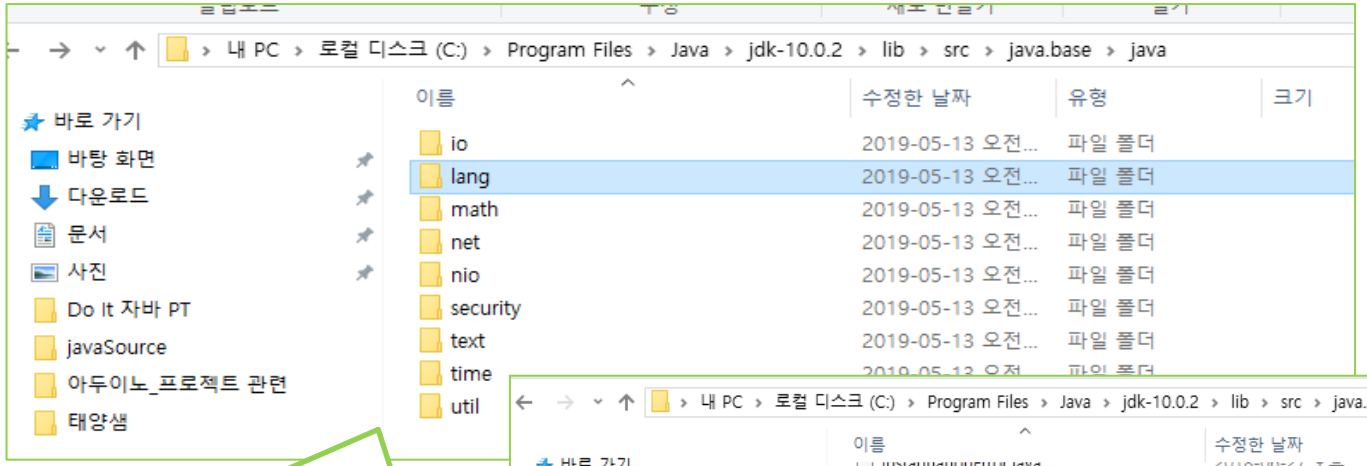
11장. 기본 클래스

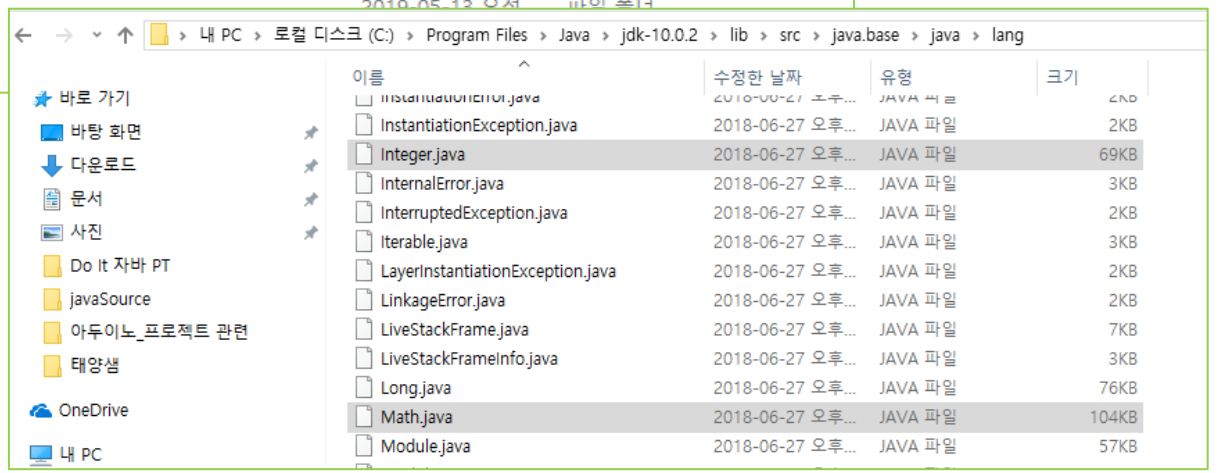
자바 JDK로 프로그래밍 날개 달기

java lang 패키지

- **java.lang 패키지**
  - Java.lang.Object / java.lang.String은 어디에 위치할까?
  - 포함된 클래스와 인터페이스는 import없이 사용할 수 있다.



자바 설치 경로에서 src.zip 압축풀기



java lang 패키지

- **주요 클래스**

클 래 스	용 도
Object	- 자바 클래스의 최상위 클래스로 사용
System	- 표준 입력 장치(키보드)로부터 데이터를 입력받을때 사용 - 표준 출력 장치(모니터)로 출력하기 위해 사용 - 자바 가상 기계를 종료시킬때 사용 - 쓰레기 수집기를 실행 요청할 때 사용
Class	- 클래스를 메모리로 로딩할 때 사용
String	- 문자열을 저장하고 여러 가지 정보를 얻을 때 사용
StringBuffer, StringBuilder	- 문자열을 저장하고 내부 문자열을 조작할 때 사용
Math	- 수학함수를 이용할 때 사용
Wrapper : Byte, Short, Character, Integer, Float, Double	- 기본 타입의 데이터를 갖는 객체를 만들 때 사용 - 문자열 기본타입으로 변환할때 사용 - 입력값 검사에 사용

Object 클래스

- **모든 클래스의 최상위 클래스 Object**
  - Java.lang.Object
  - 모든 클래스는 Object 클래스로부터 상속을 받는다. (컴파일러가 자동 변환)

```
class Student {
    int studentID;
    String studentName;
}
```

코드 작성할때

생략되어 있음

```
class Student extends Object {
    int studentID;
    String studentName;
}
```

컴파일러가 변환

Object 클래스

- **Object 클래스의 주요 메서드**

메서드	용 도
String **toString()**	객체를 문자열로 표현하여 반환한다. 재정의하여 객체에 대한 설명이나 특정 멤버 변수 값을 반환한다.
boolean **equals(Object obj)**	두 인스턴스가 동일한지 여부를 반환한다. 재정의하여 **논리적으로 동일한 인스턴스**임을 정의 할 수 있다.
int **hashCode()**	객체의 해시 코드 값을 반환한다.
Object **clone()**	객체를 복제하여 동일한 멤버 변수 값을 가진 새로운 인스턴스를 생성한다.
Class **getClass()**	객체의 클래스를 반환합니다.

Object 클래스

- **Object 클래스의 주요 메서드**

Module java.base
Package java.lang

Class Object

java.lang.Object

public class **Object**

Class Object is the root of the c

All Methods | Instance Methods | Concrete Methods | Deprecated Methods

Modifier and Type	Method	Description
protected Object	clone()	Creates and returns a copy of thi
boolean	equals(Object obj)	Indicates whether some other ob
protected void	finalize()	**Deprecated.** The finalization mechanism is inh
Class<?>	getClass()	Returns the runtime class of this
int	hashCode()	Returns a hash code value for the
void	notify()	
void	notifyAll()	
String	toString()	

toString

```
public String toString()
```

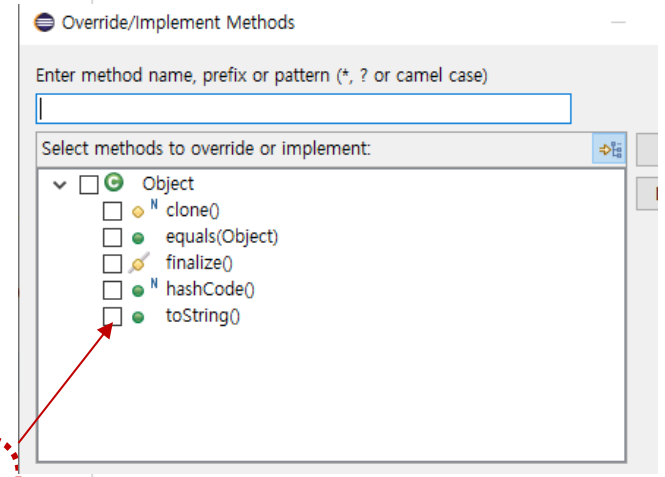
Returns a string representation of the object. In general, the toStr
that is easy for a person to read. It is recommended that all subclas

The toString method for class Object returns a string consisting
the hash code of the object. In other words, this method returns a s

toString() 메서드

- **toString() 메서드**
  - 객체 정보를 문자열로 바꾸어 줌
  - Integer나 String등 클래스는 toString() 메서드가 이미 재정의 되어 있음.
  - 사용자 정의 클래스는 toString() 재정의 해야함

```
package object.tostring;

class Book{
    int bookNumber;
    String bookTitle;

    Book(int bookNumber, String bookTitle){
        this.bookNumber = bookNumber;
        this.bookTitle = bookTitle;
    }

    @Override
    public String toString() {
        return bookTitle + "," + bookNumber;
    }
}
```



toString() 메서드

- **toString() 메서드**

```
public class ToStringEx {

    public static void main(String[] args) {
        //String으로 생성한 인스턴스는 문자열로 출력
        String name = new String("홍길동");
        System.out.println(name);
        System.out.println(name.toString());

        //Book으로 생성한 문자열은 toString() 재정의 해야함
        Book book = new Book(100, "개미");
        System.out.println(book);
        System.out.println(book.toString());
    }
}
```

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

이미 재정의 되어있음

```
홍길동
홍길동
개미,100
개미,100
```

재정의하기 이전

```
public String toString() {
    return getClass().getName() + "@" + Integer.toHexString(hashCode());
}
```

equals() 메서드

- **equals() 메서드**
  - 두 인스턴스의 주소 값을 비교하여 boolean 값(true/false)를 반환
  - 재정의하여 두 인스턴스가 논리적으로 동일함의 여부를 반환.
  - 같은 번호의 책인 경우 여러 인스턴스의 주소값은 다르지만, 같은 책으로 인정할 수 있으므로 equals() 메서드를 재정의해야 함.

```
public class EqualsTest {
    public static void main(String[] args) {
        //String으로 생성한 인스턴스의 메모리 주소와 값 비교
        String name1 = new String("홍길동");
        String name2 = new String("홍길동");
        System.out.println(name1==name2);  //메모리 주소는 다르다.
        System.out.println(name1.equals(name2)); //저장된 값은 같다.

        //Book으로 생성한 인스턴스의 메모리 주소와 값 비교
        Book book1 = new Book(100, "개미");
        Book book2 = new Book(100, "개미");
        System.out.println(book1==book2);
        System.out.println(book1.equals(book2));
    }
}
```

```
false
true
=======
false
false
```

Book 클래스에서는 equals() 메서드 재정의 필요

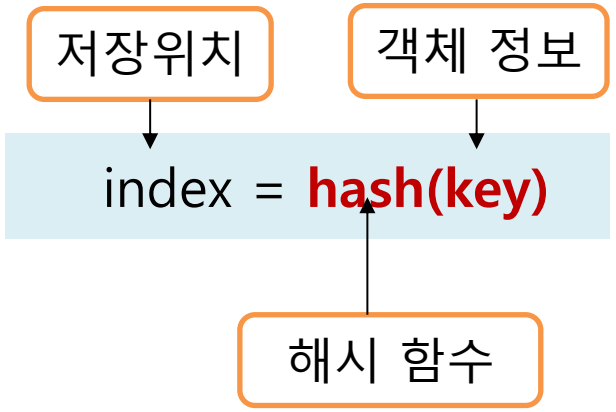
equals() 메서드

- equals() 재정의(Override)

```
class Book{
    int bookNumber;
    String bookTitle;

    Book(int bookNumber, String bookTitle){
        this.bookNumber = bookNumber;
        this.bookTitle = bookTitle;
    }

    @Override
    public boolean equals(Object obj) {
        if(obj instanceof Book) {
            Book book = (Book)obj;
            if(this.bookNumber==book.bookNumber)
                return true;
        }
        return false;
    }
```

hashCode() 메서드

- **hashCode() 메서드**
  - hash : 정보를 저장, 검색하기 위해 사용하는 자료 구조
  - 자료의 특정 값(키 값)에 대해 저장 위치를 반환해 주는 해시 함수를 사용함. hash 알고리즘은 검색(search)에 효율적임
  - **hashCode() 메서드는 인스턴스의 저장 주소를 반환함 : 10진수로 나타냄**
  - 힙 메모리에 인스턴스가 저장되는 방식이 hash임



hashCode() 메서드

- **hashCode() 재정의**

  논리적으로 동일함을 위해 equals() 메서드를 재정의 하였다면
  hashCode() 메서드도 재정의하여 동일한 값이 반환되도록 해야 함.
  즉, 자바에서는 두 인스턴스가 같다면 hashCode() 반환 해시 코드값이 같아야 한다.

```
class Book{
    int bookNumber;
    String bookTitle;

    Book(int bookNumber, String bookTitle){
        this.bookNumber = bookNumber;
        this.bookTitle = bookTitle;
    }

    @Override
    public int hashCode() {
        return bookNumber;
    }
```

책번호가 같으면 논리적으로
객체가 동일함을 재정의

hashCode() 메서드

- HashCodeTest

```
public class EqualTest {

    public static void main(String[] args) {
        String name1 = new String("흥부");
        String name2 = new String("흥부");

        //equals() 테스트
        System.out.println(name1 == name2);
        System.out.println(name1.equals(name2));

        Book book1 = new Book(100, "미생");
        Book book2 = new Book(100, "미생");
        System.out.println(book1 == book2);
        System.out.println(book1.equals(book2));

        //hashcode() 테스트
        System.out.println(name1.hashCode());
        System.out.println(name2.hashCode());

        System.out.println(book1.hashCode());
        System.out.println(book2.hashCode());
    }
}
```

clone() 메서드

- **clone() 메서드**
  - 객체의 원본을 복제하는데 사용하는 메서드
  - 원본을 유지해 놓고 복사본을 사용하고 싶을때 사용한다.

  - 객체의 정보가 같은 인스턴스가 생성되므로 객체지향의 정보은닉, 정보보호의 관점에 위배될 수 있음
    그래서, 객체의 clone() 메서드 사용을 허용한다는 의미로 cloneable 인터페이스를 명시함.

```
class Circle implements Cloneable{
    Point point;
    int radius;
}
```

Module java.base
Package java.lang

Interface Cloneable

All Known Subinterfaces:

AclEntry, Attribute, AttributedCharacterIterator, CRLSelector, Descriptor, ExtendedGSSCredential, GS

```
public interface Cloneable
```

clone() 메서드

- clone() 메서드로 배열 복사

```java
public class CloneTest {

    public static void main(String[] args) {
        int[] arr1 = {10, 20, 30, 40};
        int[] arr2 = new int[4];
        int i;

        System.out.println("일반 배열 복사");
        for(i = 0; i < arr1.length; i++) {
            arr2[i] = arr1[i];
        }

        //arr2 출력 - 일반 for
        for(i = 0; i < arr2.length; i++) {
            System.out.print(arr2[i] + " ");
        }
        System.out.println();

        System.out.println("clone()으로 복사");
        int[] arr3 = arr1.clone();


        //arr3 출력 - 향상 for문
        for(int arr : arr3)
            System.out.print(arr + " ");
    }
}
```

clone() 메서드

- clone() 메서드로 인스턴스 복제하기

Point 클래스

```
public class Point {
    int x;
    int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public String toString() {
        return "x=" + x + ", " + "y=" + y;
    }
}
```

clone() 메서드

- clone() 메서드로 인스턴스 복제하기

Circle 클래스

```
public class Circle implements Cloneable{
    //clone() 메서드를 사용하려면 Cloneable 인터페이스를 구현해야함
    Point center;
    int radius;

    public Circle(int x, int y, int radius) {
        center = new Point(x, y);
        this.radius = radius;
    }

    @Override
    public String toString() {
        return "중심점은 " + center + "이고, 반지름은 " + radius + "입니다.";
    }

    @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}
```

clone() 메서드

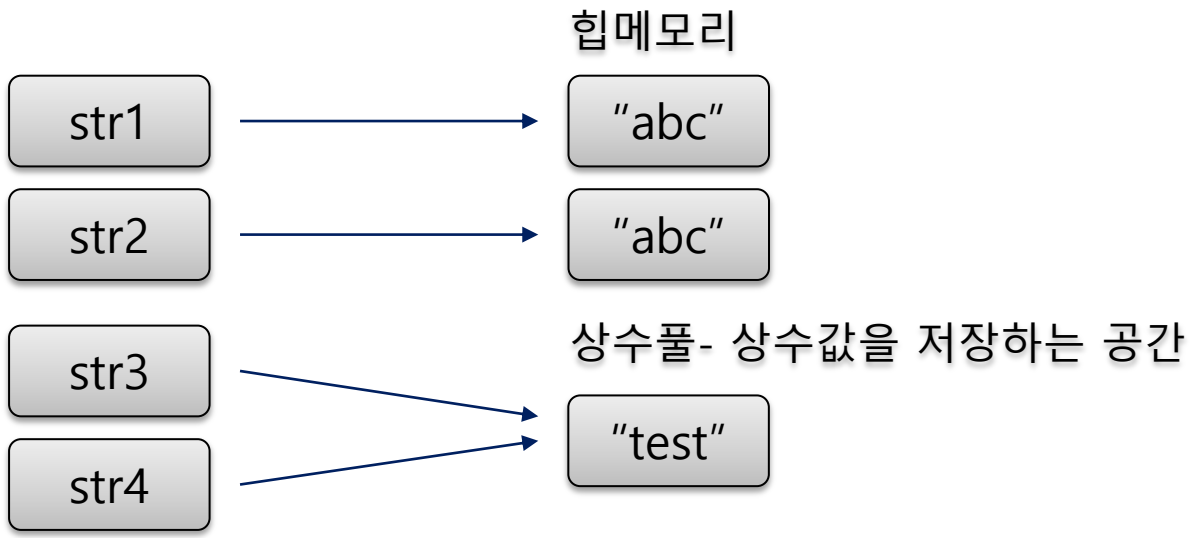
- clone() 메서드로 인스턴스 복제하기

```
public class ObjectCloneTest {

    public static void main(String[] args) throws CloneNotSupportedException {
        //Circle 객체 생성
        Circle circle = new Circle(10, 20, 4);

        //Circle 객체 복사
        Circle copyCircle = (Circle) circle.clone();

        System.out.println(circle);
        System.out.println(copyCircle);
    }

}
```

```
중심점은 x=10, y=20이고, 반지름은 4입니다.
중심점은 x=10, y=20이고, 반지름은 4입니다.
```

String 클래스

- **String 클래스**
  - 문자열을 저장하고 여러 가지 정보를 얻을 때 사용하는 클래스
  - 문자열을 사용하여 생성자의 매개변수로 하여 생성하는 방식과 이미 생성된 문자열 상수를 가리키는 방식이 있다.



String 클래스

- **String 클래스**

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

        String str1 = new String("abc");
        String str2 = new String("abc");

        //인스턴스가 매번 새로 생성되므로 주소값이 다름
        System.out.println(str1==str2);
        //문자열 값은 같음
        System.out.println(str1.equals(str2));

        String str3 = "abc";
        String str4 = "abc";

        //문자열 "abc"는 상수 풀에 저장되어 있어서 주소 값이 같음
        System.out.println(str3==str4);
        System.out.println(str3.equals(str4));
    }
}
```

```
false
true
true
true
```

String 클래스

- **String 클래스**
  - 자바는 String 클래스를 제공해 char[ ] 배열을 직접 구현하지 않고도 편리하게 문자를 사용할 수 있다.(C언어의 경우 char[ ] 배열로 문자열 구현함)
  - 한 번 생성된 문자열은 변경할 수 없다.(final char value[ ])
  - 메모리 상태 변화

String 클래스

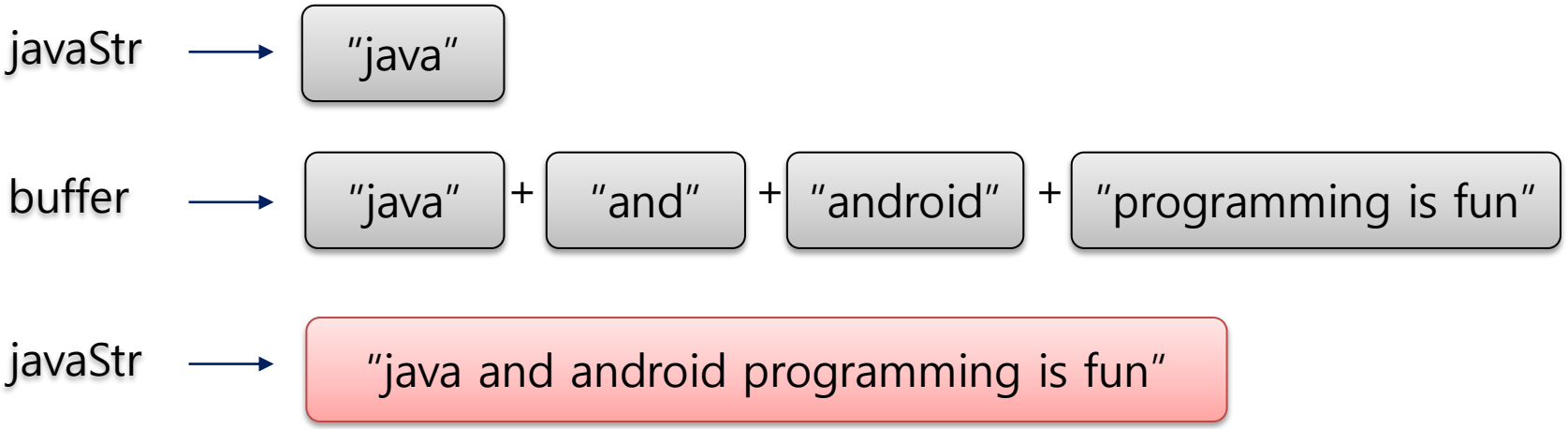
- **String 클래스**

```
public class StringTest2 {
    public static void main(String[] args) {

        String javaStr = new String("java");
        String androidStr = new String("android");

        System.out.println(javaStr);
        System.out.println("처음 문자열 주소 값 : " + System.identityHashCode(javaStr));

        //문자열 javaStr과 문자열 androidStr을 연결
        javaStr = javaStr.concat(androidStr);

        System.out.println(javaStr);
        System.out.println("처음 문자열 주소 값 : " + System.identityHashCode(javaStr));
    }
}
```

```
java
처음 문자열 주소 값 : 758529971
javaandroid
처음 문자열 주소 값 : 2104457164
```

String 클래스

- **StringBuilder와 StringBuffer 클래스 활용하기**
  - String 클래스를 사용하여 문자열을 계속 연결하거나 변경하는 프로그램을 작성하면 내부의 문자열이 변경되지 않기 때문에 메모리가 많이 낭비된다.
  - StringBuilder나 StringBuffer 클래스를 사용하면 추가 메모리를 사용하지 않고 문자열을 연결할 수 있다.



String 클래스

- **StringBuilder 클래스 예제**
  - 내부에 변경 가능한(final이 아닌) char[ ] 변수를 가지고 있다.
  - 기존 사용하던 char[ ]이 확장되므로 추가 메모리를 사용하지 않는다.

```
String javaStr = new String("java");
System.out.println("javaStr 문자열 주소 : " + System.identityHashCode(javaStr));

StringBuilder buffer = new StringBuilder(javaStr);
System.out.println("연산 전 buffer 메모리 주소 : " +  System.identityHashCode(buffer));

//문자열 추가 -> 연결
buffer.append(" and");
buffer.append(" android");
buffer.append(" programming is fun!!");
System.out.println("연산 후 buffer 메모리 주소 : " +  System.identityHashCode(buffer));

//String 클래스 형으로 반환
javaStr = buffer.toString();
System.out.println(javaStr);
```

```
javaStr 문자열 주소 : 758529971
연산 전 buffer 메모리 주소 : 2104457164
연산 후 buffer 메모리 주소 : 2104457164
java and android programming is fun!!
```

Wrapper 클래스

- **기본 자료형을 위한 클래스**
  - 자바에서 기본 자료형처럼 사용할 수 있는 클래스를 제공한다.
  - 이를 기본 자료형을 감쌌다는 의미로 Wrapper 클래스라고 한다.

지금까지 정수를 사용할 때 기본형인 int를 사용했다. 그런데 정수를 객체형으로 사용해야 하는 경우가 발생한다.

```
public void setValue(Integer i){.....}
//객체를 매개변수로 받는 경우
public Integer returnValue(){.....}
//반환값이 객체인 경우
```

기본형	Wrapper 클래스
boolean	Boolean
byte	Byte
char	Character
int	Integer
double	Double

Wrapper 클래스

- **Integer 클래스**

```java
public class IntegerTest {
    public static void main(String[] args) {
        //Integer num1 = new Integer(100);
        Integer num1 = 100;     //오토박싱-(기본형 -> 클래스형)
        int num2 = 200;
        int sum = num1.intValue() + num2;
        //언방식 - Integer num1을 int형으로 변환
        System.out.println(sum);

        //valueOf() ->정수나 문자열을 Integer 클래스로 반환
        Integer n1 = Integer.valueOf("100");
        System.out.println(n1);

        //parseInt() -> 문자열이 전달된 경우 int로 반환
        int n2 = Integer.parseInt("200");  //언박싱
        System.out.println(n2);

        //ArrayList에서 사용
        ArrayList<Integer> numberList = new ArrayList<>();
        numberList.add(10);
        numberList.add(20);
        numberList.add(30);

        for(Integer i : numberList)
            System.out.println(i);
```

```java
public static Integer valueOf(String s) throws NumberFormatException {
    return Integer.valueOf(parseInt(s, 10));
}
```

Class 클래스

- **Class 클래스**
  - 자바의 모든 클래스와 인터페이스는 컴파일 후 class 파일로 생성됨.
  - Class 클래스는 컴파일된 class 파일에서 객체의 정보를 가져올 수 있음.
  - Class 클래스 정보 가져오기(멤버변수, 생성자, 메서드)

① Object 클래스의 getClass() 메서드 사용하기

```
String s = new String();
Class c = s.getClass()
```

② 클래스 파일 이름을 class 변수에 직접 대입하기

```
Class c = String.class
```

③ Class.forName("클래스 이름") 메서드 사용하기

```
Class c = Class.forName("java.lang.String")
```

Class 클래스

- Person 클래스

```java
package classexam;
public class Person {
    private String name;
    private int age;

    public Person() {}

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

Class 클래스

- Person의 Class 정보 가져오기

```java
public class ClassTest {

    public static void main(String[] args) throws ClassNotFoundException {
        Person person = new Person();

        System.out.println("=== 클래스 이름 가져오기 ===");
        Class pClass1 = person.getClass();
        System.out.println(pClass1.getName());

        Class pClass2 = Class.forName("classinfo.Person");
        System.out.println(pClass2.getName());

        System.out.println("===== 생성자 정보 =====");
        Constructor[] cons = pClass1.getConstructors();
        for(Constructor c : cons)
            System.out.println(c);

        System.out.println("===== 멤버 변수 정보 =====");
        Field[] fields = pClass1.getDeclaredFields();
        for(Field field : fields)
            System.out.println(field);

        System.out.println("===== 메서드 정보 =====");
        Method[] methods = pClass1.getMethods();
        for(Method method : methods)
            System.out.println(method);
    }
}
```

Class 클래스

- Person의 Class 정보 가져오기

```
=== 클래스 이름 가져오기 ===
classinfo.Person
classinfo.Person
===== 생성자 정보 =====
public classinfo.Person()
===== 멤버 변수 정보 =====
private java.lang.String classinfo.Person.name
private int classinfo.Person.age
===== 메서드 정보 =====
public java.lang.String classinfo.Person.getName()
public void classinfo.Person.setName(java.lang.String)
public void classinfo.Person.setAge(int)
public int classinfo.Person.getAge()
public final native void java.lang.Object.wait(long) throws java
public final void java.lang.Object.wait(long,int) throws java.la
public final void java.lang.Object.wait() throws java.lang.Inter
public boolean java.lang.Object.equals(java.lang.Object)
public java.lang.String java.lang.Object.toString()
public native int java.lang.Object.hashCode()
public final native java.lang.Class java.lang.Object.getClass()
public final native void java.lang.Object.notify()
public final native void java.lang.Object.notifyAll()
```