자바의 유용한 클래스들

작성자: 신채린

Object Class

- 모든 클래스의 최상위 클래스
- lang.Object 클래스
- 모든 클래스는 Object 클래스에서 상속 받는다.
- 모든 클래스는 Object 클래스의 메서드를 사용할 수 있다.
- 모든 클래스는 Object 클래스의 일부 메서드를 재정의하여 사용할 수 있다.

toString() 메서드

원형

- getClass().getName() + '@' + Integer.toHexString(hashCode())

- 객체의 정보를 String으로 바꾸어 사용할 때 유용함
- 자바 클래스 중에는 이미 정의된 클래스가 많음
  - 예시) String, Integer Calendar 등
- 많은 클래스에서 재정의하여 사용

toString override하기

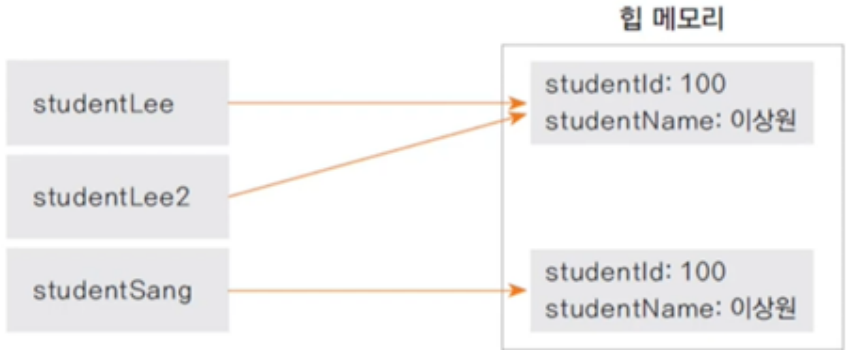
```
package Chapter10;

class Book{
    String title;
    String author;

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    @Override
    public String toString() {
        return author + "," + title;
    }

}
public class ToStringTest {
    public static void main(String[] args) {
        Book book = new Book("토지", "박경리");

        System.out.println(book);

        String str = new String("토지");
        System.out.println(str.toString());
    }
}
```

equals() 메서드

- 두 객체의 동일함을 논리적으로 재정의할 수 있음
- 물리적 동일함 : 같은 주소를 가지는 객체
- 논리적 동일함: 같은 학번의 학생, 같은 주문 번호의 주문
- **물리적으로 다른 메모리에 위치한 객체라도 논리적으로 동일함을 구현하기 위해 사용**하는 메서드.

예시

```
Student studentLee = new Student(100, "이상원");
Student studentLee2 = studentLee;
Student studentSang = new Student(100, "이상원");
```



- 물리적으로 다른 위치에 있지만, 논리적으로는 같은 학생임을 구현해야 함

```
package Chapter10;

class Student{
    int studentNum;
    String studentName;

    public Student(int studentNum, String studentName) {
        this.studentNum = studentNum;
        this.studentName = studentName;
    }

    @Override
    public boolean equals (Object obj) {
        if ( obj instanceof Student) {
            // downcasting 진행
            Student std = (Student) obj;
            return (this.studentNum == std.studentNum);
        }
        return false;
    }
}

public class EqualsTest {
    public static void main(String[] args) {
        Student Lee = new Student(100, "이순신");
        Student Lee2 = Lee;
        Student Shin = new Student(100, "이순신");

        System.out.println(Lee == Lee2); //true
        System.out.println(Lee == Shin); //false
        System.out.println(Lee.equals(Shin)); //true -> 논리적으로 같다고 재정의해야 true가 된다.


    }
}
package Chapter10;

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

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

        System.out.println(str1 == str2); //false
        System.out.println(str1.equals(str2)); //true

    }
}
```

hashCode() 메서드

- hashCode() 메서드의 반환 값
  - 인스턴스가 저장된 가상머신의 주소를 10진수로 반환
- 두 개의 서로 다른 메모리에 위치한 인스턴스가 동일하다는 것은?
  - 논리적으로 동일: equals()의 반환값이 true
  - 동일한 hashCode 값을 가짐: hashCode()의 반환 값이 동일

예시

```
Integer i1 = 100;
        Integer i2 = 100;

        System.out.println(i1.equals(i2)); //true

        Integer i3 = new Integer(100);
        Integer i4 = new Integer(100);

        System.out.println(i1.equals(i2)); //true
        System.out.println(i1.hashCode()); //100
        System.out.println(i1.hashCode()); //100
```

Clone() 메서드

- 객체의 복사본을 만듦
- 기본 틀 (prototype) 으로부터 같은 속성 값을 가진 객체의 복사본을 생성할 수 있음
- 객체지향 프로그래밍의 정보은닉에 위배되는 가능성이 있으므로 **복제할 객체는 cloneable 인터페이스를 명시**해야 함

Class 클래스

Class 클래스

- 자바의 모든 클래스와 인터페이스는 컴파일 후 class 파일로 생성됨
- class 파일에는 객체의 정보 (멤버 변수, 메서드, 생성자 등)가 포함되어 있음
- Class 클래스는 컴파일된 class 파일에서 객체의 정보를 가져올 수 있음
- 동적 loading할 때 많이 활용!

Class 클래스 가져오기

· // 1번 방법· String s = new String();· Class c = s.getClass();· · // 2번 방법· Class c = String.Class;· · // 3번 방법· Class c = Class.forName("java.lang.String"); // 동적로딩

reflection 프로그래밍

- Class 클래스로부터 객체의 정보를 가져와 프로그래밍 하는 방식
- 로컬에 객체가 없고, 자료형을 알 수 없는 경우 유용한 프로그래밍
- lang.reflect 패키지에 있는 클래스 활용

newInstance() 메서드

- Class 클래스 메서드
- new 키워드를 사용하지 않고 인스턴스를 생성

forName() 메서드와 동적 로딩

- Class 클래스 static 메서드
- 동적 로딩이란?
  - 컴파일 시에 데이터 타입이 모두 binding이 되어 자료형이 로딩되는 것
  - (static loading)이 아니라 실행 중에 데이터 타입을 알고 binding 되는 방식
- 실행 시에 로딩되므로 경우에 따라 다른 클래스가 사용될 수 있어 유용함
- 컴파일 타임에 체크할 수 없으므로 해당 문자열에 대한 클래스가 없는 경우 예외 (ClassNotFoundException)이 발생할 수 있음

String, Wrapper 클래스

String 클래스

- 선언하는 방법?

o String str1 = new String("ABC"); //인스턴스로 생성됨o String str2 = "ABC"; // 상수풀에 있는 문자열을 가리킴

String은 Immutable

- 한 번 선언되거나 생성된 문자열은 변경할 수 없음
- String 클래스의 concat() 메서드 혹은 “+”를 이용하여 String을 연결하는 경우 문자열은 새로 생성됨

StringBuilder와 StringBuffer

- 기본적인 char[] 배열을 멤버변수가 가지고 있는 클래스
- 문자열을 변경하거나 연결하는 경우 사용하면 편리한 클래스
- StringBuilder는 멀티 쓰레드로 프로그래밍에서 동기화가 보장됨
- 단일 쓰레드 프로그래밍에서는 StringBuilder를 사용하는 것이 더 좋음
- toString() 메서드로 String 반환

Wrapper 클래스

- 기본 자료형에 대한 클래스

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