기본 API 클래스

주효진

CONTENTS

Chapter 12

Wrapper(포장) 클래스

Wrapper(포장) 클래스

- 포장객체란 기본 타입의 값을 갖는 객체를 생성하는 것
  포장하고 있는 기본 타입의 값은 외부에서 변경할 수 없고 변경하기 위해선 새로운 포장 객체를 만들어야한다.

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

박싱과 언박싱

- 박싱(Boxing) : 기본 타입의 값을 포장 객체로 만드는 과정

new 생성자를 이용

```
Integer num1 = new Integer(100);
Integer num2 = new Integer("100");

Character char1 = new Character('a');
```

valueOf() 메소드를 이용

```
Integer num1 = Integer.valueOf(100);
Integer num2 = Integer.valueOf("100");
```

- 언박싱(Unboxing) : 포장 객체에서 기본 타입의 값을 얻어내는 과정

  기본타입명+Value() 메소드 호출

  int num3 = num1.intValue();

자동 박싱과 언박싱

- 직접 박싱, 언박싱을 하지 않아도 자동적으로 박싱, 언박싱이 일어나는 경우

- 자동 박싱은 포장 클래스 타입에 기본값이 대입될 경우에 발생
  Integer num = 100; // 자동박싱

- 자동 언박싱은 기본 타입에 포장 객체가 대입될 경우에 발생
  Integer num = new Integer(200);
  int value1 = num; // 자동 언박싱 > Integer 객체를 int 타입 변수의 대입
  int value2 = num + 100; // 자동 언박싱 > Integer 객체와 int 타입의 연산 시 자동 언박싱

문자열을 기본 타입 값으로 변환

- 포장 객체 외에 문자열을 기본 타입으로 변환 할 때는 parse+기본타입명 메소드를 사용

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");

포장 값 비교

- 포장 객체의 내부 값을 비교하기 위해 ==와 != 연산자를 사용할 수 없음
  equals() 메소드를 사용해서 비교

Chapter 13

Math, Random 클래스

Math 클래스

- java.lang.Math 클래스는 수학 계산에 사용할 수 있는 메소드를 제공

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

Random 클래스

- java.util.Random 클래스는 난수를 얻어내기 위한 메소드를 제공

생성자	설명
Random()	호출시 마다 다른 종자값(현재시간 이용)이 자동 설정된다.
Random(long seed)	매개값으로 주어진 종자값이 설정된다.

리턴값	메소드(매개변수)	설명
boolean	nextBoolean()	boolean 타입의 난수를 리턴
double	nextDouble()	double 타입의 난수를 리턴(0.0<= ~ < 1.0)
int	nextInt()	int 타입의 난수를 리턴(-2^{32}<= ~ <=2^{32}-1);
int	nextInt(int n)	int 타입의 난수를 리턴(0<= ~ <n)

Chapter 14

Data, Calendar 클래스

Date 클래스

- 객체 간에 날짜 정보를 주고 받을 때 주로 사용
  Data() 생성자는 컴퓨터의 현재 날짜를 읽어 객체로 만든다.
  toString() 메소드를 이용하여 현재 날짜를 문자열로 얻을 수 있다.
  이 때 리턴되는 문자열은 영문이기 때문에 특정 문자열 포맷으로 얻고 싶다면 SimpleDataFormat 클래스를 사용

```
Date now = new Date();
String strNow1 = now.toString();
System.out.println(strNow1);

SimpleDateFormat sdf =
        new SimpleDateFormat("yyyy 년  MM 월  dd 일  hh 시  mm 분  ss 초");
String strNow2 = sdf.format(now);
System.out.println(strNow2);
```

【실행 결과】

Calendar 클래스

- 달력을 표현한 클래스로 추상 클래스이기 때문에 new 생성자를 이용해 인스턴스를 생성할 수 없다.
  getInstance() 메소드를 사용하면 현재 운영체제에 설정된 시간대를 기준으로 한 Calendar 하위 객체를 얻는다.

```
Calendar now = Calendar.getInstance();
```

- 현재 운영체제의 기준이 아닌 다른 지역의 시간대를 얻기 위해선 TimeZon을 설정해줘야 한다.

```
TimeZon timeZon = TimeZone.getTimeZone("America/Los_Angeles");
Calendar   now = Calendar.getInstance( timeZon );
```

- get() 메소드를 이용해서 날짜와 시간에 대한 정보를 읽을 수 있다.

```
int year      = now.get(Calendar.YEAR);              //년도를 리턴
int month     = now.get(Calendar.MONTH)  + 1;        //월을 리턴
int day       = now.get(Calendar.DAY_OF_MONTH);      //일을 리턴
int week      = now.get(Calendar.DAY_OF_WEEK);       //요일을 리턴
int amPm      = now.get(Calendar.AM_PM);             //오전/오후를 리턴
int hour      = now.get(Calendar.HOUR);              //시를 리턴
int minute    = now.get(Calendar.MINUTE);            //분을 리턴
int second    = now.get(Calendar.SECOND);            //초를 리턴
```

Chapter 15

Format 클래스

Format 클래스

- 문자나 숫자를 원하는 형식의 문자열로 변환하기 위해 사용
  숫자 DemimalFormat / 날짜 SimpleDateFormat / 매개변수화된 문자열 MessageFormat

기호	의미	패턴 예	결과
0	10진수 빈자리는 0	0 0.0 0000000000.00000	1234568 1234567.9 0001234567.89000
#	10진수 빈자리는 공백	# #.# ##########.#####	1234568 1234567.9 1234567.89
.	소수점	#.0	1234567.9
-	음수 기호	+#.0 -#.0	+1234567.9 -1234567.9
,	단위 구분	#,###.0	1,234,567.9
E	지수문자	0.0E0	1.2E6
;	양수 음수 패턴 구분자	+#,### : -#,###	+1234567.9 -1234567.9
%	100을 곱한 후 %를 붙임	#.# %	123456789 %
₩u00A4	통화기호	₩u00A4 #,###	₩ 1,234,568

숫자 형식 클래스

- Format 클래스는 문자나 숫자를 원하는 형식의 문자열로 변환하기 위해 사용

기호	의미	패턴 예	결과
0	10진수 빈자리는 0	0 0.0 0000000000.00000	1234568 1234567.9 0001234567.89000
#	10진수 빈자리는 공백	# #.# ##########.#####	1234568 1234567.9 1234567.89
.	소수점	#.0	1234567.9
-	음수 기호	+#.0 -#.0	+1234567.9 -1234567.9
,	단위 구분	#,###.0	1,234,567.9
E	지수문자	0.0E0	1.2E6
;	양수 음수 패턴 구분자	+#,### : -#,###	+1234567.9 -1234567.9
%	100을 곱한 후 %를 붙임	#.# %	123456789 %
₩u00A4	통화기호	₩u00A4 #,###	₩ 1,234,568

날짜 형식 클래스

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy 년 MM 월 dd 일");
String strDate = sdf.format(new Date());
```

패턴 문자	의미	패턴 문자	의미
y	년	H	시(0~23)
M	월	h	시(1~12)
d	일	K	시(0~11)
D	월 구분이 없는 일(1~365)	k	시(1~24)
E	요일	m	분
a	오전/오후	s	초
w	년의 몇번째 주	S	밀리세컨드(1/1000 초)
W	월의 몇번째 주		

문자열 형식 클래스

- 데이터를 파일에 저장, 네트워크로 데이터 전송, 데이터베이스 SQL문 작성 시 사용

```java
public class MessageFormatExample {

    public static void main(String[] args) {
        String id = "java";
        String name = "신용권";
        String tel = "010-1234-5678";

        String text = "회원 ID: {0} \n회원 이름: {1} \n회원 전화: {2}";
        String result1 = MessageFormat.format(text, id, name, tel);
        System.out.println(result1);
        System.out.println();

        String sql = "insert into member value( {0}, {1}, {2} )";
        Object[] arguments = {"java", "신용권", "010-1234-5678"};
        String result2 = MessageFormat.format(sql, arguments);
        System.out.println(result2);
    }

}
```

```
회원 ID: java
회원 이름: 신용권
회원 전화: 010-1234-5678

insert into member value( java, 신용권, 010-1234-5678 )
```

Chapter 16

java.time 패키지

java.time 패키지

- 자바 7 이전까지는 Date와 Calendar 클래스를 이용하여 날짜와 시간 정보를 얻을 수 있었지만
  자바 8 이후로 java.time 패키지가 추가되면서 날짜와 시간을 조작하거나 비교하는 기능이 추가

패키지	설명
java.time	날짜와 시간을 나타내는 핵심 API 인 LocalDate, LocalTime, LocalDateTime, ZonedDateTime 을 포함하고 있다. 이들 클래스는 ISO-8601 에 정의된 달력 시스템에 기초한다.
java.time.chrono	ISO-8601 에 정의된 달력 시스템 이외에 다른 달력 시스템을 사용코저할때 사용할 수 있는 API 들이 포함되어 있다.
java.time.format	날짜와 시간을 파싱하고 포맷팅하는 API 들이 포함되어 있다.
java.time.temporal	날짜와 시간을 연산하기 위한 보조 API 들이 포함되어 있다.
java.time.zone	타임존을 지원하는 API 들이 포함되어 있다.

날짜와 시간 객체 생성

클래스명	설명
LocalDate	로컬 날짜 클래스
LocalTime	로컬 시간 클래스
LocalDateTime	로컬 날짜 및 시간 클래스(LocalDate + LocalTime)
ZonedDateTime	특정 타임존(TimeZone)의 날짜와 시간 클래스
Instant	특정 시점의 Time-Stamp 클래스

날짜와 시간에 대한 정보 얻기

클래스	리턴타입	메소드(매개변수)	설명
LocalDate	int	getYear()	년
	Month	getMonth()	Month 열거값
	int	getMonthValue()	월
	int	getDayOfYear	일년의 몇번째 일
	int	getDayOfMonth()	월의 몇번째 일
	DayOfWeek	getDayOfWeek()	요일
	boolean	isLeapYear()	윤년 여부
LocalTime	int	getHour()	시간
	int	getMinute()	분
	int	getSecond()	초
	int	getNano()	나노초 리턴

클래스	리턴타입	메소드(매개변수)	설명
ZonedDateTime	ZoneId	getZone()	존아이디를 리턴 (예: Asia/Seoul)
	ZoneOffset	getOffset()	존오프셋(시차)을 리턴

빼기와 더하기

- 빼기
  minus + 변수 (long)의 형태
  minusYears(long) 년 빼기
  minusHours(long) 시간 빼기

- 더하기
  plus + 변수 (long)의 형태
  plusYears(long) 년 더하기
  plusHours(long) 시간 더하기

변경하기

- withYear(int) 년 변경
  withHour(int) 시간 변경

- with(TemporalAdjuster adjuster)
  now.with(TemporalAdjusters.firstDayOfYear();

메소드(매개변수)	설명
firstDayOfYear()	이번 해의 첫일
lastDayOfYear()	이번 해의 마지막 일
firstDayOfNextYear()	다음 해의 첫일
firstDayOfMonth()	이번 달의 첫일
lastDayOfMonth()	이번 달의 마지막 일
firstDayOfNextMonth()	다음 달의 첫일
firstInMonth(DayOfWeek dayOfWeek)	이번 달의 첫 요일
lastInMonth(DayOfWeek dayOfWeek)	이번 달의 마지막 요일
next(DayOfWeek dayOfWeek)	돌아오는 요일
nextOrSame(DayOfWeek dayOfWeek)	돌아오는 요일(오늘 포함)
previous(DayOfWeek dayOfWeek)c	지난 요일
previousOrSame(DayOfWeek dayOfWeek)	지난 요일(오늘 포함)

날짜와 시간을 비교하기

클래스	리턴타입	메소드(매개변수)	설명
LocalDate LocalDateTime	boolean	isAfter(ChronoLocalDate other)	이후 날짜인지?
		isBefore(ChronoLocalDate other)	이전 날짜인지?
		isEqual(ChronoLocalDate other)	동일 날짜인지?
LocalTime LocalDateTime	boolean	isAfter(LocalTime other)	이후 시간인지?
		isBefore(LocalTime other)	이전 시간인지?
LocalDate	Period	until(ChronoLocalDate endDateExclusive)	날짜 차이
LocalDate LocalTime LocalDateTime	long	until(Temporal endExclusive, TemporalUnit unit)	시간 차이
Period	Period	between(LocalDate startDateInclusive, LocalDate endDateExclusive)	날짜 차이
Duration	Duration	between(Temporal startInclusive, Temporal endExclusive)	시간 차이
ChronoUnit.YEARS	long	between(Temporal temporal1Inclusive, Temporal temporal2Exclusive)	전체 년 차이
ChronoUnit.MONTHS			전체 달 차이
ChronoUnit.WEEKS			전체 주 차이
ChronoUnit.DAYS			전체 일 차이
ChronoUnit.HOURS			전체 시간 차이
ChronoUnit.SECONDS			전체 초 차이
ChronoUnit.MILLIS			전체 밀리초 차이
ChronoUnit.NANOS			전체 나노초 차이

파싱과 포맷팅

- 파싱 : 문자를 파싱해서 날짜와 시간을 생성하는 메소드

클래스	리턴타입	메소드(매개변수)
LocalDate LocalTime LocalDateTime ZonedDateTime	LocalDate LocalTime LocalDateTime ZonedDateTime	parse(CharSequence)
		parse(CharSequence, DateTimeFormatter)

- 포맷팅 : 날짜와 시간을 포맷팅된 문자열로 변환

클래스	리턴타입	메소드(매개변수)
LocalDate LocalTime LocalDateTime ZonedDateTime	String	format(DateTimeFormatter formatter)

감사합니다.