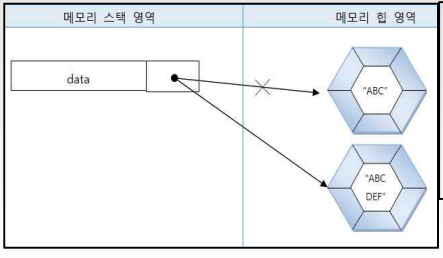
제18장
java.lang패키지-기타 클래스들

1. StringBuffer, StringBuilder클래스

▣ 기존의 String의 문자열은 내부적으로 문자열의 수정이 불가하다. 하여, 대체 String객체로 대신 리턴을 하게 되어 있다.

```
String data = "ABC";
data += "DEF";
```



String클래스는 +연산을 하게 되면 heap영역에 새로운 인스턴스가 생성되며, 아울러 참조하게 되는 것이다. 그리고, 참조를 잃은 객체는 GC에 의해 제거가 되는 것이다.

1. StringBuffer, StringBuilder클래스

▣ 버퍼(buffer)라는 것은 데이터를 임시로 저장하기 위한 작은 공간을 의미한다.
하지만, 버퍼를 사용하면 상당히 입출력 속도가 빨라진다.

▣ StringBuffer와 StringBuilder클래스는 버퍼를 이용하여, 객체가 만들어지는데
String클래스와는 달리 버퍼 내부에서 문자열의 추가, 수정, 삭제가 가능하다.
단, 인스턴스가 하나라는 것이 중요하다.

- StringBuffer : 멀티 스레드 환경에서 사용한다.
- StringBuilder : 싱글 스레드 환경에서 사용한다.

```
StringBuilder sBuilder = new StringBuilder("JAVA");
sBuilder.append(" P/G");
System.out.println(sBuilder);
```

```
Console
<terminated> CarExam
JAVA P/G
```

append(...)	문자열 붙임
insert(int offset, ...)	
delete(int start, int end)	
deleteCharAt(int index)	
replace(int start, int end, String str)	
StringBuilder reverse()	문자열 역출력
setCharAt(int index, char ch)	

2. Math클래스

▣ 수학에서 사용되어지는 각종 메서드들을 정적메서드로 제공하고 있는 클래스이다.

메소드	설명	예제 코드	리턴값
int abs(int a) double abs(double a)	절대값	int v1 = Math.abs(-5); double v2 = Math.abs(-3.14);	v1 = 5 v2 = 3.14
double ceil(double a)	올림값	double v3 = Math.ceil(5.3); double v4 = Math.ceil(-5.3);	v3 = 6.0 v4 = -5.0
double floor(double a)	버림값	double v5 = Math.floor(5.3); double v6 = Math.floor(-5.3);	v5 = 5.0 v6 = -6.0
int max(int a, int b) double max(double a, double b)	최대값	int v7 = Math.max(5, 9); double v8 = Math.max(5.3, 2.5);	v7 = 9 v8 = 5.3
int min(int a, int b) double min(double a, double b)	최소값	int v9 = Math.min(5, 9); double v10 = Math.min(5.3, 2.5);	v9 = 5 v10 = 2.5
double random()	랜덤값	double v11 = Math.random();	0.0<= v11<1.0
double rint(double a)	가까운 정수의 실수값	double v12 = Math.rint(5.3); double v13 = Math.rint(5.7);	v12 = 5.0 v13 = 6.0
long round(double a)	반올림값	long v14 = Math.round(5.3); long v15 = Math.round(5.7);	v14 = 5 v15 = 6

3. 포장(wrapper)클래스

▣ 'wrap'이란 '무엇을 감싼다'라는 뜻을 가지고 있다.

하여, 포장클래스는 기본타입(byte, short, char, long, float, double 등)의 값을 객체로 포장하는 클래스들이다.(컬렉션 프레임웍에서 사용하기 위한 용도)

기본 타입	포장 클래스
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

3. 포장(wrapper)클래스

▣ 박싱(boxing)과 언박싱(unboxing)의 개념은 아래와 같다.

- 박싱(boxing) : 기본형 타입을 포장객체로 만드는 것을 말한다.
- 언박싱(unboxing) : 포장객체에서 기본형 타입의 값을 가져오는 과정을 말한다.

▣ 박싱(boxing)을 하는 방법

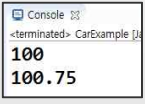
1) 생성자를 이용하는 방법

기본 타입의 값을 줄 경우	문자열을 줄 경우
Byte obj = new Byte(10);	Byte obj = new Byte("10");
Character obj = new Character('가');	
Short obj = new Short(100);	Short obj = new Short("100");
Integer obj = new Integer(1000);	Integer obj = new Integer("1000");
Long obj = new Long(10000);	Long obj = new Long("10000");
Float obj = new Float(2.5F);	Float obj = new Float("2.5F");
Double obj = new Double(3.5);	Double obj = new Double("3.5");
Boolean obj = new Boolean(true);	Boolean obj = new Boolean("true");

3. 포장(wrapper)클래스

2) valueOf() 메서드 이용하는 방법

```
Integer i = Integer.valueOf(100);
Double d = Double.valueOf(100.75);
System.out.println(i);
System.out.println(d);
```



▣ 언박싱(boxing)을 하는 방법

– 해당 포장클래스의 XXXvalue()메서드를 호출하하면 된다.

```
//언박싱 코드
int iVar = i.intValue();
double dVar = d.doubleValue();
```

▣ 자동 박싱(Auto-boxing)과 자동 언박싱(Auto-unboxing) 하는 방법

```
Integer obj = 100;
int iVar = obj;
```

3. 포장(wrapper)클래스

▣ 문자열을 기본 타입으로 변환하는 방법

– parse + 기본 타입명 으로 정적 메서드를 이용하면 된다.

기본 타입의 값을 이용		
byte	num	= Byte.parseByte("10");
short	num	= Short.parseShort("100");
int	num	= Integer.parseInt("1000");
long	num	= Long.parseLong("10000");
float	num	= Float.parseFloat("2.5F");
double	num	= Double.parseDouble("3.5");
boolean	bool	= Boolean.parseBoolean("true");

```
String str = "100";
int iVar = Integer.parseInt(str);
```

4. System 클래스

▣ System클래스는 통상 프로그램의 '시스템적인 부분을 담당한다' 라고 보면 된다.

– 프로그램 종료, 키보드로부터 입력, 모니터 출력, 메모리 정리, 시간 읽기 시스템 속성 읽기, 환경 변수 읽기 등이 있다.

▣ System.exit() 메서드

– exit()메서드는 강제적으로 JVM을 종료시키고자 할 때, 사용한다.

– 매개변수의 값으로 어떤 값을 주더라도 종료가 된다. 통상, 0을 준다.

```
System.exit(0);
```

▣ System.gc() 메서드

– JVM에게 가능한 빨리 GC()를 실행해 달라고 요청한다.

– GC()가 호출되면 바로 실행되는 것이 아니라, 최대한 빠른 시간 내 실행을 요청하는 것이다.

4. System 클래스

▣ System.currentTimeMillis() 메서드, System.nanoTime()메서드

- System.currentTimeMillis() : 시스템의 현재 시간을 읽어 밀리 세컨드(1/1000초) 단위로 돌려준다.
- System.nanoTime() : 나노 세컨드(1/10의 9승)단위로 돌려준다.
- 주로, 프로그램 성능 테스트 용도로 사용한다.

```
long milli = System.currentTimeMillis();
long nano = System.nanoTime();
```

4. System 클래스

▣ System.getProperty()메서드

– 시스템 property는 JVM이 시작할 때 자동으로 설정되는 시스템의 속성 값을 칭한다. 통상 키를 주고 값을 받아온다.

키(key)	설명	값(value)
java.version	자바의 버전	1.7.0_25
java.home	사용하는 JRE 의 파일 경로	<jdk 설치경로>₩jre
os.name	Operating system name	Windows 7
file.separator	File separator ("/" on UNIX)	₩
user.name	사용자의 이름	사용자계정
user.home	사용자의 홈 디렉토리	C:₩Users₩사용자계정
user.dir	사용자가 현재 작업 중인 디렉토리 경로	다양

– 시스템 property를 읽어오는 방법

```
String value = System.getProperty("os.name");
```

5. Class 클래스

▣ Class 클래스는 클래스와 인터페이스의 메타 데이터관리, 문자열로 된 클래스명으로 부터 동적 객체를 생성하는 클래스이다.

– 메타데이터 : 클래스의 이름, 생성자 정보, 필드, 메서드 정보 등

▣ getClass(), forName() – Class객체 얻는 방법

– 현재 객체로부터 Class를 얻는 방법(Object의 getClass()임)

```
Class class1 = account.getClass();
```

– 문자열로부터 Class를 얻는 방법

```
try {
    Class class1 = Class.forName("java.lang.String");
} catch (ClassNotFoundException e) {}
```

5. Class 클래스

▣ **리플렉션**

– 클래스의 생성자, 필드, 메서드 정보를 알아내는 것을 말한다.

```
Constructor[] con = clazz.getDeclaredConstructors();
Field[] field = clazz. getDeclaredFields();
Method[] methods = clazz.getDeclaredMethods();
```

▣ 동적 객체 생성

– 동적 객체란 프로그램이 실행 중에 클래스 이름이 결정되는 것을 말한다.

이런 경우 Class의 newInstance()를 이용하면 된다.

```
try {
    Class class1 = Class.forName("Taxi or Bus");
    Repairable repairable = (Repairable)class1.newInstance();
    repairable.repair();
} catch (Exception e) {}
```

Class의 newInstance()는 반환형이 Object클래스이다. 그래서 동적객체를 생성하기 위해 해당클래스로 강제 캐스팅을 해주어야 한다.(다형성)

감사합니다.