객체지향 프로그래밍

Object-oriented Programming

Chapter 07.
인터페이스와 특수 클래스

- 추상 클래스
- 인터페이스 기본
- 인터페이스 응용
- 인터페이스와 다형성
- 중첩 클래스와 중첩 인터페이스
- 익명 클래스

중첩 클래스와 중첩 인터페이스

중첩 클래스

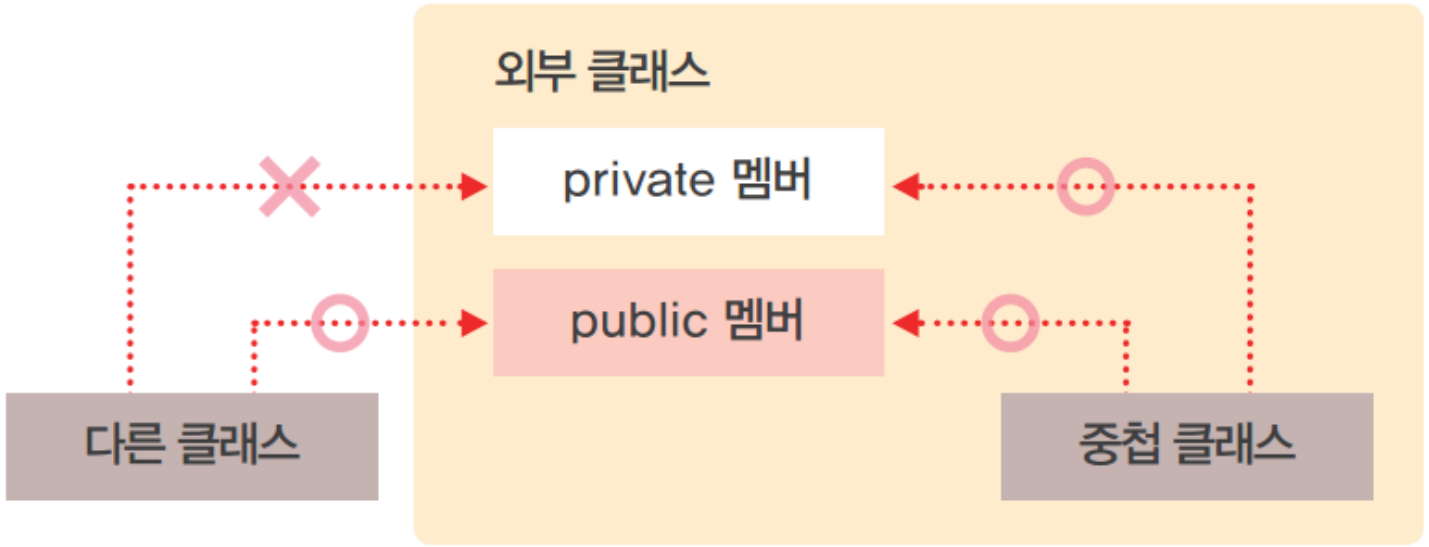
- 클래스 또는 메서드 내부에 정의된 클래스

```
public class Outer {

  public class Inner {

  }
}
```

```
void foo() {

    class Inner {
        String name = "unknown";

    }

    System.out.println(new Inner().name);
}
```

중첩 클래스와 중첩 인터페이스

중첩 클래스의 외부 필드 접근

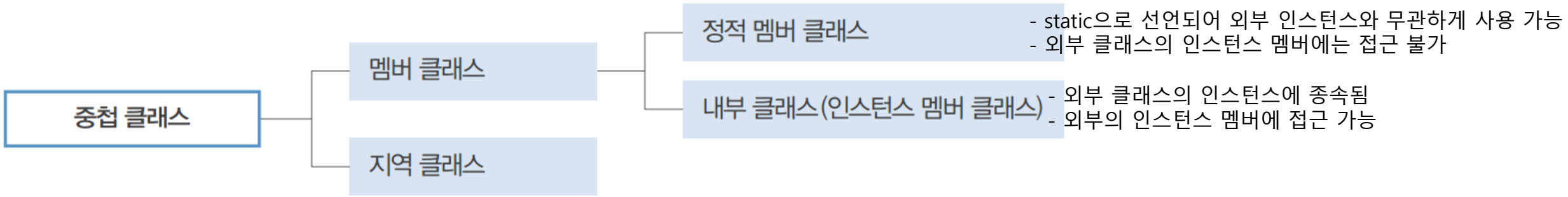


중첩 클래스는 외부 클래스를
상속할 필요 없이
외부 클래스의 private 멤버까지
사용할 수 있다.

중첩 클래스와 중첩 인터페이스

중첩 클래스의 종류

- **멤버 클래스**: 클래스 내부에 선언
- **지역 클래스**: 메서드 내부에 선언




```
class 외부클래스 {
   class 멤버클래스 {
   }

   interface 중첩인터페이스 {
   }
}
```

외부 클래스의 멤버로 선언된 클래스이다.

```
class 외부클래스 {
   void 메서드() {
      class 지역클래스 {
      }
   }
}
```

외부 클래스의 메서드 내부에 선언된 클래스이다.

중첩 클래스와 중첩 인터페이스

중첩 클래스

- 클래스 또는 메서드 내부에 정의된 클래스

```
public class Outer {

  public class Inner {

  }
}
```

```
void foo() {

    class Inner {
        String name = "unknown";

    }

    System.out.println(new Inner().name);
}
```

```java
class Outer {
    static class Inner {
        void show() {
            System.out.println("정적 멤버 클래스입니다.");
        }
    }
}

public class Test {
    public static void main(String[] args) {
        // 외부 인스턴스 없이 바로 생성 가능
        Outer.Inner inner = new Outer.Inner();
        inner.show();
    }
}
```

```java
class Outer {
    private String name = "Outer";

    // ✅ 인스턴스 멤버 클래스
    class Inner {
        void print() {
            // 외부 클래스의 인스턴스 멤버 접근 가능
            System.out.println("외부 이름: " + name);
        }
    }

    void makeInner() {
        Inner inner = new Inner();
        inner.print();
    }
}

public class Test {
    public static void main(String[] args) {
        Outer outer = new Outer();
        outer.makeInner(); // "외부 이름: Outer"
    }
}
```

중첩 클래스와 중첩 인터페이스

중첩 클래스 내부 코드 블록에서 외부 클래스 접근

- **방법 1**: 외부 클래스 이름을 명시해 외부 클래스의 필드, 메서드 접근
- **방법 2**: 명시 없이 직접 접근

```java
public class Outer {
    private int outerValue = 10;

    public class Inner {
        public void printOuterValue() {
            System.out.println("외부 클래스 값: " + outerValue); // 직접 접근 가능
            System.out.println("외부 클래스 값(명시적): " + Outer.this.outerValue);
        }
    }
}
```

중첩 클래스와 중첩 인터페이스

외부 클래스 객체 접근

외부클래스.this

```
public class Outer {
    private int outerValue = 10;

    public class Inner {
        public void printOuterValue() {
            System.out.println("외부 클래스 값: " + outerValue); // 직접 접근 가능
            System.out.println("외부 클래스 값(명시적): " + Outer.this.outerValue);
        }
    }
}
```

중첩 클래스와 중첩 인터페이스

멤버 클래스 객체 생성

- 멤버 클래스 유형별로 생성 방식이 상이 (정적 멤버 클래스 / 내부 클래스)
- 내부 클래스의 경우, 외부 클래스 객체를 참조하여 생성


```
외부클래스.내부클래스 변수 = 외부클래스의객체변수.new 내부클래스생성자();
외부클래스.정적멤버클래스 변수 = new 외부클래스.정적멤버클래스생성자();
```

Practice

중첩 클래스의 사용

예제:

- 7-14 (p. 288)
- 7-15 (p. 289)
- 7-16 (p. 290)

익명 클래스

익명 클래스

- **익명 클래스**: 중첩 클래스의 특수한 형태, 이름이 명시되지 않는 클래스
- 코드가 단순해지기 때문에 이벤트 처리나 스레드 등에서 자주 사용
- **한번만 사용되어 이름을 붙일 필요가 없을 때 사용 가능**

```
class OnlyOnce [ extends / implements ] Parent {
  // Parent가 클래스라면 오버라이딩한 메서드
  // Parent가 인터페이스라면 구현한 메서드
}

Parent p = new OnlyOnce();
```

```
Parent p = new Parent() {
  // Parent가 클래스라면 오버라이딩한 메서드
  // Parent가 인터페이스라면 구현한 메서드
};
```

무명 클래스 본체로서 OnlyOnce 클래스의 본체와 동일하다.

하나의 실행문이므로 세미콜론(;)으로 끝난다.

Practice

익명 클래스

예제:

- 7-17 (p. 291)
- 7-18 (p. 292)
- 7-19 (p. 293)
- 7-20 (p. 293)
- 7-21 (p. 294)

참고: 레코드

레코드 (Record)

- **레코드:** 필드의 유형과 이름만 필요한 불변 데이터를 위한 특별한 클래스
- final 클래스 (상속 불가)
- 모든 필드는 private final
- 주로 불변(immutable) 데이터 객체를 표현할 때 사용

```
public record Person(String name, int age) {
    public Person {
        if (age < 0) {
            throw new IllegalArgumentException("나이는 음수일 수 없습니다.");
        }
    }
}
```

참고: 봉인 클래스

봉인 클래스 (Sealed Class)

- sealed 키워드를 사용하여 상속을 제한하는 클래스
- 어떤 클래스가 상속할 수 있는지 명시적으로 지정
- Java 15에서 도입 → Java 17에서 정식 기능
- **클래스 계층 구조를 명확히 제어**

```java
public sealed class Shape permits Circle, Rectangle, Triangle {
    // 공통 속성 또는 메서드
}
```

```java
public sealed class Shape permits Circle, Rectangle, Triangle {
    public abstract double area();
}

public final class Circle extends Shape {
    private final double radius;
    public Circle(double radius) { this.radius = radius; }
    @Override public double area() { return Math.PI * radius * radius; }
}

public final class Rectangle extends Shape {
    private final double width, height;
    public Rectangle(double width, double height) {
        this.width = width; this.height = height;
    }
    @Override public double area() { return width * height; }
}

public non-sealed class Triangle extends Shape {
    private final double base, height;
    public Triangle(double base, double height) {
        this.base = base; this.height = height;
    }
    @Override public double area() { return 0.5 * base * height; }
}
```

Chapter 08. 기본 패키지

- 패키지와 API 문서
- java.lang 패키지
- java.util 패키지
- java.text 패키지

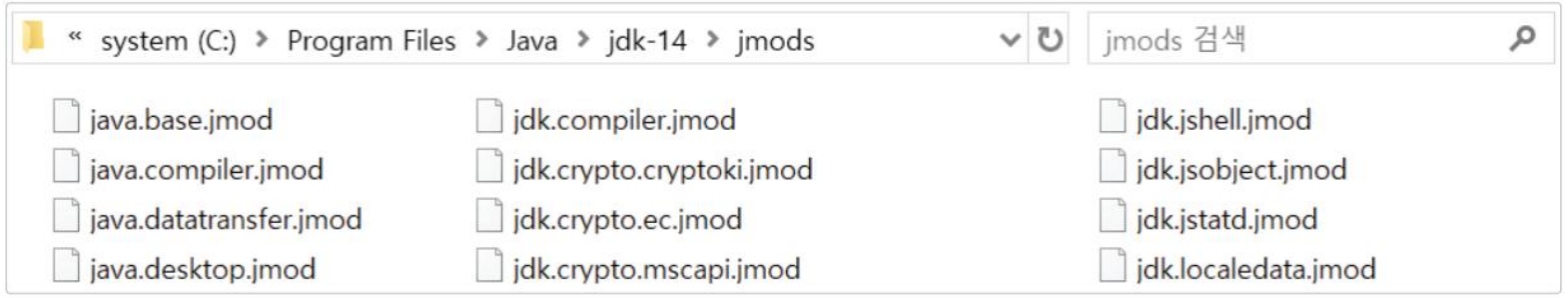
패키지와 API 문서

Java 라이브러리

- 개발자가 편리하게 사용할 수 있도록 패키지 혹은 모듈을 압축한 파일

패키지와 모듈

- **패키지**: 상호 관련 있는 클래스와 인터페이스를 한곳에 묶어 놓은 것
- **모듈**: 밀접한 관계가 있는 패키지와 리소스를 묶어 놓은 것.



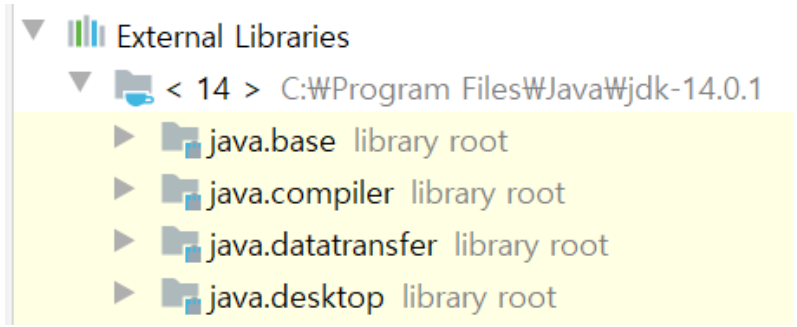
JDK 8까지는 개발자가 편리하게 프로그래밍할 수 있도록 기본 패키지를 rt.jar 파일로 제공,
JDK 9부터는 jmod 파일을 통하여 필요한 패키지를 제공

패키지와 API 문서

Java의 주요 패키지 및 모듈

패키지	설명
java.awt	그래픽을 처리하는 API
java.io	입출력을 스트림하는 API
java.lang	자바 프로그램의 필수 API
java.math	수학과 관련된 API
java.net	네트워크를 처리하는 API
java.text	날짜, 시간, 문자열 등 지역화를 처리하는 API
java.time	JDK 8이 지원하는 날짜 및 시간을 처리하는 API
java.util	날짜, 리스트, 벡터 등 유틸리티 API
javax.swing	스윙 컴포넌트 API

IntelliJ IDEA에서 확인 가능한 다수의 모듈 파일



패키지와 API 문서

API 문서

- **API 문서:** 개발자가 참고하여 사용할 수 있도록 패키지 기능들을 정리해 놓은 문서
- JDK에 포함된 라이브러리는 매우 방대해 개발자가 모두 기억할 수 없음
- 패키지를 올바르게 사용하기 위해 API 문서를 자주 참고해야 함

Oracle 공식 자바 API 문서: https://docs.oracle.com/en/java/javase/14/docs/api

1. 개념적 중요성: “이름 충돌 방지와 구조화”

- 패키지(Package)는 “클래스의 이름공간(namespace)”을 제공
  → 즉, 같은 이름의 클래스가 여러 개 존재하더라도, 서로 다른 패키지에 있으면 충돌하지 않음.
- 규모가 커질수록 소스 코드가 복잡해지고, 클래스가 수십~수백 개로 늘어나는데, 패키지가 없으면 관리가 불가능.
  → “패키지는 클래스 폴더 구조이자, 대규모 소프트웨어의 최소 단위”

2. 실무적 중요성: “재사용성과 모듈화”

- 나중에 **팀 프로젝트나 API 설계**를 할 때, 기능별로 패키지를 구분(service, model, controller 등)하면 유지보수가 쉬움.

“패키지는 자바에서 클래스의 주소체계입니다. 주소가 있어야 파일을 구분하고, 접근 권한을 설정할 수 있습니다. 패키지는 “대규모 소프트웨어의 설계 원리(모듈화, 캡슐화)”로 이어집니다.”

java.lang 패키지

java.lang 패키지

- 자바 프로그램에서 가장 기본이 되는 클래스와 인터페이스를 보관
- import문 없이 사용
- Java.lang 패키지에 포함된 주요 클래스

클래스	설명
Class	실행 중에 클래스 정보를 제공한다.
Math	각종 수학 함수를 제공한다.
Object	최상위 클래스로 기본적인 메서드를 제공한다.
String, StringBuffer, StringBuilder	문자열을 처리하는 메서드를 제공한다.
System	시스템 정보나 입출력을 처리하는 메서드를 제공한다.
Thread	스레드를 처리하는 메서드를 제공한다.
포장 클래스	기초 타입 데이터를 객체로 처리하는 메서드를 제공한다.

java.lang 패키지

java.lang.Object 클래스

- 모든 클래스의 조상 클래스. Java에서 정의하는 모든 클래스는 Object 클래스를 상속.

- Object 클래스가 제공하는 주요 메서드

메서드	설명
public String toString( )	객체의 문자 정보를 반환한다.
public boolean equals(Object o)	현재 객체와 동일한지 여부를 반환한다.
public int hashCode( )	객체의 해시코드를 반환한다.
protected Object clone( )	객체의 사본을 생성한다.
protected void finalize( )	가비지 컬렉터가 객체를 수거할 때 호출한다.
public final Class<?> getClass( )	객체 정보를 반환한다.

java.lang 패키지

java.lang.Object #toString(), #equals()

- 대다수 클래스는 Object 클래스가 제공하는 toString( )과 equals( ) 메서드를 오버라이딩해서 사용
- String은 이 두 메서드를 오버라이딩한 클래스

메서드	설명
public String toString( )	객체의 문자 정보를 반환한다.
public boolean equals(Object o)	현재 객체와 동일한지 여부를 반환한다.
public int hashCode( )	객체의 해시코드를 반환한다.
protected Object clone( )	객체의 사본을 생성한다.
protected void finalize( )	가비지 컬렉터가 객체를 수거할 때 호출한다.
public final Class<?> getClass( )	객체 정보를 반환한다.

java.lang 패키지

java.lang.Class 클래스

- 실행 중인 자바 프로그램 내부에 포함된 클래스와 인터페이스 정보를 제공
- Class 클래스는 생성자가 없으며, 객체를 생성하면 JVM이 대응하는 Class 객체를 자동으로 생성

```
String instance = new String("HI");
System.out.println(instance + ": " + instance.getClass());
```

출력: 'HI: class java.lang.String'

java.lang 패키지

java.lang.Math 클래스

- 지수와 로그, 삼각함수 등 기본 산술 연산을 수행하는 정적 메서드들을 제공
- 모든 메서드가 static이므로, 객체를 생성하지 않고 메서드 호출

메서드	반환 값
static double abs(double a)	실수 a의 절댓값
static double cos(double a)	실수 a의 cosine 값
static double exp(double a)	e^a 값
static double log(double a)	실수 a에 대한 자연 로그 값
static double log10(double a)	실수 a에 대한 10의 로그 값
static double max(double a, double b)	실수 a와 b 중 큰 값
static double min(double a, double b)	실수 a와 b 중 작은 값
static double pow(double a, double b)	a^b 값
static double random( )	0.0 이상 1.0 미만의 난수
static double sin(double a)	실수 a의 sine 값
static double sqrt(double a)	실수 a의 제곱근 값
static double tan(double a)	실수 a의 tangent 값

java.lang 패키지

java.lang.String 클래스

- String 객체 간 합치기, String 객체의 일부 문자열만 불러오기 등 연산은 매번 새로운 String 객체를 생성
- 문자열 조작이 자주 발생할 경우 String 클래스만을 사용하는 것은 비효율적

```
String a = "HI";
String b = a + " My Name is "; // String 객체 생성
String c = "Cat"; // String 객체 생성
```

java.lang 패키지

java.lang.StringBuilder 클래스

- 문자열 관련 연산을 효율적으로 수행할 수 있도록 제공되는 클래스
- 다중 스레드 환경에서는 StringBuilder 대신 StringBuffer 클래스를 사용하는 것이 안전

메서드	설명
StringBuilder append(String s)	문자열 s를 버퍼에 덧붙인다.
int capacity( )	현재 버퍼의 크기를 반환한다.
StringBuilder delete(int start, int end)	문자열의 일부분을 버퍼에서 제거한다.
StringBuilder insert(int offset, String s)	문자열 s를 버퍼의 offset 위치에 삽입한다.
StringBuilder replace(int start, int end, String s)	문자열의 일부분을 문자열 s로 대체한다.
StringBuilder reverse( )	버퍼에 있는 문자열을 반대 순서로 변경한다.

java.lang 패키지

java.lang.System 클래스

- 표준 입출력을 비롯한 실행 시스템과 관련된 필드와 메서드를 static으로 제공
- System.out.println() 또한 System 클래스가 제공하는 메서드

- System클래스의 세 가지 필드

필드	설명
static InputStream in	표준 입력 스트림이다.
static PrintStream out	표준 출력 스트림이다.
static PrintStream err	표준 오류 출력 스트림이다.

java.lang 패키지

java.lang.System 클래스

- System클래스가 제공하는 주요 메서드

메서드	설명
static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)	주어진 위치에서 주어진 길이만큼 배열 src를 배열 dest로 복사한다.
static long currentTimeMillis( )	현재 시각을 밀리초 단위로 반환한다.
static void exit( )	현재 실행 중인 JVM을 종료한다.
static void gc( )	가비지 컬렉터를 실행한다.
static String getenv(String name)	지정된 환경 변수 값을 반환한다.
static String getProperty(String key)	주어진 key 값에 해당하는 시스템 특성을 반환한다.
static long nanoTime( )	현재 시각을 나노초 단위로 반환한다.

java.lang 패키지

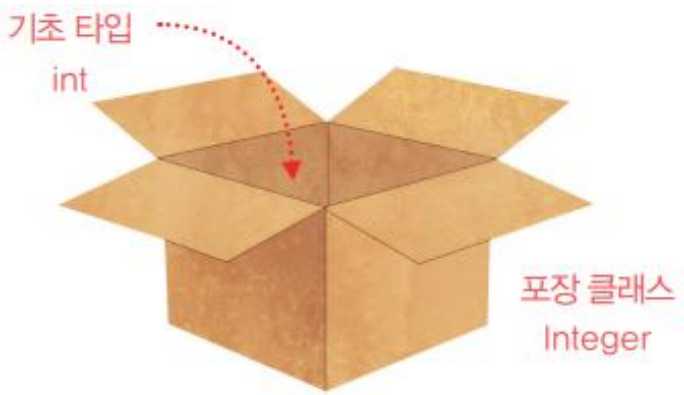
java.lang.System 클래스

- 자바에서는 운영체제로부터 할당 받은 메모리를 JVM이 관리
- JVM은 메모리가 부족하거나 주기적으로 가비지 컬렉터를 사용해 가비지를 수거
- 가비지를 수거하는 순서는 객체의 생성 순서와는 무관
- 프로그램에서 가비지 컬렉터를 직접 호출 불가.
  대신에 System.gc()로 JVM에 가능하면 빨리 가비지 컬렉터를 실행하도록 요청 가능

java.lang 패키지

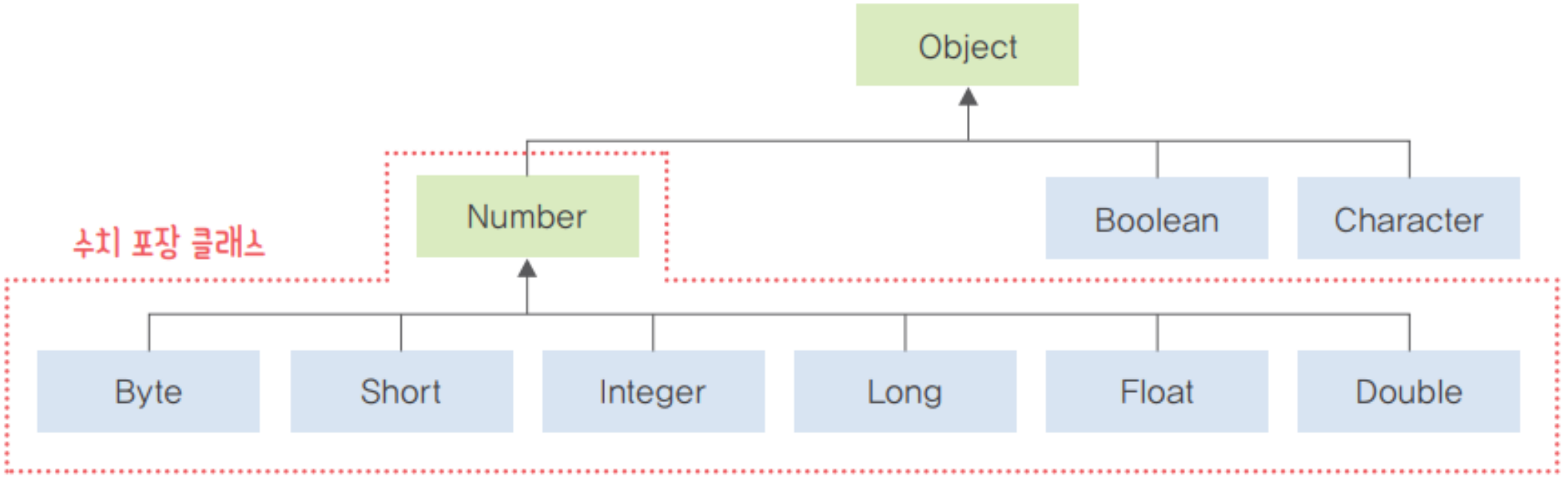
포장 클래스 (Wrapper Class)

- 대부분의 기본 패키지가 제공하는 클래스의 메서드는 **참조 타입**을 매개변수로 사용하기 때문에 기초 타입을 사용하면 객체 지향 언어의 특징을 이용 불가
- 자바는 **기초 타**입을 포장해 클래스화 한 포장 클래스(wrapper class)를 제공해 기초 타입 데이터도 기본 패키지와 상호작용 가능하도록 함



java.lang 패키지

Wrapper 클래스의 계층 구조



java.lang 패키지

Wrapper 클래스의 생성자

포장 클래스	생성자
Byte	Byte(byte value), Byte(String s)
Short	Short(short value), Short(String s)
Integer	Integer(int value), Integer(String s)
Long	Long(long value), Long(String s)
Float	Float(double value), Float(float value), Float(String s)
Double	Double(double value), Double(String s)
Character	Character(char value)
Boolean	Boolean(boolean value), Boolean(String s)

java.lang 패키지

Integer 클래스의 주요 메서드

- Integer 클래스는 기초 타입인 int에 대한 포장 클래스

메서드	설명
int intValue( )	int 타입으로 반환한다.
double doubleValue( )	double 타입으로 반환한다.
float floatValue( )	float 타입으로 반환한다.
static int parseInt(String s)	문자열을 int 타입으로 반환한다.
static String toBinaryString(int i)	int 타입을 2진수 문자열로 반환한다.
static String toHexString(int i)	int 타입을 16진수 문자열로 반환한다.
String toString(int i)	int 타입을 10진수 문자열로 반환한다.
static Integer valueOf(String s)	문자열을 Integer 객체로 반환한다.
static Integer valueOf(String s, int radix)	radix 진수의 문자열을 Integer 객체로 반환한다.

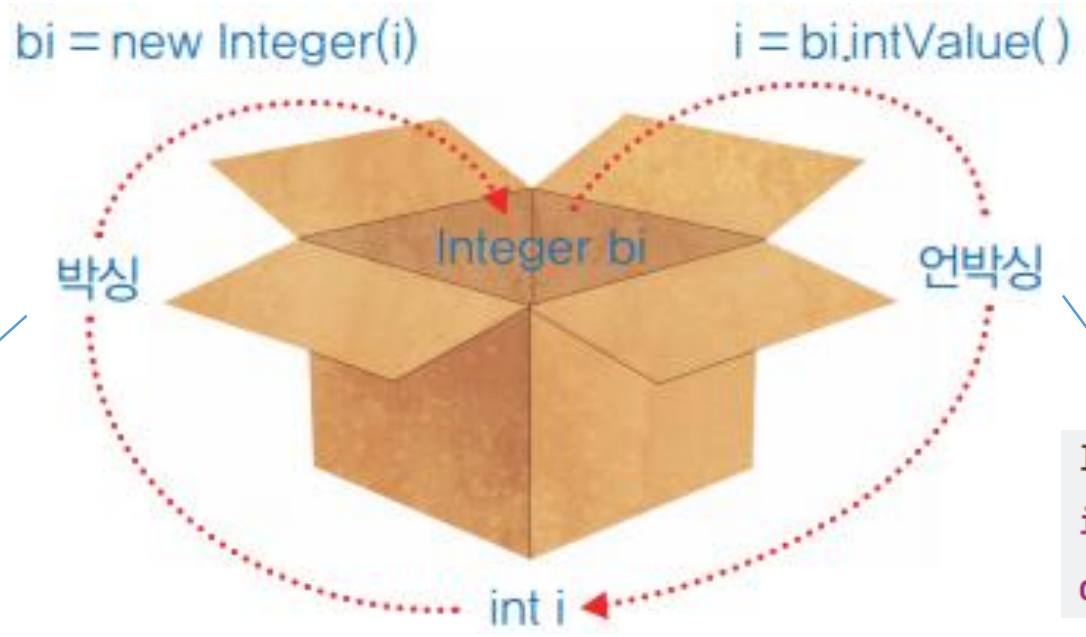
java.lang 패키지

박싱과 언박싱

- Integer 클래스는 기초 타입인 int에 대한 포장 클래스

```
Integer bi = new Integer(10);
Integer bi = Integer.valueOf(10);
```

```
Integer bi = 10;        // 자동 박싱
int i1 = bi;            // 자동 언박싱
int i2 = bi + 20;       // 자동 언박싱
```




```
Integer bi = new Integer(10);
int i = bi.intValue();
double d = bi.doubleValue();
```

Practice

java.lang 패키지 활용

예제:

- 8-1 (p. 314)
- 8-2 (p. 315)
- 8-3 (p. 316)
- 8-4 (p. 317)
- 8-8 (p. 325)

java.util 패키지

java.util 패키지

- 날짜, 시간, 리스트, 벡터, 해시 테이블, 컬렉션 등 다양한 유틸리티 클래스와 인터페이스를 제공

- java.util 패키지가 제공하는 주요 클래스

클래스	설명
Arrays	배열을 비교, 복사, 정렬 등 조작할 때 사용한다.
Calendar	날짜와 시간 정보가 필요할 때 사용한다.
Date	밀리초 단위의 현재 시각이 필요할 때 사용한다.
StringTokenizer	특정 문자로 구분된 문자열을 뽑아낼 때 사용한다.
Random	난수가 필요할 때 사용한다.

java.util 패키지

java.util.Arrays 클래스

- Arrays 클래스가 제공하는 주요 정적 메서드

메서드	설명
List asList(배열)	배열을 리스트로 변환한다.
int binarySearch(배열, 키)	배열에서 키 값이 있는 인덱스를 반환한다.
배열 copyOf(배열, 길이)	원본 배열을 길이만큼 복사한다.
배열 copyOfRange(배열, 시작, 끝)	원본 배열을 지정한 영역만큼 복사한다.
boolean equals(배열, 배열)	두 배열의 동일 여부를 비교한다.
void fill(배열, 값)	배열을 지정된 값으로 저장한다.
void fill(배열, 시작, 끝, 값)	배열의 지정된 영역에 지정된 값을 저장한다.
void sort(배열)	배열을 오름차순으로 정렬한다.

java.util 패키지

java.util.Date 클래스

- 날짜와 시각 정보 표현.
- **국제화에 맞지 않아 대부분의 메서드는 현재 폐기 중**
- 주로 하위 호환성이나 간단한 날짜 정보를 원할 때만 사용

java.util 패키지

java.util.Calendar 클래스

- 지역이나 문화에 따라 달력을 표시하는 방식이 다르기 때문에 추상 클래스로 되어 있음
- 표준 달력을 사용한다면 다음과 같이 객체 생성

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

java.util 패키지

java.util.Calendar 클래스

- Calendar 클래스가 제공하는 정수 타입의 상수

필드 이름	의미
AM, AM_PM, PM	오전 및 오후
DATE	날짜
JANUARY, FEBRUARY, ...	1월, 2월 등
SUNDAY, MONDAY, ...	일요일, 월요일 등
MINUTE	분
HOUR	시간(0~11)
HOUR_OF_DAY	시간(0~23)
MONTH	월(0~11)
DAY_OF_MONTH	한 달 내에서의 날짜
WEEK_OF_YEAR	일 년 내에서의 몇 주차
YEAR	연도

java.util 패키지

java.util.Calendar 클래스

- Calendar 클래스가 제공하는 주요 메서드

메서드	설명
boolean after(Object when)	주어진 시간보다 뒤쪽이면 true를 반환한다.
boolean before(Object when)	주어진 시간보다 앞쪽이면 true를 반환한다.
void clear(int field)	지정된 필드를 미정의 상태로 변경한다.
int compareTo(Calendar anotherCalendar)	2개의 Calendar 객체를 비교한다.
int get(int fields)	주어진 필드 값을 반환한다.
int getFirstDayOfWeek( )	첫 날이 무슨 요일인지 반환한다.
Date getTime( )	Calendar 객체를 Date 객체로 변환한다.
void set(int field, int value)	주어진 필드를 주어진 값으로 변경한다.
void set(int year, int month, int date)	연, 월, 일 값을 변경한다.
void setTime(Date date)	Date 객체로 Calendar 객체를 설정한다.

java.util 패키지

java.util.StringTokenizer 클래스

- 문자열을 토큰으로 분리하는 데 사용
- 토큰은 공백이나 줄 바꿈 등 구분자를 사용해 문자열을 분리

- StringTokenizer 클래스의 주요 생성자

생성자	설명
StringTokenizer(String s)	주어진 문자열을 기본 구분자로 파싱한 StringTokenizer 객체를 생성한다.
StringTokenizer(String s, String delim)	주어진 문자열을 delim 구분자로 파싱한 StringTokenizer 객체를 생성한다.

기본 구분자: 공백, 탭, 줄 바꿈, 복귀, 용지 먹임 문자

java.util 패키지

java.util.Random 클래스

- 난수 생성 기능 제공

생성자	설명
Random( )	Random 객체를 생성한다.
Random(long seed)	주어진 시드를 사용하는 Random 객체를 생성한다.

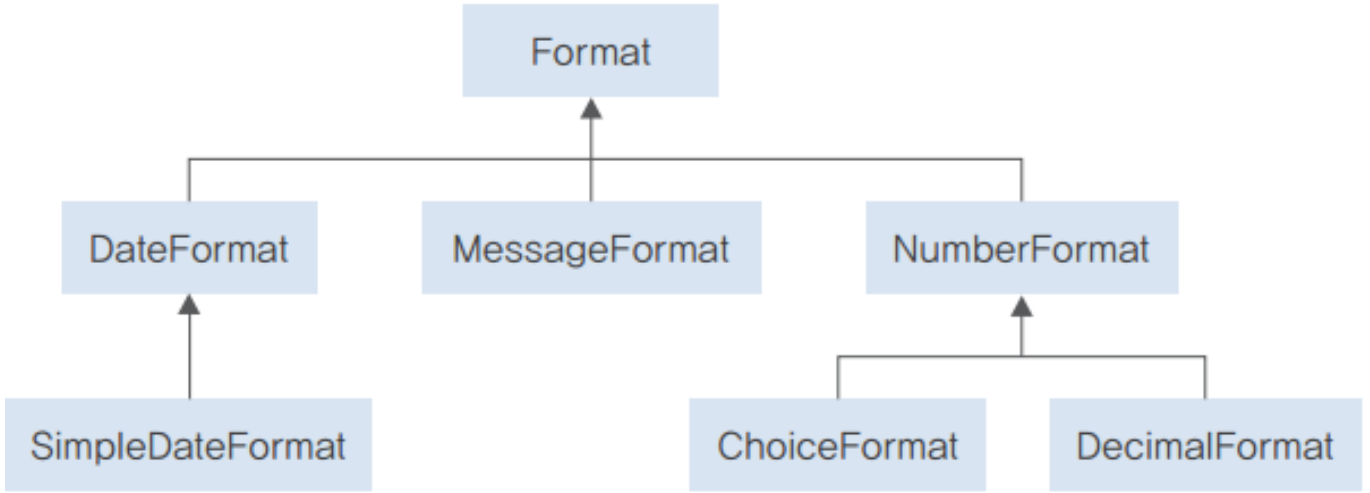
메서드	설명
boolean nextBoolean( )	논리 타입 난수를 발생시킨다.
double nextDouble( )	0.0 이상 1.0 미만의 double 타입 난수를 발생시킨다.
float nextFloat( )	0.0 이상 1.0 미만의 float 타입 난수를 발생시킨다.
double nextGaussian( )	평균, 표준편차가 0.0 및 1.0인 정규분포 난수를 발생시킨다.
int nextInt( )	int 타입의 난수를 발생시킨다.
int nextInt(int n)	0~(n−1) 사이의 int 타입 난수를 발생시킨다.
long nextLong( )	long 타입의 난수를 발생시킨다.
void setSeed(long seed)	시드 값을 설정한다.

java.text 패키지

java.text 패키지

- 현지화가 필요한 데이터의 효율적 처리를 위한 패키지
- 특히 패키지의 **Format** 클래스는 지역에 민감한 데이터를 현장에 맞게 문자열로 표현하고 포맷할 수 있도록 지원

- **Format** 클래스의 계층 구조



java.text 패키지

java.text.SimpleDateFormat 클래스

- 날짜 정보를 현지화하는 클래스로 날짜를 텍스트로 포맷하거나 텍스트를 날짜로 파싱
- SimpleDateFormat 클래스에서 사용할 수 있는 패턴 기호

패턴 기호	설명	패턴 기호	설명
y	연	h	시(1~12)
M	월	H	시(0~23)
w	월 구분 없는 주	k	시(1~24)
W	주	K	시(0~11)
d	일	m	분
D	월 구분 없는 일	s	초
E	요일	S	밀리초
a	오전과 오후	z	타임존

java.text 패키지

java.text.SimpleDateFormat 클래스

- 사용 방법

Date ➔ String

```
SimpleDateFormat f = new SimpleDateFormat("패턴");
String s = f.format(new Date());
```

String ➔ Date

```
SimpleDateFormat f = new SimpleDateFormat("패턴");
Date d = f.parse("날짜 문자열");
```

java.text 패키지

java.text.MessageFormat 클래스

- 문자열을 특정 포맷에 맞추어 깔끔하게 처리하는 클래스

- MessageFormat 객체 생성

```
MessageFormat(String pattern)
MessageFormat(String pattern, Locale locale)
static String format(String pattern, Object... arguments)
```

java.text 패키지

java.text.MessageFormat 클래스

```
String template = "안녕하세요, {0}님! 오늘은 {1}입니다.";
String message = MessageFormat.format(template, "세진", "화요일");
System.out.println(message);
```

출력: 안녕하세요, 세진님! 오늘은 화요일입니다.

java.text 패키지

java.text.DecimalFormat 클래스

- 정수, 실수를 포함한 다양한 종류의 수를 과학적 표기, 퍼센트 표시, 화폐 표시 등으로 포맷 지원
- DecimalFormat 클래스에서 사용할 수 있는 패턴 기호

패턴 기호	설명	사용 예	1234567.890의 반환 값
#	10진수	#	12345678
0	선행 제로 10진수	000000000.00	01234567.89
.	소수점	#.000	1234567.890
,	구분자	#,###.##	1,234,567.89
+ 또는 -	양수 또는 음수	-#.0	-1234567.9
E	지수	#.##E00	1.23E06
;	패턴 구분	+#.#;-#.#	+1234567.9
%	백분율	#.00%	123456789.00%

```
DecimalFormat f = new DecimalFormat("패턴");
String s = f.format(숫자);
```