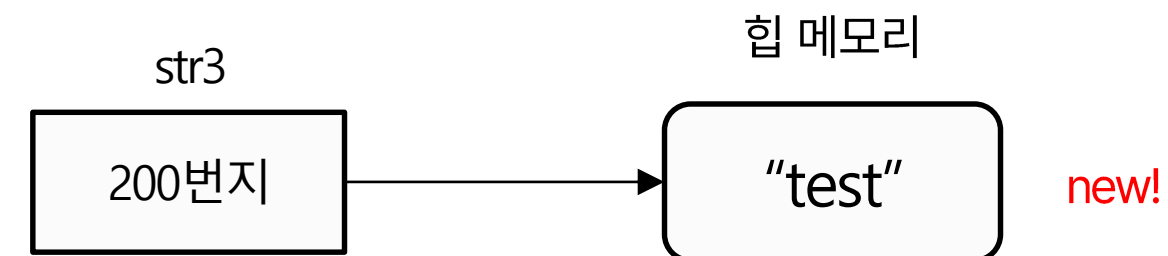
```
String str1 = "test";
```

```
String str2 = "test";
```



#### 2. new 연산자 사용하기

```
String str3 = new String("test");
```



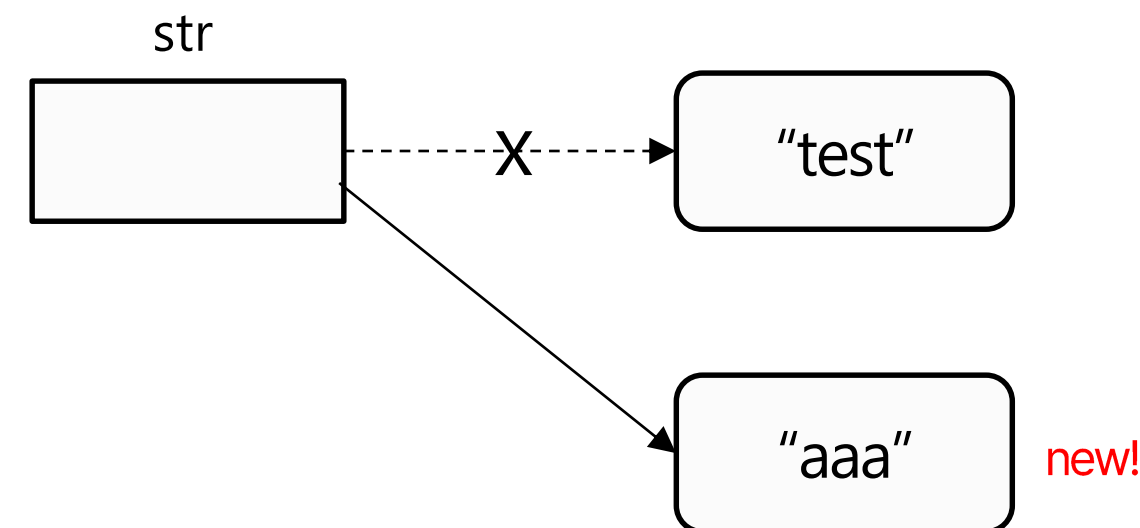
### String 클래스의 특징

- String 클래스는 문자열을 저장하는 배열과 문자열을 조작하는 다양한 메소드를 가지고 있다.
- String 객체는 한번 생성된 문자열은 변경할 수 없다.
- 이 특성은 클래스 내부에 있는 배열이 final로 선언되어 있기 때문이다.

String클래스의 final byte[] 변수

```
@Stable  
private final byte[] value;
```

```
String str = new String("test");  
str = "aaa";
```



# String 클래스

## String클래스의 메소드

메소드	설명
char charAt(int index)	지정된 위치(index)에 있는 문자를 반환한다
String concat(String str)	문자열 뒤에 str을 이어 붙여서 새로운 문자열을 생성한다
boolean contains(String str)	str문자열이 포함되어 있는지 확인한다
boolean endsWith(String suffix)	문자열 suffix로 끝나는지 확인한다
boolean equals(String str)	대소문자를 구분하여 자기 자신과 str이 같은 문자열인지 비교한다
boolean equalsIgnoreCase(String str)	대소문자 구분없이 자기 자신과 str이 같은 문자열인지 비교한다
int indexOf(char ch)	문자ch 의 위치(index)를 찾는다

메소드	설명
int length()	문자열 길이를 반환한다
String replace(char a, char b)	문자열에서 문자a를 찾아서 문자b로 바꾼다
String[] split(String regex)	문자열을 지정된 구분자regex로 분리하여 문자열 배열을 만든다
boolean startsWith(String prefix)	문자열 prefix로 시작하는지 확인한다
String substring(int begin, int end)	시작위치(begin)부터 마지막 위치(end)까지 문자열을 자른다
String toLowerCase()	모든 문자열을 소문자로 변환한다
String toUpperCase()	모든 문자열을 대문자로 변환한다
String trim()	문자열의 왼쪽 끝과 오른쪽 끝에 있는 공백이 있으면 제거한다
static String valueOf(모든 자료형)	지정된 값을 문자열로 변환한다

# Wrapper 클래스

## 기본자료형을 위한 클래스

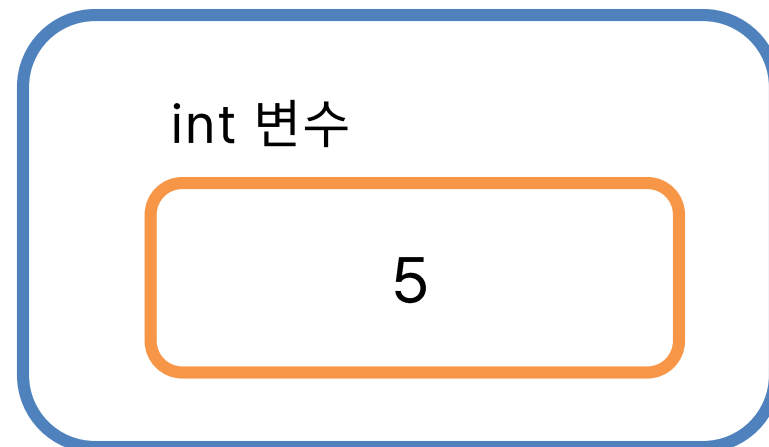
### Wrapper 클래스란?

- Wrapper 클래스는 기본 자료형을 객체로 다룰 때 사용하는 클래스이다
- Wrapper 클래스는 기본 자료형을 객체로 감싸서, 기본자료형을 객체처럼 사용할 수 있도록 해준다.

### Wrapper는 언제 사용하지?

1. 메소드의 매개변수나 반환타입이 객체 타입일 때
2. 리스트에 기본 자료형을 저장할 때

#### Integer 클래스



#### <리스트에 기본 자료형을 저장하는 경우>

```
new ArrayList<int> ();  
new ArrayList< Integer > ();  
new ArrayList< char > ();  
new ArrayList< Character > ();
```



기본형과 Wrapper클래스는 일대일 관계이며, Wrapper클래스는 대문자로 시작한다.

기본형	Wrapper 클래스
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

# Wrapper 클래스

## Wrapper 클래스 구조

### Wrapper 클래스의 특징

- 기본자료형과 값의 범위가 같다.
- 값을 처리하는 다양한 메소드를 가지고 있다.
- 오토박싱과 언박싱을 지원한다.

Integer 클래스 내부

int형 멤버변수 + 메소드

```
    */
    private final int value;

    public static final int    MAX_VALUE = 0x7fffffff;

    public static final int    MIN_VALUE = 0x80000000;

    public static int parseInt(String s)
        return parseInt(s,10);
    }

    @IntrinsicCandidate
    public int intValue() {
        return value;
    }
```

기존에는 객체를 생성할 때 new 생성자함수를 호출했다.

```
Integer iNum = new Integer(100);
```

자바5부터는 오토박싱과 언박싱이라는 기능이 도입되었다.

“오토박싱”은 기본자료형 값을 해당 래퍼클래스의 객체로 자동으로 변환하는 것을 의미한다.

```
Integer iNum = 100;    //int → Integer
```

반대로 “언박싱”은 래퍼클래스에서 기초자료형 값을 추출하는 것을 의미한다.

```
int num = iNum;        //Integer → int
```

