Java프로그래밍

# 8강. java.lang 패키지 (교재 7장)

컴퓨터과학과 김희천 교수

한국방송통신대학교

# 오늘의 학습목차

1. Object 클래스
2. String 클래스
3. StringBuffer 클래스
4. 포장 클래스
5. System 클래스

# 1. Object 클래스

1. Object 클래스

# 1) java.lang 패키지

- 자바 프로그래밍에 필요한 기본 클래스를 제공
  - ✓ java.lang 패키지에 존재하는 클래스를 사용할 때는 import 문이 필요 없음

주요 클래스

- Object, System, Math
- String, StringBuffer
- Thread
- Exception, Throwable, Error
- 포장 클래스
  - Number, Integer, Double, Character, Boolean 등

# 2) Object 클래스와 주요 메소드

- 자동으로 모든 클래스의 조상이 되는 클래스
  - ✓ 클래스 계층 구조에서 루트가 되는 클래스
  - ✓ 모든 클래스는 자동으로 Object 클래스를 상속받음

주요 메소드

- protected Object clone( )
  - 객체를 복제하여 반환
- public boolean equals(Object obj)
- public int hashCode( )
  - 객체를 식별하는 정수값을 반환
- public String toString( )

# 3) String toString( ) 메소드(1)

- 객체의 문자열 표현을 반환
  - ✓ 클래스이름@16진수해시코드로 표현
- 문자열의 + 연산, System.out.print( ) 등에서 필요함
- 자식 클래스에서 재정의할 수 있음
  - ✓ String, Integer 클래스 등에서 재정의되어 있음

# 3) String toString( ) 메소드(2)

```
class MyClass1 {  }
class MyClass2 extends Object {
    public String toString( ) { return "This MyClass2 class"; }
}
public class MyClass3 {
    public static void main(String args[]) {
       MyClass1 my_class1 = new MyClass1( );
       MyClass2 my_class2 = new MyClass2( );
       System.out.println(my_class1);
       System.out.println(my_class2);
… …
   }
}
```

```
MyClass1@762efe5d
This MyClass2 class
```

# 4) boolean equals(Object obj) 메소드

- 두 객체 변수를 비교해서 두 변수의 참조값이 같을 때 true를 반환
- Object 클래스에서 equals( )의 의미
  - ✓ obj1.equals(obj2)의 결과는 (obj1==obj2)와 같음
- 자식 클래스에서 재정의할 수 있음
  - ✓ String, Integer 클래스 등에서 재정의되어 있음

```
Integer x = new Integer(5);
Integer y = new Integer(10);
Integer z = 5;
Short a = 5;
```

```
System.out.println(x.equals(y));  //false
System.out.println(x.equals(z));  //true
System.out.println(x.equals(a));  //false
System.out.println(x==y);         //false
System.out.println(x==z);         //false
```

# 5) Object clone( ) 메소드(1)

- 객체를 복제하여 리턴함
- 'Cloneable 인터페이스를 구현한 클래스'의 객체만 clone( ) 메소드를 호출할 수 있음
  - ✓ 예외(CloneNotSupportedException)처리를 해 주어야 함

**Class Calendar**

java.lang.Object
    java.util.Calendar

All Implemented Interfaces:

Serializable, **Cloneable**, Comparable<Calendar>

Direct Known Subclasses:

GregorianCalendar

# 5) Object clone( ) 메소드(2)

```
class Box implements Cloneable {
   private int width, height;
   public Box(int w, int h) {
         width=w; height=h;
   }

   public int width( ) { return width; }
   public int height( ) { return height; }
   public Object clone( ) {
       try {
              return super.clone( );
       } catch (CloneNotSupportedException e) {
              return null;
       }
   }
}
```

```
public class CloneTest {
    public static void main(String args[ ]) {
        Box b1 = new Box(20, 30);
        Box b2 = (Box) b1.clone( );
        System.out.println(b2.width( ));  //20
        System.out.println(b2.height( )); //30

        System.out.println(b1);
        System.out.println(b2);
    }
}
```

```
20
30
Box@28a418fc
Box@5305068a
```

# 2. String 클래스

2. String 클래스

# 1) String 클래스와 생성자

- 문자열을 표현하고 처리하기 위한 클래스
- 기본 자료형처럼 다룰 수 있음
  - ✓ String s1 = “Java”; //리터럴을 대입
  - ✓ 같은 리터럴은 1개만 만들어져 공유됨
- String 객체는 내용이 변하지 않는(immutable) 상수 객체

- 생성자
  - ✓ public String( ) : 빈 문자열 객체 생성
  - ✓ public String(String original)
  - ✓ public String(char[ ] value)
  - ✓ public String(char[ ] value, int offset, int count)

# 2) 문자열의 비교 메소드

- int compareTo(String anotherString)
  - ✓ 같으면 0을 리턴하고, 다르면 0이 아닌 정수값을 리턴함
- int compareToIgnoreCase(String anotherString)
- boolean equals(Object anObject)
  - ✓ 문자열이 같으면 true를 리턴하고, 다르면 false를 리턴함
- boolean equalsIgnoreCase(String anotherString)

```
String s1 = "Java";
String s2 = "Java";
String s3 = new String("Java");
String s4 = new String("Java");
```

# 3) 문자열의 검색 메소드

- int indexOf(String str), int indexOf(String str, int fromIndex)
  - ✓ 처음 위치부터 문자열 str을 찾아 처음 등장하는 위치(인덱스)를 리턴함. 없으면 -1을 리턴함
  - ✓ System.out.println( "hamburger".indexOf("urge") );
- int lastIndexOf(String str), int lastIndexOf(String str, int fromIndex)
  - ✓ 마지막 위치부터 앞 방향으로 찾음

# 4) 문자열의 추출 메소드

- char charAt(int index)
  - ✓ index 위치에 있는 문자를 리턴한다.
- String substring(int beginIndex)
  - ✓ beginIndex 위치부터 마지막까지의 문자열을 리턴함
  - ✓ System.out.println( "hamburger".substring(3) );
- String substring(int beginIndex, int endIndex)
  - ✓ beginIndex 위치부터 (endIndex-1)까지의 문자열을 리턴함

# 5) 문자열의 변환 메소드

- 원본 문자열은 변경되지 않고 새로운 객체가 만들어짐
- String replace(char oldChar, char newChar)
  - ✓ oldChar 문자를 newChar 문자로 변환하여 리턴함
- String trim( )
  - ✓ 문자열 앞과 뒤에 나오는 화이트 스페이스 문자를 제거하여 리턴함
- String toUpperCase( )
- String toLowerCase( )
- String concat(String str )
  - ✓ 두 문자열을 연결함

# 6) 다른 자료형을 문자열로 변환하는 메소드

```
public class TransformData {
    public static void main(String args[ ]) {
      System.out.println(String.valueOf(123));
      System.out.println(String.valueOf(5 > 3));
      System.out.println(String.valueOf(3.0));
      System.out.println(String.valueOf('c'));
      char[ ] a = { 'J', 'a ‘ , 'v', 'a' };
      System.out.println(String.valueOf(a));
    }
}
```

```
123
true
3.0
c
Java
```

# 7) 기타 메소드

- boolean startsWith(String prefix)
  - ✓ prefix로 시작하면 true를 리턴함
- boolean endsWith(String suffix)
  - ✓ suffix로 끝나면 true를 리턴함
- char[ ] toCharArray( )

# 3. StringBuffer 클래스

3. StringBuffer 클래스

# 1) StringBuffer 클래스와 생성자

- 객체 생성 이후 문자열을 수정할 수 있는 기능을 제공
  - ✓ StringBuffer는 내용 변경이 가능한 mutable 클래스
- 내부적으로 문자열을 저장하기 위해 크기가 조절되는 버퍼를 사용함

- 생성자
  - ✓ StringBuffer( )
    - 초기 버퍼의 크기는 16
  - ✓ StringBuffer(int length)
  - ✓ StringBuffer(String str)
    - 초기 버퍼의 크기는 (str의 길이+16)

3. StringBuffer 클래스

# 2) 주요 메소드(1)

- int capacity( ), int length( )
- char charAt(int index), int indexOf(String str)
- String substring(int start, int end)
- StringBuffer append(char c)
  - ✓ 인자를 String 표현으로 바꾸고 원 문자열 끝에 추가하여 반환함
  - ✓ 인자는 char[ ], Object, String, 기본 자료형도 가능함

```
StringBuffer s1 = new StringBuffer("start");
System.out.println(s1.capacity( ));
System.out.println(s1.length( ));
System.out.println(s1.append("le" )); //"startle"
```

# 2) 주요 메소드(2)

- StringBuffer delete(int start, int end)
  - ✓ start 위치에서 (end-1)까지의 문자열을 삭제
- StringBuffer insert(int offset, String s)
  - ✓ offset 위치부터 s를 삽입
- StringBuffer replace(int start, int end, String s)
  - ✓ start 위치부터 (end-1)까지의 문자열을 s로 교체
- StringBuffer reverse( )
  - ✓ 문자열을 역순으로 변경

# 3) String 클래스를 사용할 때의 문제점

◆ 문자열을 빈번하게 변경하는 프로그램

✓ String은 immutable 클래스

✓ 기존 String 객체는 놔둔 채 새로운 String 객체가 계속 생성됨

```
public class StringTest {
   public static void main(String args[ ]) {
      final String aValue = "abcde";
      String str = new String( );

      for (int i = 0; i < 1000; i++)
         str = str + aValue;
      System.out.println(str);
   }
}
```

## 3. StringBuffer 클래스

# 4) StringBuffer와 StringBuilder의 사용

```
public class StringTest2 {
   public static void main(String args[]) {
         final String tmp = "abcde";    long start, end;

         String str = new String( );
         StringBuffer sb1 = new StringBuffer( );
         StringBuilder sb2 = new StringBuilder( );

         start = System.nanoTime( );
         for(int i = 0; i < 10000; i++) str = str + tmp;
         end = System.nanoTime( );
         System.out.println((end-start)/1000000.0 +" msecs");

         start = System.nanoTime( );
         for(int i = 0; i < 10000; i++) sb1.append(tmp);
         end = System.nanoTime( );
         System.out.println((end-start)/1000000.0 +" msecs");

         start = System.nanoTime();
         for(int i = 0;i < 10000; i++) sb2.append(tmp);
         end = System.nanoTime();
         System.out.println((end-start)/1000000.0 +" msecs");
   }
}
```

```
402.328576 msecs
0.79763 msecs
0.416001 msecs
```

# 4. 포장 클래스

4. 포장 클래스

# 1) 포장 클래스

◆ 기본형을 참조형으로 표현하기 위한 클래스

✓ 기본형의 값을 가지고 객체로 포장(boxing)함

| 구분 | 정수형 | | | | 실수형 | | 문자형 | 논리형 |
|---|---|---|---|---|---|---|---|---|
| 기본형 | byte | short | int | long | float | double | char | boolean |
| 참조형 | **Byte** | **Short** | **Integer** | **Long** | **Float** | **Double** | **Character** | **Boolean** |

사용 목적

◆ 메소드의 인자로 객체가 필요할 때

◆ 클래스가 제공하는 상수를 사용할 때

✓ Integer.MIN_VALUE, Integer.MAX_VALUE 등

◆ 클래스가 제공하는 다양한 메소드를 사용할 때

# 2) Number 클래스

- Number는 Byte, Short, Integer, Long, Float, Double의 추상 부모 클래스
- Number의 자식 클래스에서 구현된 주요 메소드
  - ✓ byte byteValue( ), short shortValue( ), …
    - 객체를 해당 기본형의 숫자로 변환(unboxing)
  - ✓ int compareTo(Byte anotherByte), …
    - this와 인자를 비교하여 같으면 0을 리턴
  - ✓ boolean equals(Object obj)
    - 같은 유형이고, 값이 같으면 true를 리턴

# 3) String과 기본형 데이터 간의 변환

◆ 포장 클래스가 제공하는 static 메소드를 사용함

◆ String을 int(또는 long)형으로 변환할 때

- ✓ int n = Interger.parseInt("123");
- ✓ long l = Long.parseLong("1234");

◆ int형(또는 long)을 String 형으로 변환

- ✓ String s1 = Integer.toString(4);
- ✓ String s2 = Long.toString(5);
- ✓ String s3 = String.valueOf(123);

# 4) Integer 클래스

- Integer, String, int 사이의 변환 기능을 제공
- 다른 클래스들도 유사한 기능을 제공함

주요 메소드

- static int parseInt(String s) : String을 int로
- static String toString(int i) : int를 String으로
- static Integer valueOf(int i) : int를 Integer로
- String toString( ) : Integer를 String으로
- static Integer valueOf(String s) : String을 Integer로

# 5) 박싱

- 기본형 데이터를 포장 클래스의 객체로 변환하는 것
- 예
  - ✓ Double radius = new Double(2.59); //생성자 사용, 구식
  - ✓ Double radius = Double.valueOf(10.4); //valueOf( ) 사용
  - ✓ Double radius = 2.59; //자동 boxing

**자동 박싱**

- 기본형에서 포장 클래스의 객체로 자동 변환되는 것
- 인자에 전달되거나 변수에 대입될 때 적용됨

# 6) 언박싱

- 포장 클래스의 객체를 기본형 데이터로 변환하는 것
- 포장 클래스에서 기본형Value( ) 메소드를 사용
- 예
  - ✓ radius는 Double형 객체라 가정
  - ✓ double r = radius.doubleValue( ); //객체.기본형Value( )
  - ✓ double r = radius; //자동 unboxing
  - ✓ System.out.println(new Integer(3) % 2); //자동 unboxing

**자동 언박싱**

- 포장 클래스의 객체에서 기본형으로 자동 변환되는 것
- 인자에 전달되거나 변수에 대입될 때 적용됨

# 5. System 클래스

# 1) System 클래스

- Java 플랫폼 및 시스템과 관련된 기능 제공
  - ✓ 유용한 클래스 필드와 메소드를 가짐
  - ✓ 모든 멤버는 static, 사용 시 객체를 생성할 필요 없음

주요 기능

- 표준 입출력
- JVM 또는 운영체제 속성과 시스템 환경 변수의 사용
- 배열 복사
- …

# 2) System 클래스의 표준 입출력 필드

- System.in
  - ✓ 표준 입력 스트림으로 InputStream 유형
  - ✓ 키보드로부터 입력을 받을 때 사용
    - System.in.read( )는 키보드로부터 1바이트 문자를 입력 받음
- System.out
  - ✓ 표준 출력 스트림으로 PrintStream 유형
  - ✓ 화면에 데이터를 출력할 때 사용
- System.err
  - ✓ 표준 에러 출력 스트림으로 PrintStream 유형
  - ✓ 오류 메시지를 화면에 출력할 때 사용

# 3) 키보드로부터 문자 입력받기

```
import java.io.*;
public class ExInput {
   public static void main(String args[ ]) throws IOException {
        char cInput = 0;
        int i;
        System.out.print("Input a character: ");
        cInput = (char)System.in.read( );
        System.out.println(cInput);
        i = System.in.read( );
        System.out.println(i);
        i = System.in.read( );
        System.out.println(i);
        System.out.println(System.in.available( ));
   }
}
```

```
Input a character: a↵
a
13
10
0
```

## 5. System 클래스

# 4) 여러 자리 숫자 입력받기

```
import java.io.*;

public class ExInout4 {
   public static void main(String args[]) throws IOException {
        String szInputLine;
        int nValue;
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader is = new BufferedReader(isr);
        System.out.print("Input a number: ");
        szInputLine = is.readLine( );
        nValue = Integer.parseInt(szInputLine);
        System.out.println(nValue);
        is.close();
   }
}
```

```
Input a number: 134↵
134
```

Java프로그래밍

다음시간안내

# 9강. java.io 패키지와 스트림