제 6 장
유용한 참조타입과 배열
2부-Number 클래스

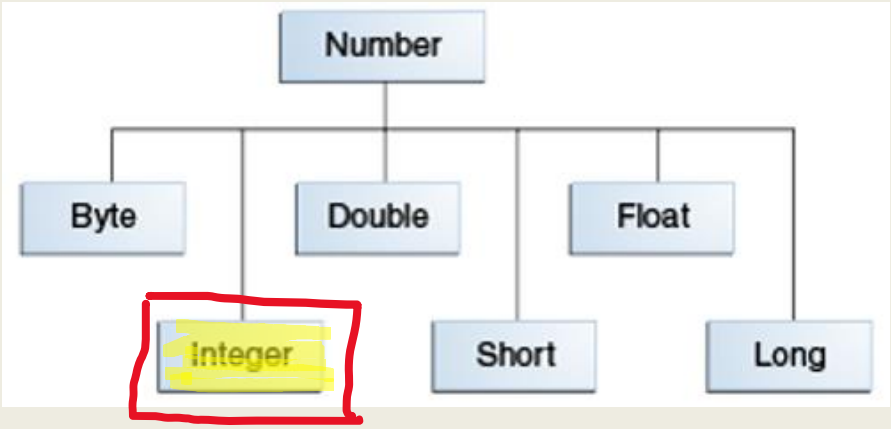
참조타입

- 참조(reference)타입, 객체(object)타입의 변수
  - 정보가 저장된 (위치를 가리키는) 주소를 저장하고 있는 유형의 변수 (cf. 기본 데이터 타입)
  - 모든 객체는 참조타입의 변수로써 할당된다
- 자주 사용하는 참조 타입의 객체
  - **String**
  - **Number**
    - wrapper 클래스 : 기본 데이터 타입을 참조타입의 객체로 변환
    - 자동박싱(Autoboxing)과 언박싱(Unboxing)
  - **Math** : 각종 연산 메소드를 포함하고 있는 클래스
  - **배열** 객체
  - **Enum** 객체

Number 클래스

- 프로그래밍에 있어서 수를 처리하는 경우에, 대부분은 기본 데이터 타입을 사용하는 것이 보통, 그러나 기본 타입 대신에 객체 타입을 사용해야 하는 경우가 종종 존재
  - 객체를 인수로 요구하는 메소드를 사용하고자 하는 경우
    - 컬렉션 클래스의 경우 대부분 객체를 인수로 요구하는 메소드를 포함
  - MAX_VALUE, MIN_VALUE 등과 같이, 클래스에 정의된 상수를 사용하고자 하는 경우
  - 기본 타입을 다른 타입으로, 혹은 어떤 타입을 다른 타입으로 변환하는 클래스에 정의된 메소드를 사용하고자 하는 경우
    - 스트링을 정수로 혹은 정수를 스트링으로,
    - 10진수를 16진수로, 혹은 16진수를 2진수로 등 진법 변환을 원하는 경우 등

- 자바는 각 기본 타입에 대응하는 **참조 타입의 객체**로서 래퍼 (wrapper) 클래스를 제공
- 수를 타내는 각 기본 타입에 대한 **래퍼 클래스**의 계층도



- 각 래퍼클래스는 Number 클래스의 서브클래스이다
- 수는 아니지만 자바는 모든 기본 타입에 대해 래퍼 클래스를 지원
  - char → Character
  - boolean → Boolean

기본타입(Primitive type)	래퍼클래스(Wrapper class)
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

[표 6-10] Integer 클래스가 제공하고 있는 변환 메소드

메소드	설 명
static Integer decode(String s)	인수로 받은 스트링을 정수로 변환한 Integer 객체를 리턴하며, 인수는 10진수, 8진수, 16진수 스트링을 받을 수 있다
static int parseInt(String s)	인수로 받은 스트링을 10진 정수로 변환하여 리턴
static int parseInt(String s, int radix)	10진, 2진, 8진, 16진수의 스트링을 그에 대한 10진 정수로 변환한 값을 리턴
String toString()	정수의 값에 대한 스트링 객체를 리턴
static String toString(int i)	명시된 정수 타입의 값에 대한 스트링 객체를 리턴
static Integer valueOf(int i)	명시된 정수 타입의 값을 포함하는 Integer 객체 리턴
static Integer valueOf(String s)	명시된 스트링 값을 포함하는 Integer 객체 리턴
static Integer valueOf(String s, int radix)	명시된 radix 진법의 스트링 s를 10진법으로 변환한 Integer 객체를 리턴. 예를 들어 s="333"이고 radix=8이라면 8진수 333에 대응하는 10진 정수를 리턴

이외에도 비교 및 값 변환에 관한 다양한 메소드를 지원(*[표 6-9]를 참조)

자동박싱(Autoboxing)

- 기본 타입을 그에 대응하는 래퍼 클래스로 변환
- 자바 컴파일러에 의해 자동으로 이루어지는 타입 변환

```
Character ch = 'a';
```

- 기본 문자 타입인 'a'를 그에 대응하는 래퍼 클래스인 Character 클래스에 할당하는 것은 문법적으로 맞지 않지만, 컴파일러가 자동으로 래퍼 클래스로 변환하여 할당
- 컴파일러는 다음과 같은 경우 자동박싱을 적용
  - 대응하는 래퍼 클래스의 참조 변수로 기본 타입의 값을 할당할 때 동작한다
  - 래퍼 클래스의 객체를 매개변수로 요구하는 메소드의 매개변수로 기본 타입의 값을 넘길 때

- 자동박싱 예

```
public static int sumEven(List<Integer> mylist)
{
    int sum = 0;
    for (Integer i: mylist)     // for-each 반복문
        if (i % 2 == 0)
            sum += i;
        return sum;
}
```

```
public static int sumEven(List<Integer> mylist) {
    int sum = 0;
    for (Integer i : mylist)
        if (i.intValue() % 2 == 0)
            sum += i.intValue();
        return sum;
}
```

- **언박싱(Unboxing) – 자동박싱의 역**
  - 래퍼 타입의 객체를 그에 대응하는 기본 타입의 값으로 변환
  - 컴파일러는 다음과 같은 경우 자동 언박싱을 적용
    - 기본 타입이 요구되는 경우인데 불구하고, 래퍼 클래스의 객체가 메소드의 매개변수로 넘겨질 때,
    - 래퍼 클래스 객체가 기본 타입의 변수로 할당되는 경우

```
import java.util.ArrayList;
import java.util.List;
public class Unboxing {
    public static void main(String[] args) {
        Integer i = new Integer(-8);
        // 1. 언박싱
        int absVal = absoluteValue(i);      //기본타입이 요구되는데 래퍼 객체를 인수로 사용
        System.out.println("absolute value of " + i + " = " + absVal);

        List<Double> ld = new ArrayList<>();
        ld.add(3.1416);       // 자동박싱 – 객체를 요구하는데 기본 타입을 사용
        // 2. 언박싱 through assignment
        double pi = ld.get(0);    // 기본타입을 할당해야 하는데 래퍼 객체를 할당
        System.out.println("pi = " + pi);
    }
    public static int absoluteValue(int i) {
        return (i < 0) ? -i : i;
    }
}
```

```
absolute value of -8 = 8
pi = 3.1416
```

- **Math 클래스**
  - 상수와 기본 메소드
    - 2개의 상수(자연상수 e(Math.E), 원주율 pi(Math.PI))와 40여개 이상의 정적 수학 메소드
  - 지수함수와 로그함수 메소드
  - 삼각함수 메소드
  - 난수함수 메소드 random() – <cf. Random 클래스 제공>
    - 메소드 random()은 0.0과 1.0 사이의 pseudo 난수를 리턴
    - 범위는 0.0은 포함하지만 1.0은 포함하지 않는 범위
      - 즉 0.0 <= Math.random() < 1.0
      - 다른 범위의 값을 얻으려면 리턴 값을 대상으로 스케일 변환 필요
      - 예: 0과 10사이의 정수를 얻고자 할 때
        - **int number = (int)(Math.random() * 10);**

[표 6.12] 기본적인 Math 메소드들

메소드	설 명
double abs(double d) float abs(float f) int abs(int i) long abs(long lng)	주어진 타입의 인수의 절대값을 리턴
double ceil(double d)	주어진 인수 이상의 정수 중에서 가장 작은 정수를 double 타입으로 리턴
double floor(double d)	주어진 인수 이하의 정수 중에서 가장 큰 정수를 double 타입으로 리턴
double rint(double d)	주어진 인수에 가장 근접한 정수를 double 타입으로 리턴
long round(double d) int round(float f)	주어진 인수에 가장 근접한 정수를 long (int) 타입으로 리턴
double min(double arg1, double arg2) float min(float arg1, float arg2) int min(int arg1, int arg2) long min(long arg1, long arg2)	주어진 2 인수 중에서 더 작은 인수를 각 해당 타입으로 리턴
double max(double arg1, double arg2) float max(float arg1, float arg2) int max(int arg1, int arg2) long max(long arg1, long arg2)	주어진 2 인수 중에서 더 큰 인수를 각 해당 타입으로 리턴

[표 6.13] 지수 함수와 로그 함수

Method	Description
double exp(double d)	자연로그의 밑수인 e에 대해 ed를 리턴
double log(double d)	주어진 인수의 자연로그 값(loged)를 리턴
double pow(double base, double exponent)	(base)exponent 값을 리턴
double sqrt(double d)	$\sqrt{d}$ 를 리턴

[표 6.14] 삼각함수 메소드

메소드	설 명
double sin(double d)	명시된 double 타입 d에 대한 sine 값을 리턴
double cos(double d)	명시된 double 타입 d에 대한 cosine 값을 리턴
double tan(double d)	명시된 double 타입 d에 대한 tangent 값을 리턴
double asin(double d)	명시된 double 타입 d에 대한 arcsine 값을 리턴
double acos(double d)	명시된 double 타입 d에 대한 arccosine 값을 리턴
double atan(double d)	명시된 double 타입 d에 대한 arctangent 값을 리턴
double atan2(double y, double x)	직각 좌표 (x, y)를 극좌표 (r,)로 변환하고 를 리턴한다
double toDegrees(double d) double toRadians(double d)	주어진 인수를 각도 값 혹은 라디안 값으로 변환.

- **enum 타입 : 클래스**
  - 하나의 변수를 미리 정의된 상수들의 집합과 대응하여 효율적으로 사용할 수 있도록 하는 특별한 참조형 데이터 타입
  - Enum 타입의 변수는 미리 정의된 값 중의 하나와 반드시 일치
    - 예) 나침반 방위, 요일, 무지개색, 태양계 행성
  - 값이 상수이므로 "enum" 타입 필드의 이름은 대문자를 사용
  - 선언 예

```
public enum Day {
SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}
```

  - 정해진 수의 상수들의 집합을 나타낼 필요가 있다면, enum 타입을 사용하는 것이 바람직하다

```
public enum Day {
SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}

public class EnumTest {
    Day day;
    public EnumTest(Day day) {
        this.day = day;
    }
    public void tellWhatDayIs() {
        switch (day) {
            case MONDAY:
                System.out.println("Mondays are bad.");
                break;
            case FRIDAY:
                System.out.println("Fridays are better.");   }
                break;
            case SATURDAY:
            case SUNDAY:
                System.out.println("Weekends are best.");
                break;
            default:
                System.out.println("Midweek days are so-so.");
                break;
        }
    }

    public static void main(String[] args)  {
        EnumTest firstDay = new EnumTest(Day.MONDAY);
        firstDay.tellWhatDayIs();
        EnumTest thirdDay = new EnumTest(Day.WEDNESDAY);
        thirdDay.tellWhatDayIs();
        EnumTest fifthDay = new EnumTest(Day.FRIDAY);
        fifthDay.tellWhatDayIs();
        EnumTest sixthDay = new EnumTest(Day.SATURDAY);
        sixthDay.tellWhatDayIs();
        EnumTest seventhDay = new EnumTest(Day.SUNDAY);
        seventhDay.tellWhatDayIs();
    }
```

```
Mondays are bad.
Midweek days are so-so.
Fridays are better.
Weekends are best.
Weekends are best.
```

- enum의 선언 == enum 타입의 클래스를 정의
  - enum 타입의 클래스는 **메소드와 필드를 포함**할 수 있다
  - 필드와 메소드가 존재할 경우 enum 상수들의 리스트는 ';' 으로 끝난다
  - 비록 어떤 메소드나 필드가 선언되어 있지 않더라도, 컴파일러는 선언된 enum 클래스의 객체를 생성할 때, 자동으로 몇 가지 메소드를 추가한다
  - 예 : 메소드 values()
    - enum의 모든 값을 선언된 순서로 포함하는 배열을 리턴
    - for-each 반복문과 조합하여 enum 의 각 값에 대해 효율적인 반복 동작 수행
  - enum 클래스는 묵시적으로 java.lang.Enum 클래스로부터 상속을 받는다
    - 그러므로 enum 클래스는 그 외 어떤 것으로부터도 상속 받을 수 없다

- 각 enum 상수는 소괄호를 이용하여 매개변수를 위한 값을 가지고 선언될 수 있으며, 이는 enum 객체가 생성될 때, 생성자에게로 전달된다
- enum 클래스의 생성자는 자동으로 enum 클래스에 정의된 상수 객체들을 생성한다.
  - 그러므로 enum 생성자를 따로 구동할 필요가 없다

```
public enum Planet {
    MERCURY (3.303e+23, 2.4397e6),
    VENUS (4.869e+24, 6.0518e6),
    EARTH (5.976e+24, 6.37814e6),
    MARS (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27, 7.1492e7),
    SATURN (5.688e+26, 6.0268e7),
    URANUS (8.686e+25, 2.5559e7),
    NEPTUNE (1.024e+26, 2.4746e7) ;

    private final double mass; // in kilograms
    private final double radius; // in meters

    // 자동으로 실행
    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
    }

    private double mass() { return mass; }
    private double radius() { return radius; }

    // 중력 상수
    public static final double G = 6.67300E-11;
```

```
    // 지표면 중력
    double surfaceGravity() {
        return G * mass / (radius * radius);
    }
    // 지표면 무게
    double surfaceWeight(double otherMass) {
        return otherMass * surfaceGravity();
    }
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java Planet <earth_weight>");
            System.exit(-1);
        }
        double earthWeight = Double.parseDouble(args[0]);
        double mass = earthWeight/EARTH.surfaceGravity();
        for (Planet p : Planet.values())
            System.out.printf("Your weight on %s is %f\n", p,
                                                p.surfaceWeight(mass));
    }
}
```

실행결과 예

```
$ java Planet 175

Your weight on MERCURY is 66.107583

Your weight on VENUS is 158.374842

Your weight on EARTH is 175.000000

Your weight on MARS is 66.279007

Your weight on JUPITER is 442.847567

Your weight on SATURN is 186.552719

Your weight on URANUS is 158.397260

Your weight on NEPTUNE is 199.207413
```