

파워자바(개정3판)

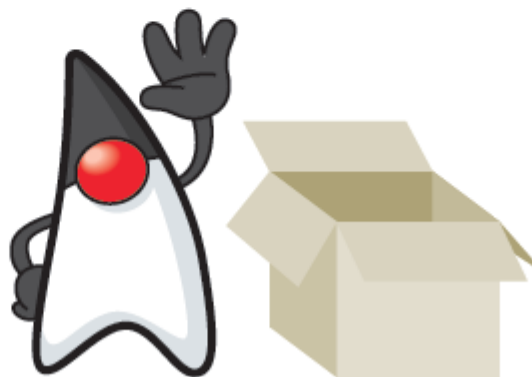


8장 자바API패키지, 예외
처리, 모듈



8장의 목표

1. 관련된 클래스들을 패키지로 묶을 수 있나요?
2. 외부 패키지들을 불러와서 패키지 안에 포함된 클래스들을 사용할 수 있나요?
3. 자바의 기본 패키지와 유틸리티 패키지 안의 클래스의 사용법을 파악하고 있나요?
4. String 클래스를 이용하여 문자열을 처리할 수 있나요?
5. 자바 프로그램에서 나타나는 오류들을 우아하게 처리할 수 있나요?





패키지

- 자바도 클래스들을 모아 놓는 상자를 가지고 있다. 이것이 바로 패키지이다.



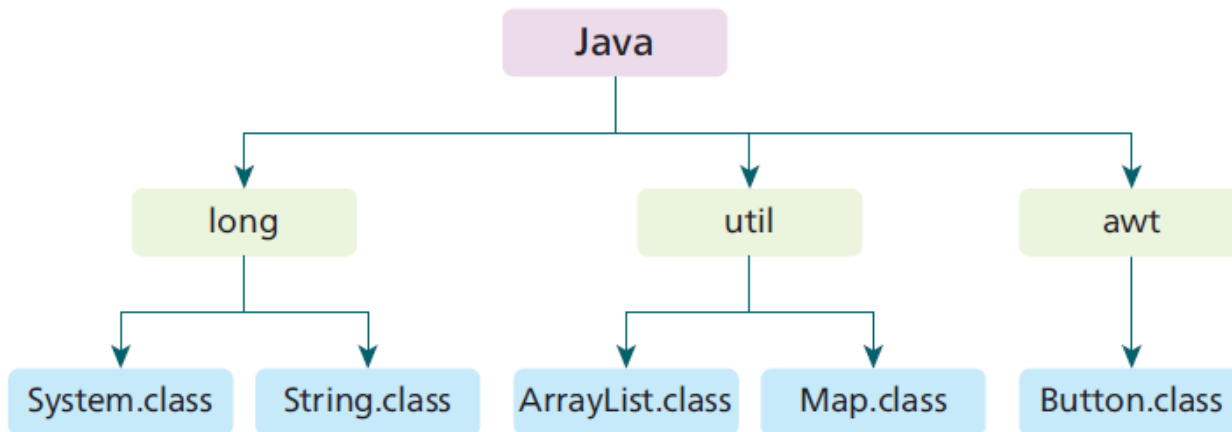
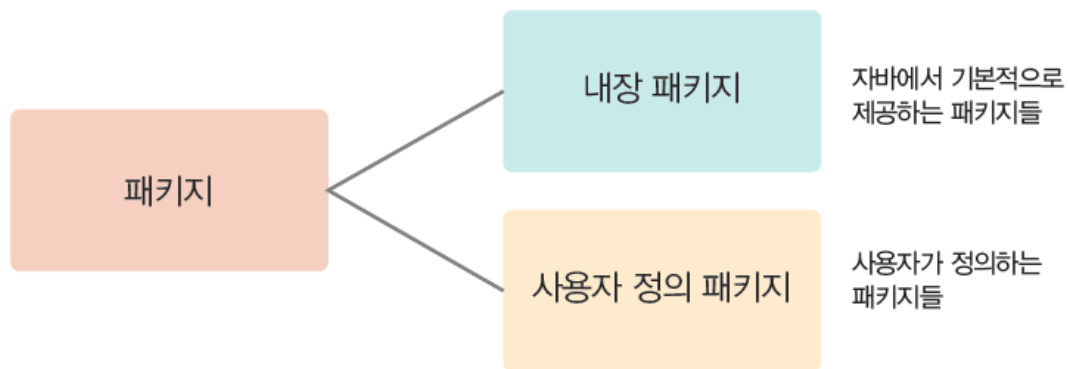


- [illegible]





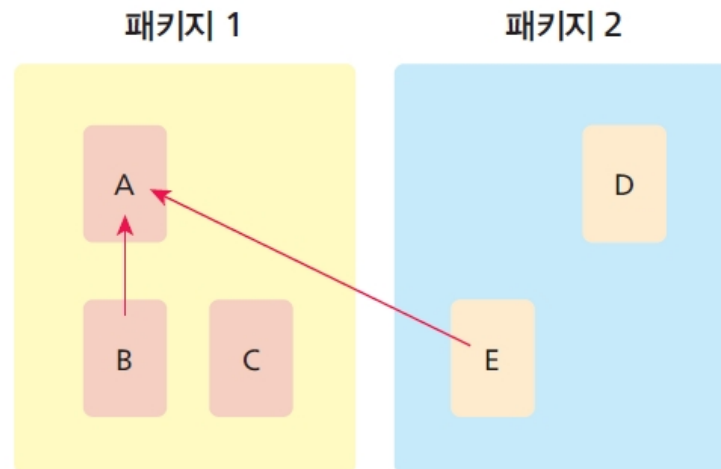
패키지의 종류





왜 패키지가 필요할까?

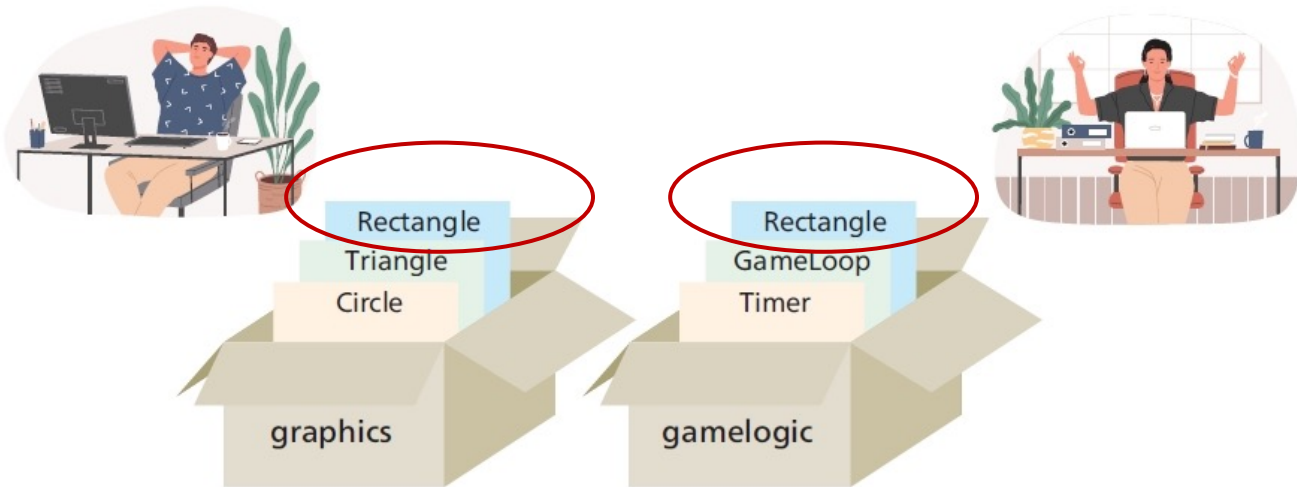
- 패키지를 이용하면 서로 관련된 클래스들을 하나의 단위로 모을 수 있다.
- 패키지를 사용하는 중요한 이유 중 하나는 “이름공간(**name space**)” 때문이다.
- 패키지를 이용하여 세밀한 접근 제어를 구현할 수 있다. 패키지 안의 클래스들은 패키지 안에서만 사용이 가능하도록 선언될 수 있다.





이름 공간

- 이러한 이름 충돌을 방지할 방법이 필요하다. 패키지는 이런 경우에 사용할 수 있는 중요한 도구이다. 패키지만 다르면 개발자들은 동일한 클래스 이름을 마음 놓고 사용할 수 있다.





중간점검



중간점검

1. 패키지는 관련있는 _____나 _____를 하나로 묶은 것이다.
2. 패키지의 장점은 무엇인가?



패키지 선언하기

Circle.java

```
01 package graphics;  
02  
03 public class Circle {  
04     double radius;  
05 }
```

소스 파일을 패키지에 넣으려면 소스
파일의 맨 처음에 package 패키지이름;
문장을 넣으면 됩니다.





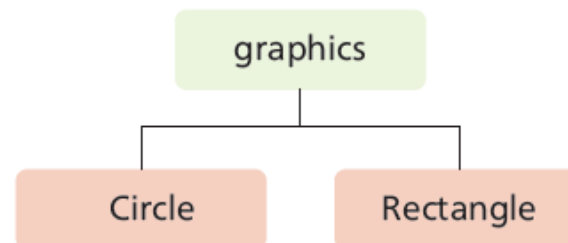
패키지 선언하기

```
package graphics;
```

```
public class Circle {  
    double radius;  
}
```

```
package graphics;
```

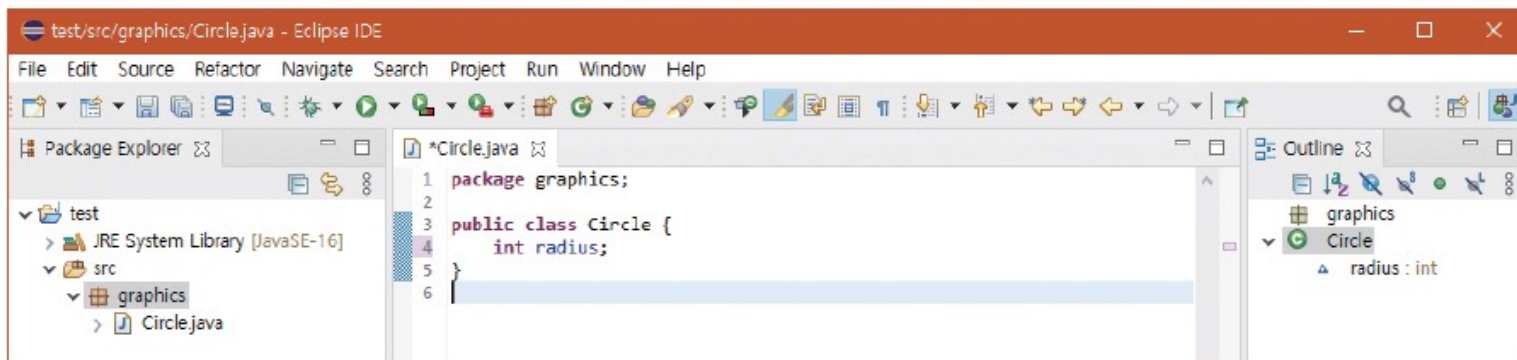
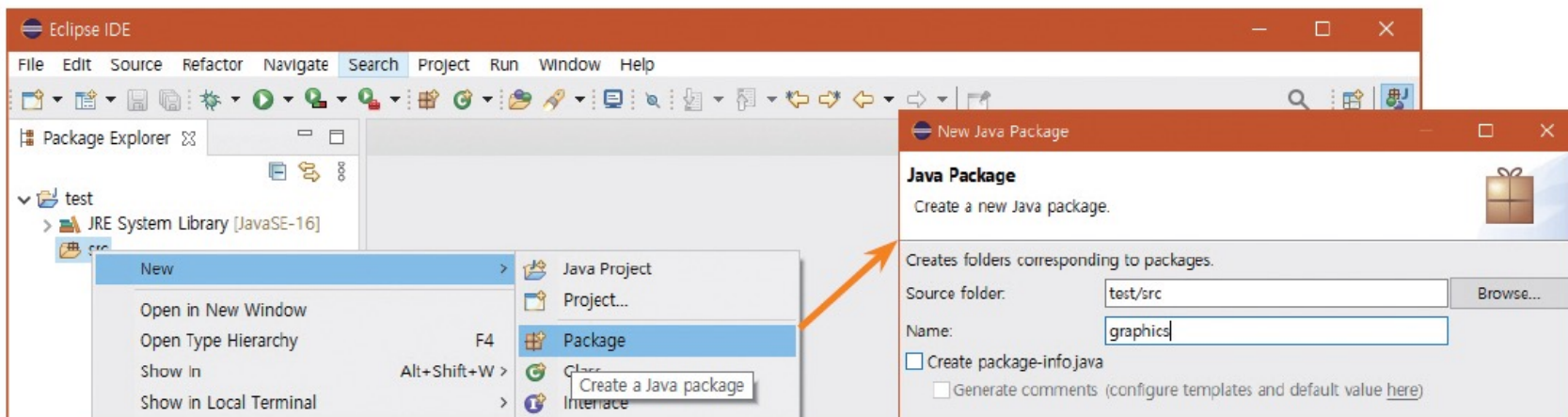
```
public class Rectangle {  
    double width, height;  
}
```



예를 들어서 Circle.java와 Rectangle.java
파일에 package graphics; 문장
을 넣으면 Circle 클래스와 Rectangle
클래스가 모두 graphics 패키지에 속하게
된다.



이클립스에서 패키지 만들기





소스 파일과 클래스 파일의 위치

```
D:\test\  
---> src\  
    ---> graphics\  
        ---> Circle.java
```

```
D:\test\  
---> bin\  
    ---> graphics\  
        ---> Circle.class
```

패키지 구조를 반영하여 디렉토리를 만든다.



중간점검



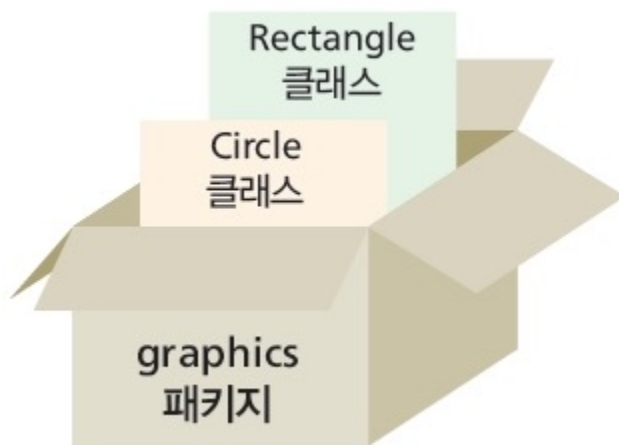
중간점검

1. Circle.java 파일을 util 패키지에 속하도록 하려면 어떤 문장을 맨 처음에 두어야 하는가?
2. 패키지 이름을 짓는 일반적인 관례는 무엇인가?
3. 인터넷 도메인 이름이 hu.ac.kr이라면 권장되는 패키지 이름은?
4. 만약 패키지의 이름이 company.project와 같다면 소스 파일이 저장되는 디렉터리는?
5. 만약 패키지의 이름이 company.project와 같다면 클래스 파일이 저장되는 디렉터리는?



패키지 사용하기

1. 완전한 이름으로 참조한다. 완전한 이름이란 패키지 이름이 클래스 앞에 붙은 것이다.
2. 패키지 안에서 우리가 원하는 클래스만을 포함한다.
3. 패키지 안의 모든 클래스를 포함한다.



`graphics.Rectangle ???`





완전한 이름으로 참조하는 방법

- 외부 패키지에 있는 클래스를 사용하려면 원칙적으로 앞에 패키지를 붙인 정식 클래스 이름을 써주어야 한다.

```
graphics.Rectangle myRect = new graphics.Rectangle();
```



원하는 패키지 멤버만을 포함하는 방법

- 외부 패키지의 특정한 멤버를 **import**하려면 다음과 같은 문장을 사용한다.

```
import graphics.Rectangle;
```

```
Rectangle myRect = new Rectangle();
```




전체 패키지를 포함하는 방법

- 하나의 패키지 안에 포함된 모든 클래스를 포함하려면 다음과 같이 별표 문자(*)를 사용하면 된다.

```
import graphics.*;           // 패키지의 모든 클래스 포함
```

```
Rectangle myRect = new Rectangle();
```



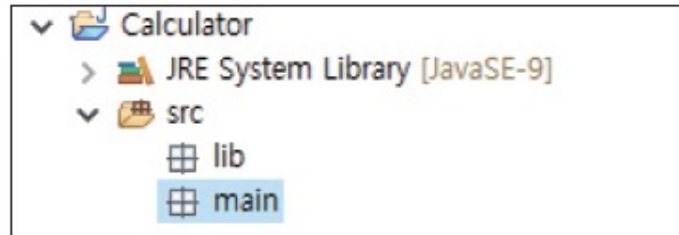
계층 구조의 패키지 포함하기

- 예를 들어서 `java.awt.*`를 포함시키면 `java.awt` 아래에 있는 패키지, 즉 `java.awt.font`와 같은 패키지가 자동으로 포함될 거라고 생각하기 쉽다. 그러나 `java.awt.font` 패키지는 자동으로 포함되지 않는다. 다음과 같이 별도로 포함하여야 한다.

```
import java.awt.*;           // java.awt 패키지의 클래스 포함
import java.awt.font.*;      // java.awt.font 패키지의 클래스 포함
```

예제: 이클립스로 패키지 만들고 사용하기

1. 이클립스로 패키지를 만드는 방법을 학습해보자. 먼저 Calculator라는 프로젝트를 만든다. 이어서 src 폴더 위에서 마우스 오른쪽 버튼을 눌러서 [New] → [Package]를 선택하고 lib와 main 패키지를 각각 생성한다.



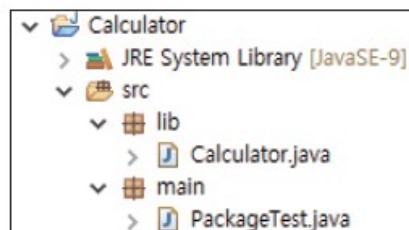
예제: 이클립스로 패키지 만들고 사용하기

- lib 패키지 위에서 마우스 오른쪽 버튼을 눌러서 [New] → [Class]를 선택하고 Calculator 클래스를 추가한다. 그리고 다음과 같이 입력한다. 첫 번째 문장은 자동으로 생성된다.

Calculator.java

```
01 package lib;
02
03 public class Calculator {
04     public int add(int a, int b) {
05         return a + b;
06     }
07 }
```

- main 패키지 위에서 마우스 오른쪽 버튼을 눌러서 [New] → [Class]를 선택하고 PackageTest 클래스를 추가한다.





예제: 이클립스로 패키지 만들고 사용하기

PackageTest.java

```
01 package main;
02 import lib.*;
03
04 public class PackageTest {
05     public static void main(String args[]) {
06         Calculator obj = new Calculator();
07         System.out.println(obj.add(100, 200));
08     }
09 }
```

300



중간점검



중간점검

1. java.util 클래스의 ArrayList 클래스를 import 문장 없이 사용하려고 한다. 어떻게 하면 되는가?
2. java.util 클래스의 ArrayList 클래스만 포함시키는 문장을 작성해보자.
3. java.util 패키지의 모든 클래스를 포함시키려면 어떤 문장을 사용해야 하는가?



클래스 파일은 언제 로드될까?

- 자바 소스 파일(확장자 **.java**)이 컴파일되면 **.class** 확장자를 가지는 클래스 파일로 변환되고 파일 시스템에 저장된다. 클래스 파일은 자바 가상 기계(JVM)에 의하여 로드된다.
 - 클래스 파일은 요청 시 JVM에 로드된다. 이것이 기본적인 방법으로 지연 클래스 로드(Lazy Class Loading)이라고도 한다.
 - 애플리케이션 코드를 구성하는 기본적인 클래스 파일은 시작 시 로드된다.



간단한 예

- 기본적인 방법은 지연 클래스 로드이다. 이 방법의 가장 큰 장점은 가상 머신의 메모리를 절약할 수 있다는 것이다. 예를 들어서 사용자 프로그램에 다음과 같은 코드 섹션이 있다고 가정하자.

```
if(result != null) {  
    MyClass object = new MyClass();  
    // 어떤 작업을 한다.  
}
```

- 위의 코드에서 **result**가 **null**이면 내부의 코드 블록은 절대 실행되지 않는다. 이러한 상황에서 가상 기계는 **MyClass**를 메모리에 로드하지 않는다.



자바 가상 기계가 클래스를 찾는 순서

1. 부트스트랩 클래스 - 자바 플랫폼을 구성하는 핵심적인 클래스이다. 디렉터리 `jre/lib`에 있는 `rt.jar` 및 기타 여러 `jar` 파일이다. Java 9부터는 모듈화되어 있는 `jmods` 파일들이 로드된다.
2. 확장 클래스 - 자바 확장 메커니즘을 사용하는 클래스이다. 이들은 확장 디렉터리에 있는 `jar` 파일들이다. 디렉터리 `jre/lib/ext`의 모든 `jar` 파일은 확장으로 간주되며 로드된다.
3. 사용자 클래스 - 확장 메커니즘을 활용하지 않는 개발자 및 타사에서 정의한 클래스이다. 가상 머신 명령줄에서 `-classpath` 옵션을 사용하거나 `CLASSPATH` 환경 변수를 사용하여 이러한 클래스의 위치를 식별한다.



가상 기계가 클래스를 찾는 3가지의 방법

1. 첫 번째로 가상 기계는 현재 디렉터리부터 찾는다.
2. 일반적으로는 환경 변수인 **CLASSPATH**에 설정된 디렉터리에서 찾는다. **CLASSPATH** 변수를 설정하려면 명령 프롬프트에서 다음과 같은 명령어를 사용한다.

```
C:\> set CLASSPATH=C:\classes;C:\lib;.
```

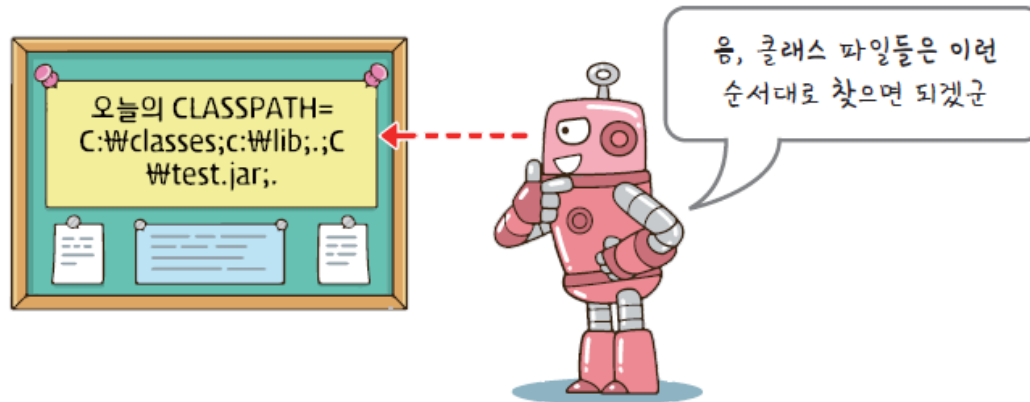
3. 가상 머신을 실행할 때 옵션 **-classpath**를 사용할 수 있다. 즉 가상 머신을 실행할 때, 클래스경로를 알려주는 것이다. 이클립스는 내부적으로 이 방법을 사용한다. 이 방법이 가장 권장되는 방법이라고 한다.

```
C:\> java -classpath C:\classes;C:\lib;. Main
```



CLASSPATH

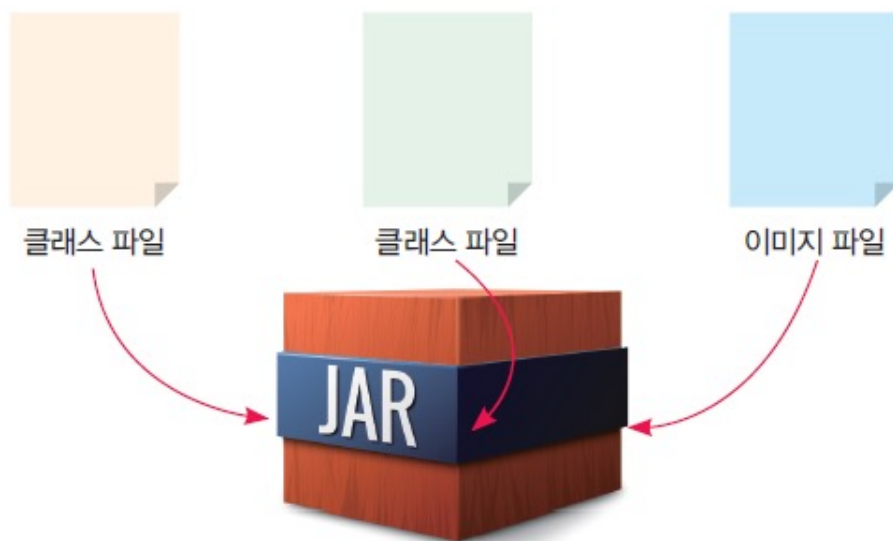
- 사용자가 직접 작성하였거나 외부에서 다운로드 받은 클래스를 찾기 위하여 가상 머신이 살펴보는 디렉터리들을 모아둔 경로가 클래스 경로(class path)이다.





JAR 압축 파일

- 자바 애플리케이션을 한 개의 파일로 만들어서 사용자에게 전달할 방법-
>JAR 파일이다. JAR는 “Java Archive”의 줄임말로서 자바 파일들을 압축하여 하나의 파일로 만드는 데 사용된다.





JAR 파일 만드는 방법

- JDK 안에 포함된 **jar** 도구를 이용하여서 **JAR** 파일을 생성할 수 있다.

```
C:\> jar cvf Game.jar *.class icon.png
```

- 만약 실행 가능한 **JAR** 파일을 생성하려면 다음과 같이 **e**를 추가하여야 한다.

```
C:\> jar cvfe Game.jar Main *.class icon.png
```

- **JAR** 파일로 압축된 파일을 실행하려면 어떻게 하면 될까?

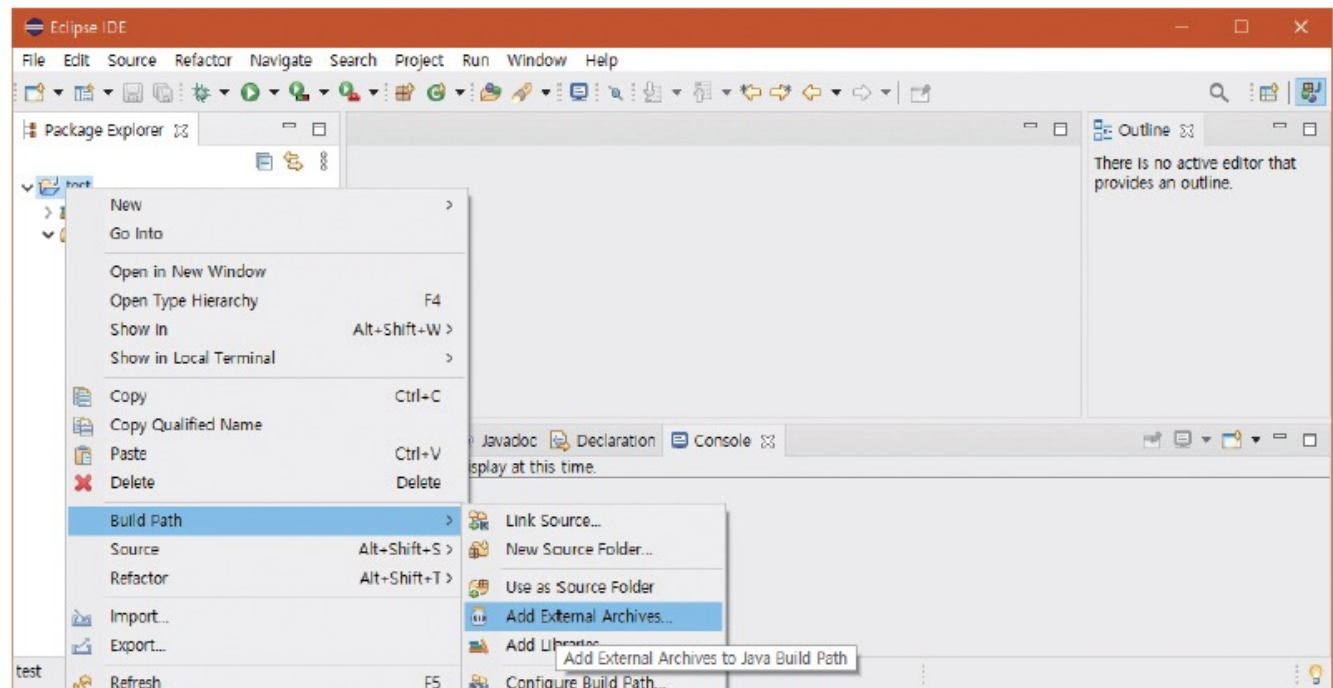
```
C:\> java -jar Game.jar
```

JAR 형태의 라이브러리 사용하기

- 외부로부터 받은 JAR 파일 `graphics.jar`를 사용하려면 다음과 같이 클래스 경로에 `graphics.jar` 파일을 포함시키면 된다.

```
C:\> set CLASSPATH=C:\classes;C:\lib;C\graphics.jar;.
```

- 이클립스





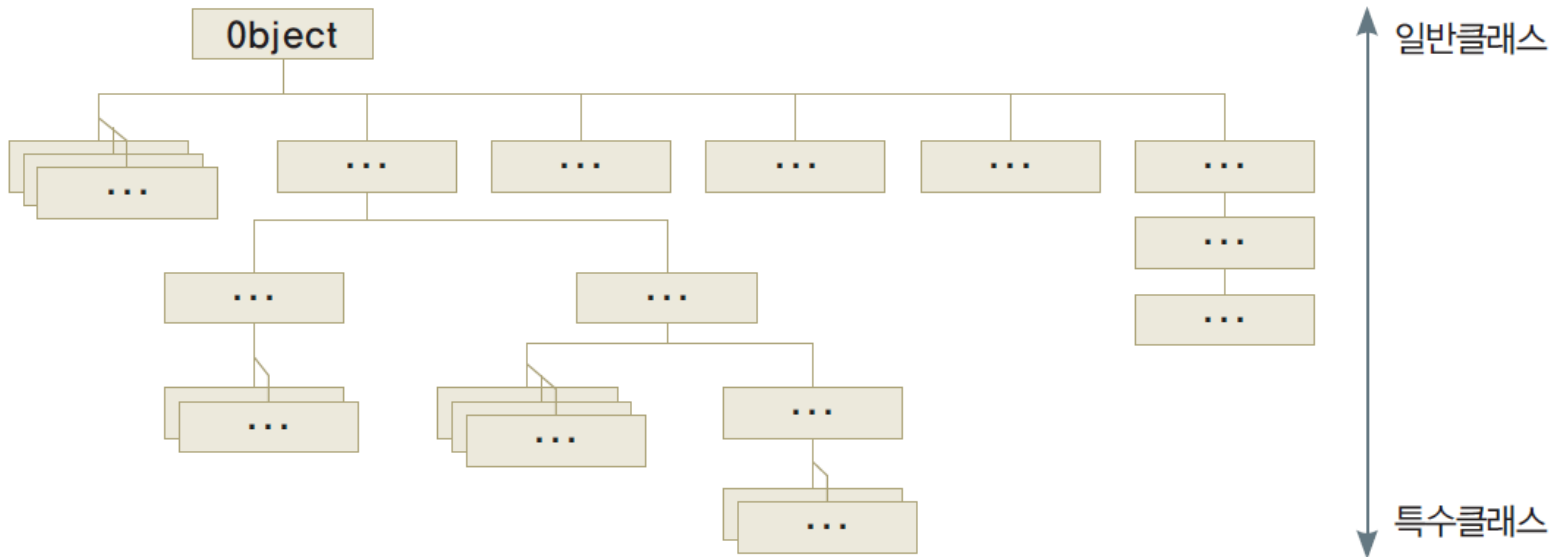
자바 API 패키지

패키지	설명
java.applet	애플릿을 생성하는 데 필요한 클래스
java.awt	그래픽과 이미지를 위한 클래스
java.io	입력과 출력 스트림을 위한 클래스
java.lang	자바 프로그래밍 언어에 필수적인 클래스
java.math	수학에 관련된 클래스
java.net	네트워킹 클래스
java.nio	새로운 네트워킹 클래스
java.security	보안 프레임워크를 위한 클래스와 인터페이스
java.sql	데이터베이스에 저장된 데이터를 접근하기 위한 클래스
java.util	날짜, 난수 생성기 등의 유틸리티 클래스
javax.imageio	자바 이미지 I/O API
javax.net	네트워킹 애플리케이션을 위한 클래스
javax.swing	스윙 컴포넌트를 위한 클래스
javax.xml	XML을 지원하는 패키지



Object 클래스

- Object 클래스는 `java.lang` 패키지에 들어 있으며 자바 클래스 계층 구조에서 맨 위에 위치하는 클래스





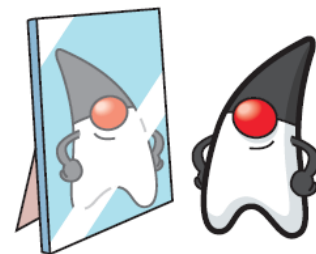
Object의 메소드

- `public boolean equals(Object obj)` : `obj`가 이 객체와 같은지를 검사한다.
- `public String toString()` : 객체의 문자열 표현을 반환한다.
- `protected Object clone()` : 객체 자신의 복사본을 생성하여 반환한다.
- `public int hashCode()` : 객체에 대한 해쉬 코드를 반환한다.
- `protected void finalize()` : 가비지 콜렉터에 의하여 호출된다.
- `public final Class getClass()` : 객체의 클래스 정보를 반환한다.



getClass()

- `getClass()`는 객체가 어떤 클래스로 생성되었는지에 대한 정보를 반환한다.



CircleTest.java

```
01  class Circle {    }
02  public class CircleTest {
03      public static void main(String[] args) {
04          Circle obj = new Circle();
05          System.out.println("obj is of type " + obj.getClass().getName());
06          System.out.println("obj의 해쉬코드=" + obj.hashCode());
07      }
08  }
```

```
obj is of type test.Circle
obj의 해쉬코드=1554547125
```



toString()

- toString() 메소드는 객체의 문자열 표현을 반환

CircleTest.java

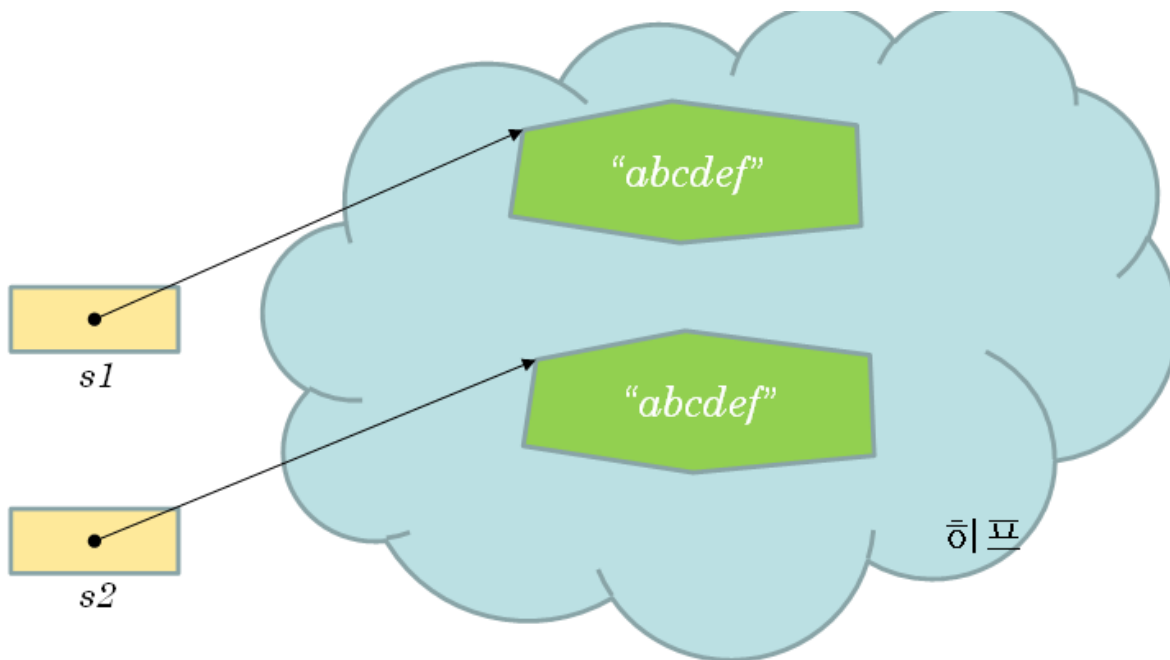
```
01  class Circle {  
02      int radius;  
03      public Circle(int radius) {          this.radius = radius;          }  
04      public String toString() {          return "Circle(r="+radius+")";          }  
05  }  
06  public class CircleTest {  
07      public static void main(String[] args) {  
08          Circle obj = new Circle(10);  
09          System.out.println(obj);  
10      }  
11  }
```

Circle(r=10)



equals() 메소드

- Object에서 제공되는 equals()는 == 연산자를 사용하여 객체의 주소가 동일한지를 검사하여서 true 또는 false를 반환한다. 하지만 객체에 대해서는 이것이 옳바르지 않는 경우가 많이 있다.





equals() 메소드

CircleTest2.java

```
01  class Circle {  
02      int radius;  
03      public Circle(int radius) {          this.radius = radius;      }  
04  }  
05  public class CircleTest {  
06      public static void main(String[] args) {  
07          Circle obj1 = new Circle(10);  
08          Circle obj2 = new Circle(10);  
09          if( obj1 == obj2 ) System.out.println("2개의 원은 같습니다.");  
10          else System.out.println("2개의 원은 같지 않습니다.");  
11      }  
12  }
```

2개의 원은 같지 않습니다.



equals() 메소드

CircleTest.java

```
01  class Circle {
02      int radius;
03      public Circle(int radius) {          this.radius = radius;      }
04      public boolean equals(Circle c1) {
05          if( radius == c1.radius ) return true;
06          else return false;
07      }
08  }
09  public class CircleTest {
10      public static void main(String[] args) {
11          Circle obj1 = new Circle(10);
12          Circle obj2 = new Circle(10);
13          if( obj1.equals(obj2) ) System.out.println("2개의 원은 같습니다.");
14          else System.out.println("2개의 원은 같지 않습니다.");
15      }
16  }
```

2개의 원은 같습니다.



중간점검

1. 객체가 자신을 만든 클래스에 대하여 알고 싶다. 어떤 메소드를 사용해야 하는가?
2. 어떤 객체와 다른 객체가 동일한 내용물을 가지고 있는지를 검사하고 싶다. 어떤 메소드를 재정의해야 하는가?
3. 객체를 출력했을 때 객체의 요약 문자열이 출력되게 하려면 어떤 메소드를 재정의하여야 하는가?



중간점검

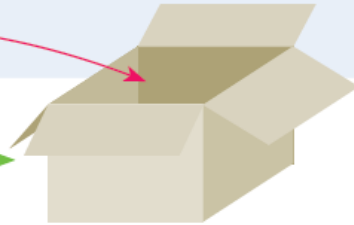


래퍼 클래스

- 정수와 같은 기초 자료형도 객체로 포장하고 싶은 경우가 있다. 이때 사용되는 클래스가 래퍼 클래스(Wrapper Class)이다.

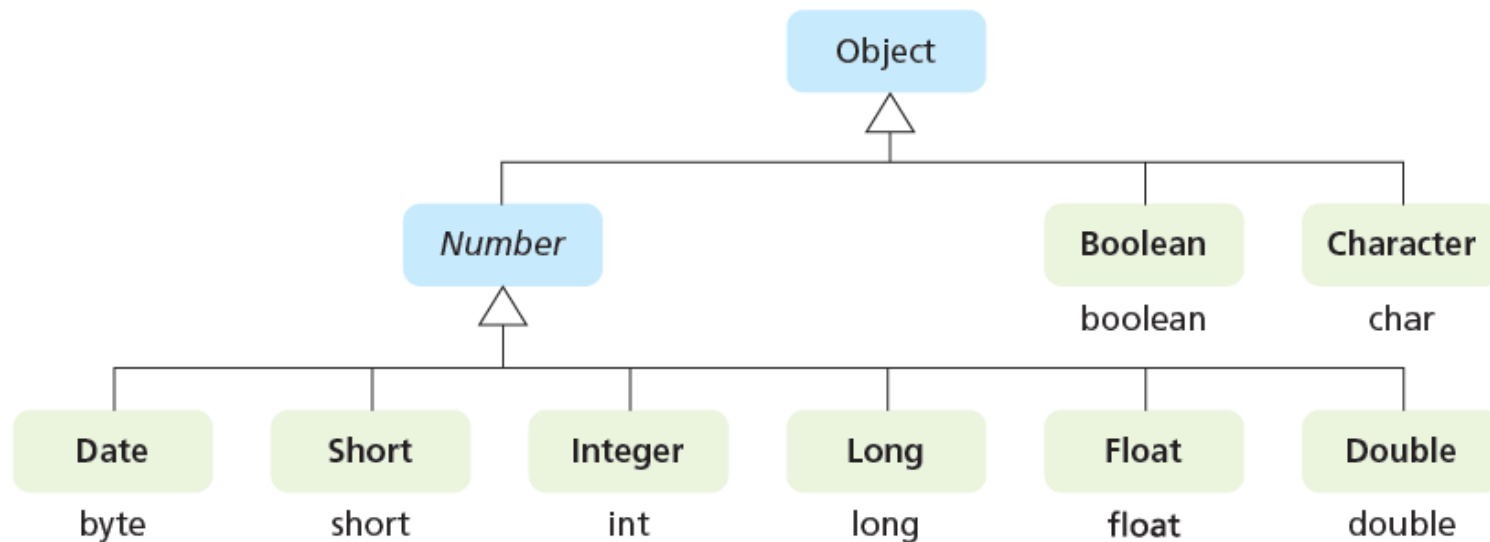
```
int i = 100;  
Integer obj = new Integer(i);
```

```
int i = 100; Integer obj = Integer(i);
```





래퍼 클래스의 종류





랩퍼 클래스의 메소드

반환값	메소드 이름	설명
static int	intValue()	int형으로 반환한다.
static double	doubleValue()	double형으로 반환한다.
static float	floatValue()	float형으로 반환한다.
static int	parseInt(String s)	문자열을 int형으로 변환한다.
static String	toBinaryString(int i)	int형의 정수를 2진수 형태의 문자열로 변환한다.
static String	toString(int i)	int형의 정수를 10진수 형태의 문자열로 변환한다.
static Integer	valueOf(String s)	문자열 s를 Integer 객체로 변환한다.
static Integer	valueOf(String s, in radix)	문자열 s를 radix진법의 Integer 객체로 변환한다.



래퍼 클래스 메소드의 예

- 래퍼 객체 안에 포장된 정수 100을 꺼내려면 `intValue()` 함수를 사용한다.
 - `Integer obj = new Integer(100);`
 - `int value = obj.intValue();` // value는 정수 100
- 만약 정수 100을 문자열로 변환하고 싶다면 `Integer` 클래스의 `toString()` 메소드를 이용한다.
 - `String s = Integer.toString(100);` // s는 "100"
- 문자열 "100"을 정수 100으로 변환하려면 `Integer` 클래스의 `parseInt()` 메소드를 사용한다.
 - `int i = Integer.parseInt("100");`



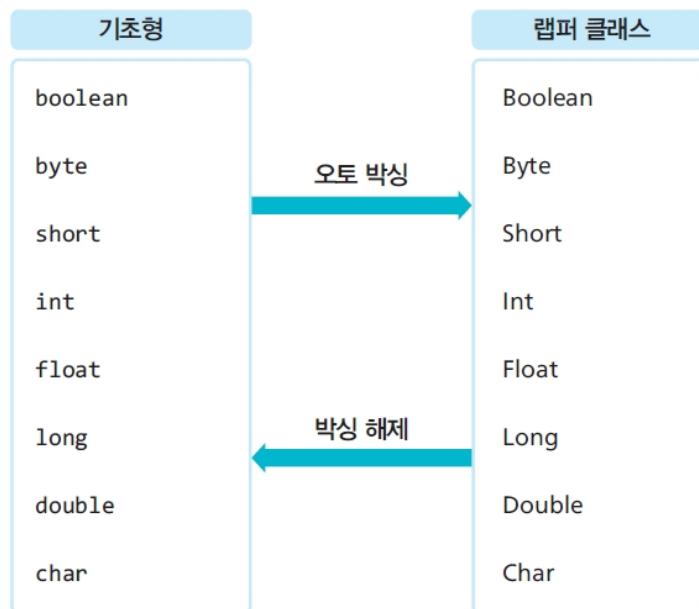
오토 박싱

- 랩퍼 객체와 기초 자료형 사이의 변환을 자동으로 해주는 기능

```
Integer obj;
```

```
obj = 10; // 정수 -> Integer 객체
```

```
System.out.println(obj + 1); // Integer 객체 -> 정수
```





중간점검



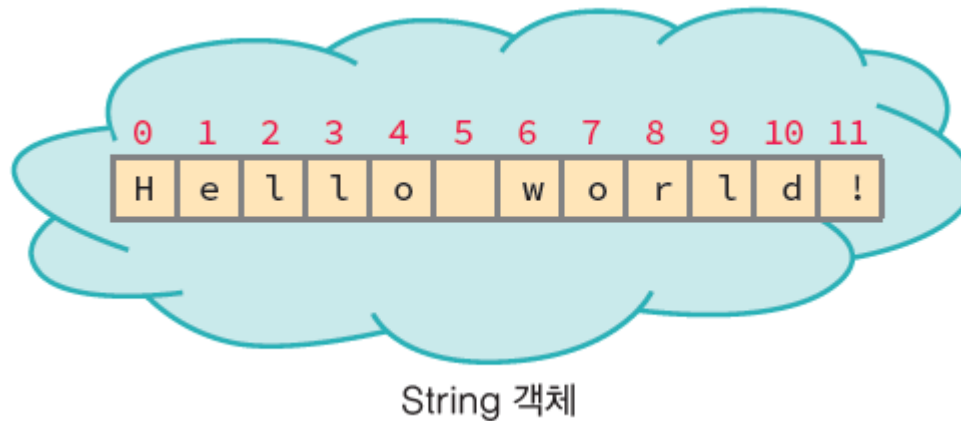
중간점검

1. 자바에서 정수 10과 Integer(10)은 어떻게 다른가?
2. 3.14를 적절한 래퍼 클래스에 저장하는 문장을 작성하라.
3. 3.14가 저장된 래퍼 객체에서 3.14를 추출하는 문장을 작성하라.
4. 문자열 "3,14"을 부동소수점 수로 변환하는 문장을 작성하라.
5. 오토박싱이란 무엇인가?



String 클래스

- String 클래스는 문자열을 구성하는 문자들을 내부에 저장한다.





String 클래스

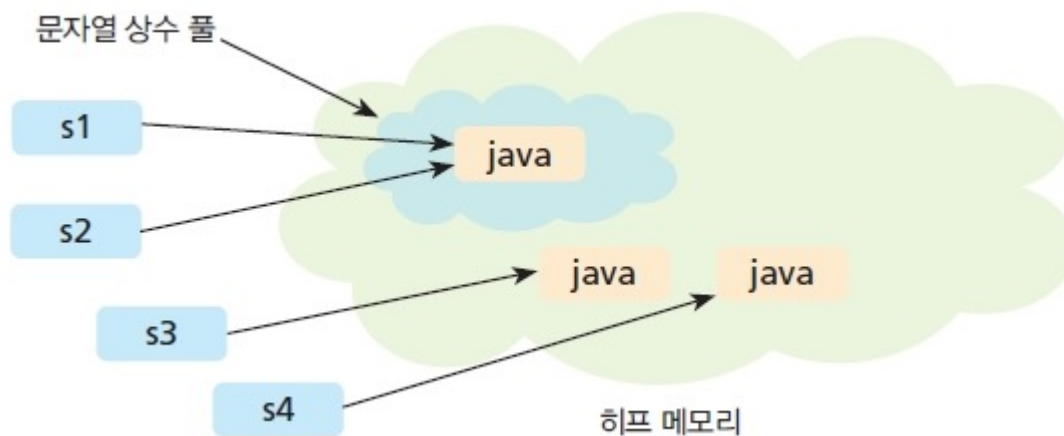
메소드 요약

char	charAt (int index) 지정된 인덱스에 있는 문자를 반환한다.
int	compareTo (String anotherString) 사전적 순서로 문자열을 비교한다. 앞에 있으면 -1, 같으면 0, 뒤에 있으면 1이 반환된다.
String	concat (String str) 주어진 문자열을 현재의 문자열 뒤에 붙인다.
boolean	equals (Object anObject) 주어진 객체와 현재의 문자열을 비교한다.
boolean	isEmpty () length()가 0이면 true를 반환한다.
int	length () 현재 문자열의 길이를 반환한다.
String	toLowerCase () 문자열의 문자들을 모두 소문자로 변경한다.
String	toUpperCase () 문자열의 문자들을 모두 대문자로 변경한다.



String 클래스

```
String s1 = "Java";    // (1) 많이 사용되는 방법  
String s2 = "Java";  
String s3 = new String("Java");    // (2) 원칙임  
String s4 = new String("Java");
```





문자열의 기초 연산

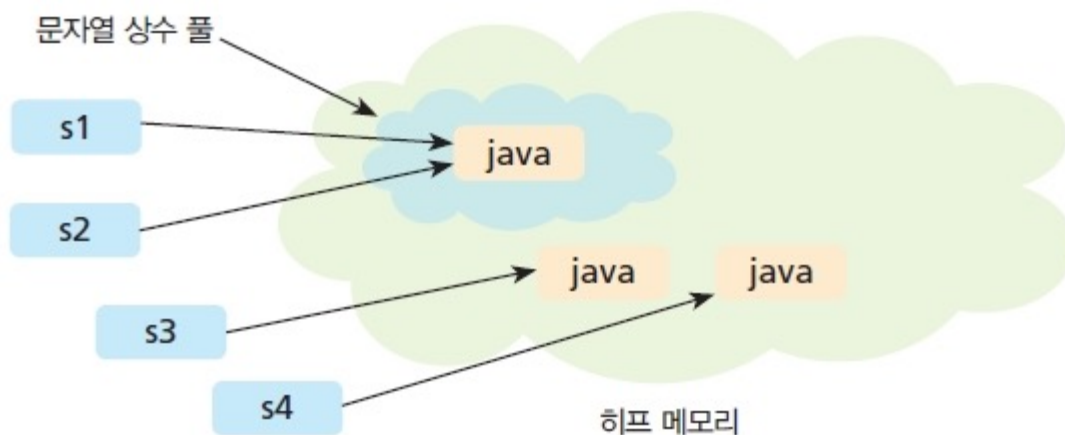
- String 클래스의 `length()` 메소드는 문자열의 길이를 반환한다.
 - `String s = "Hello World!";` // 객체 생성
 - `int size = s.length();` // 12가 반환됨
- String 객체 안에 들어 있는 문자를 추출하려면 `charAt()`이라는 메소드를 호출하면 된다. 문자 번호는 0부터 시작한다.
 - `String s = "Hello World!";` // 객체 생성
 - `char c = s.charAt(0);` // 'H'가 반환된다.
- 자바에서 2개의 문자열을 붙이려면 물론 `concat()`라는 메소드를 호출하여도 되지만 + 연산자를 사용하는 것이 더 편리하다.
 - `String result = "A chain" + " is only as strong" + "as its weakest link";`



문자열 비교하기

```
String s1 = "Java"; // (1) 많이 사용되는 방법
String s2 = "Java";
String s3 = new String("Java"); // (2) 원칙임
String s4 = new String("Java");

System.out.println(s1.equals(s2)); // true, 올바른 방법
System.out.println(s1.equals(s3)); // true, 올바른 방법
System.out.println(s1==s2); // true이지만 올바르지 않은 방법
System.out.println(s1==s3); // false, 올바르지 않은 방법
```





문자열 안에서 단어 찾기

```
String s = "The cat is on the table";  
int index = s.indexOf("table");  
  
if(index == -1)  
    System.out.println("table은 없습니다. ");  
else  
    System.out.println("table의 위치: " + index);
```

table의 위치: 18



문자열을 단어로 분리할 때

- 예전에 자바에서 문자열을 단어들로 분리할 때는 **StringTokenizer** 클래스를 사용하였다. 하지만 최근 버전에서는 **String** 클래스가 제공하는 **split()** 메소드를 사용하면 훨씬 쉽게 문자열을 단어로 분리할 수 있다.

```
String[] tokens = "I am a boy.".split(" ");  
for (String token : tokens)  
    System.out.println(token);
```

```
I  
am  
a  
boy.
```

```
String[] tokens = "100,200,300".split(",");  
for (String token : tokens)  
    System.out.println(token);
```

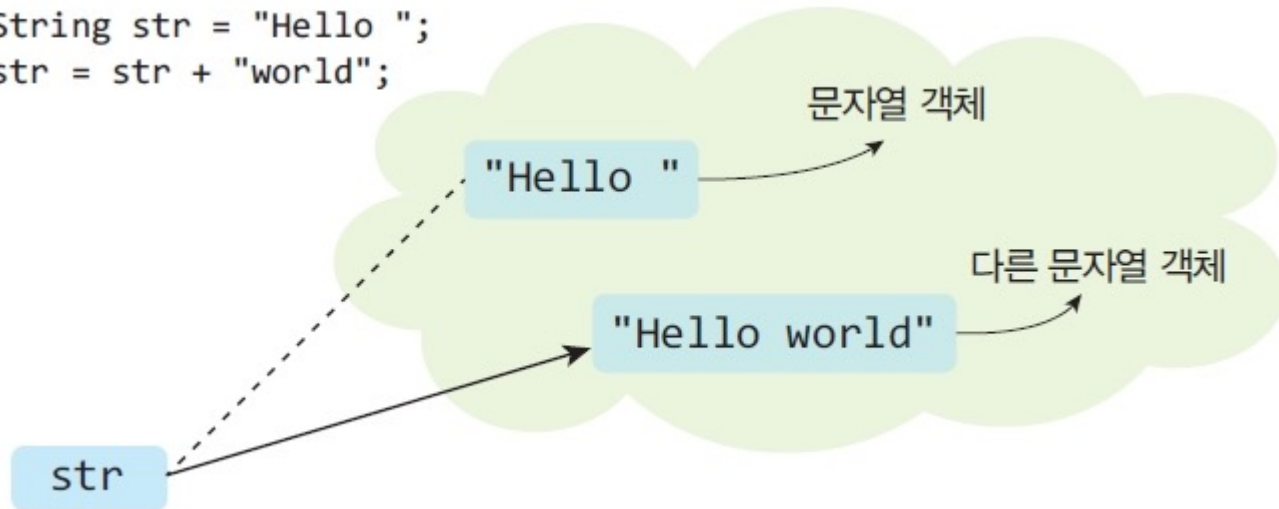
```
100  
200  
300
```



StringBuffer 클래스

- String 클래스의 경우, 빈번하게 문자열을 변경할 때에는 비효율적일 수 있다. 왜냐하면 문자열의 내용을 변경하는 String 클래스 메소드의 경우, 새로운 String 객체를 생성하고 기존의 내용을 복사해야 하기 때문이다

```
String str = "Hello ";  
str = str + "world";
```





StringBuffer 클래스

- 자바는 변경 가능한 문자열을 위하여 **String** 클래스의 대안으로 **StringBuffer** 클래스를 제공한다.

```
StringBuffer s = new StringBuffer("Happiness depends upon ourselves");
```

	String	StringBuffer
메모리 위치	String pool	Heap
수정 여부	No(immutable)	Yes(mutable)
스레드 안전성	Yes	Yes
동기화 여부	Yes	Yes
성능	Fast	Slow



StringBuffer 클래스

- **StringBuffer**만이 제공하는 가장 중요한 메소드는 **append()**와 **insert()**이다. 이 두 메소드는 어떤 타입의 데이터도 받을 수 있도록 중복 정의되어 있다.
각

```
StringBuffer sb = new StringBuffer("Hello"); // 16바이트의 공간이 할당된다.  
int length = sb.length(); // 5  
int capacity = sb.capacity(); // 21
```

```
StringBuffer sb = new StringBuffer("10+20=");  
sb.append(10+20);  
sb.insert(0, "수식 "); // sb= "수식 10+20=30"
```



중간점검

1. StringBuffer와 String은 어떻게 다른가?
2. 다음 문장이 실행되면 sb에는 어떤 문자열이 저장되는가?

```
StringBuffer sb = new StringBuffer("Hello");  
sb.append(1+2);
```



중간점검



Math 클래스

- **Math** 클래스는 지수나 로그, 제곱근, 삼각함수와 같은 기본적인 수치 연산을 위한 메소드들을 제공한다.

MathTest.java

```
01 public class MathTest {  
02     public static void main(String[] args){  
03         double x = Math.PI;  
04         System.out.println(Math.sin(x));  
05         System.out.println(Math.random());  
06     }  
07 }
```

1.2246467991473532E-16

0.8082738632799703



Random 클래스

- Random 클래스의 객체는 난수를 발생하는데 사용된다.

RandomTest.java

```
01 import java.util.*;
02 public class RandomTest {
03     public static void main(String[] args)    {
04         Random random = new Random();
05         for (int i = 0; i < 10; i++)
06             System.out.print(random.nextInt(100)+"");
07     }
08 }
```

12,48,7,4,15,77,84,60,8,62,



Array 클래스

ArraysTest.java

```
01  import java.util.*;
02
03  public class ArraysTest {
04      public static void main(String[] args) {
05          int[] array = { 9, 4, 5, 6, 2, 1 };
06          Arrays.sort(array);           // 배열을 정렬한다.
07          printArray(array);
08          System.out.println(Arrays.binarySearch(array,9)); // 9을 탐색한다.
09          Arrays.fill(array,8);         // 배열을 특정한 값으로 채운다.
10          printArray(array);
11      }
12      private static void printArray(int[] array) {
13          System.out.print("[");
14          for(int i=0 ;i< array.length;i++)
15              System.out.print(array[i]+" ");
16          System.out.println("]");
17      }
18  }
```

[1 2 4 5 6 9]

5

[8 8 8 8 8 8]



Calendar 클래스

- Calendar 클래스는 추상 클래스로서 날짜와 시간에 대한 정보를 가지고 있고 특정 시각을 연도, 월, 일 등으로 변환하는 메소드도 가지고 있다

CalendarTest.java

```
01 import java.util.*;
02
03 public class CalendarTest {
04     public static void main(String[] args) {
05         Calendar d = Calendar.getInstance();    // 객체 생성
06         System.out.println(d);
07         System.out.println(d.get(Calendar.YEAR)+"년");
08         System.out.println(d.get(Calendar.MONTH)+1 +"월");
09         System.out.println(d.get(Calendar.DATE)+"일");
10
11         d.set(Calendar.HOUR, 12);
12         d.set(Calendar.MINUTE, 34);
13         d.set(Calendar.SECOND, 0);
14         System.out.println(d);
15     }
16 }
```

```
java.util.GregorianCalendar[time=1633217253036,...]
2021년
10월
3일
java.util.GregorianCalendar[...]
```



중간점검



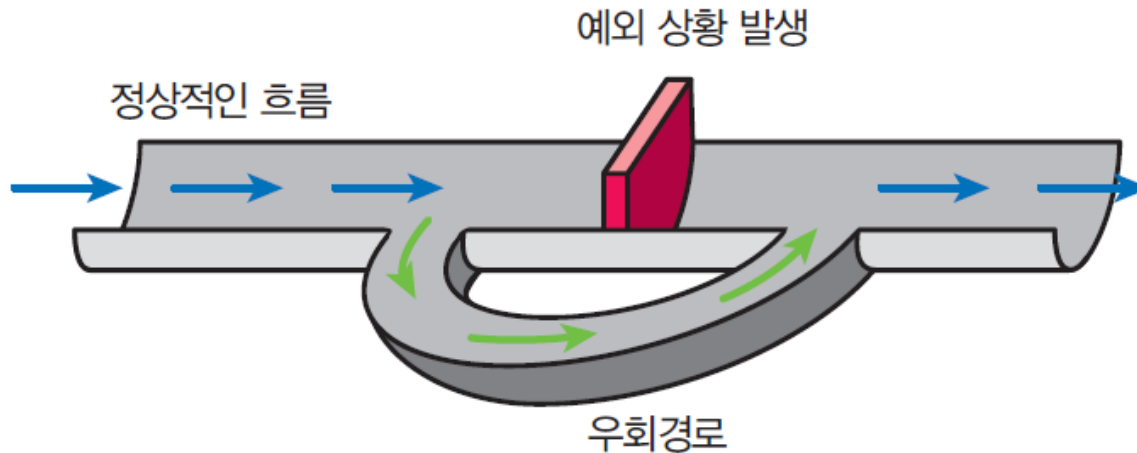
중간점검

1. 1에서 10 사이의 난수를 생성하여 크기가 10인 배열 a를 채우는 문장을 작성하시오.
2. `Arrays.sort()` 메소드를 이용하여 위의 배열을 정렬해보자.



예외처리

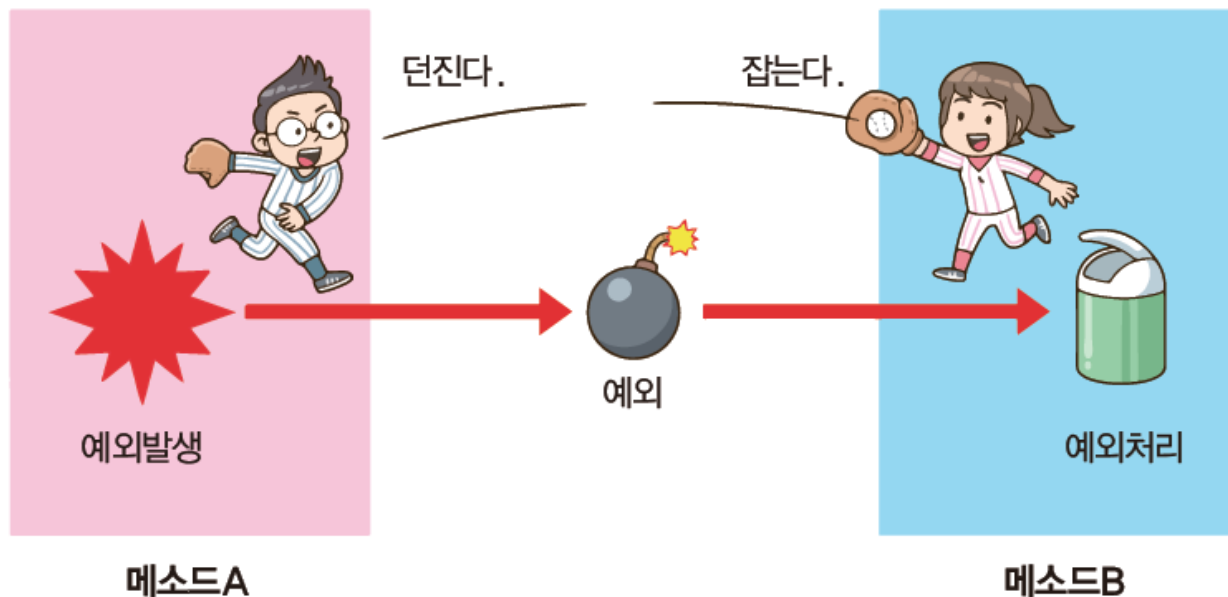
- 오류가 발생했을 때 오류를 사용자에게 알려주고 모든 데이터를 저장하게 한 후에 사용자가 우아하게(**gracefully**) 프로그램을 종료할 수 있도록 하는 것이 바람직하다





예외란?

- 예외(exception): 잘못된 코드, 부정확한 데이터, 예외적인 상황에 의하여 발생하는 오류
 - (예) 0으로 나누는 것과 같은 잘못된 연산이나 배열의 인덱스가 한계를 넘을 수도 있고, 디스크에서는 하드웨어 에러가 발생할 수 있다.





예외 발생 예제

DivideByZero.java

```
01 public class DivideByZero {  
02     public static void main(String[] args) {  
03         int result = 10 / 0;  
04         System.out.println("나눗셈 결과: " + result);  
05     }  
06 }
```

Exception in thread "main" java.lang.ArithmeticException: / by zero
at DivideByZero.main(DivideByZero.java:14)



try-catch 블록

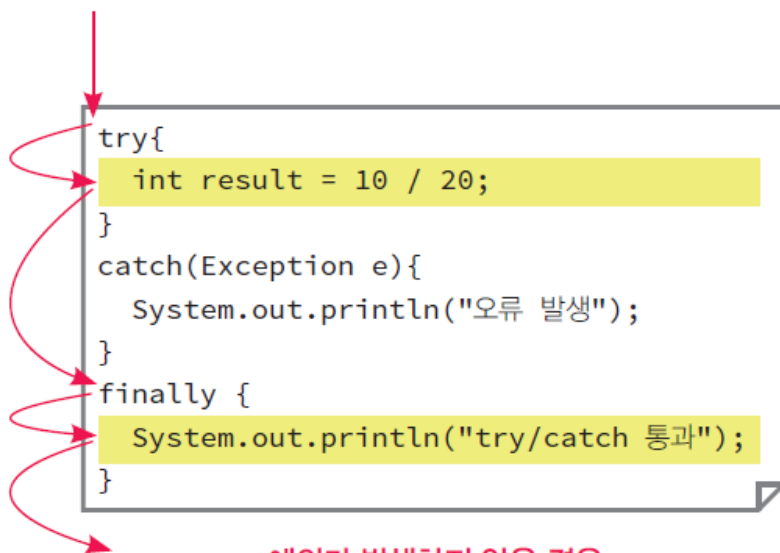
Syntax: 예외 처리 구조

```
try {  
    // 예외가 발생할 수 있는 코드  
} catch (예외클래스 변수) {  
    // 예외를 처리하는 코드  
}  
} finally {  
    // 여기 있는 코드는 try 블록이 끝나면 무조건 실행된다.  
}
```

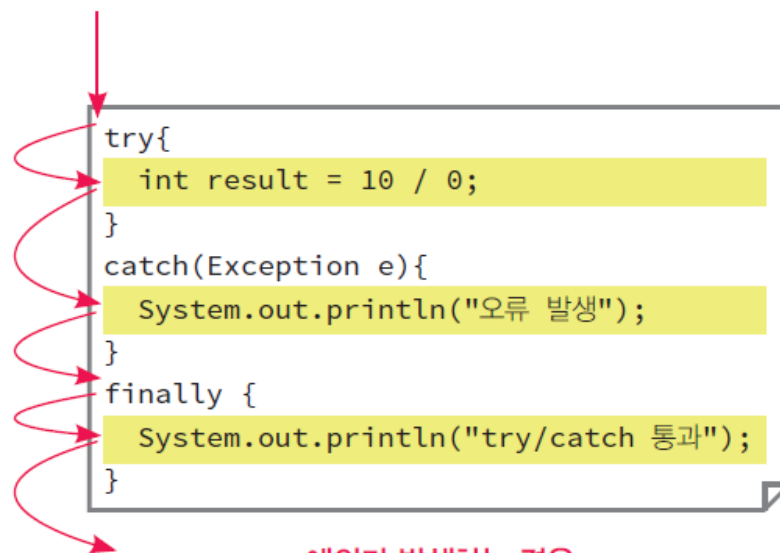
생략이 가능하다.



try/catch 블록에서의 실행 흐름



예외가 발생하지 않은 경우



예외가 발생하는 경우



0으로 나누는 예외 처리

DivideByZeroOK.java

```
07 import java.util.Scanner;
08
09 public class DivideByZeroOK {
10     public static void main(String[] args) {
11         try {
12             int result = 10 / 0;    // 예외 발생!
13         } catch (ArithmeticException e) {
14             System.out.println("0으로 나눌 수 없습니다.");
15         }
16         System.out.println("프로그램은 계속 진행됩니다.");
17     }
18 }
```

여기서 오류를 처리한다. 현재는 그냥 콘솔에 오류 메시지를 출력하고 계속 실행한다.

0으로 나눌 수 없습니다.

프로그램은 계속 진행됩니다.



예외의 종류



예외



컴파일러

Error

너무 심각해서 할 수 있는
방법이 없음
→ 통과

RuntimeException

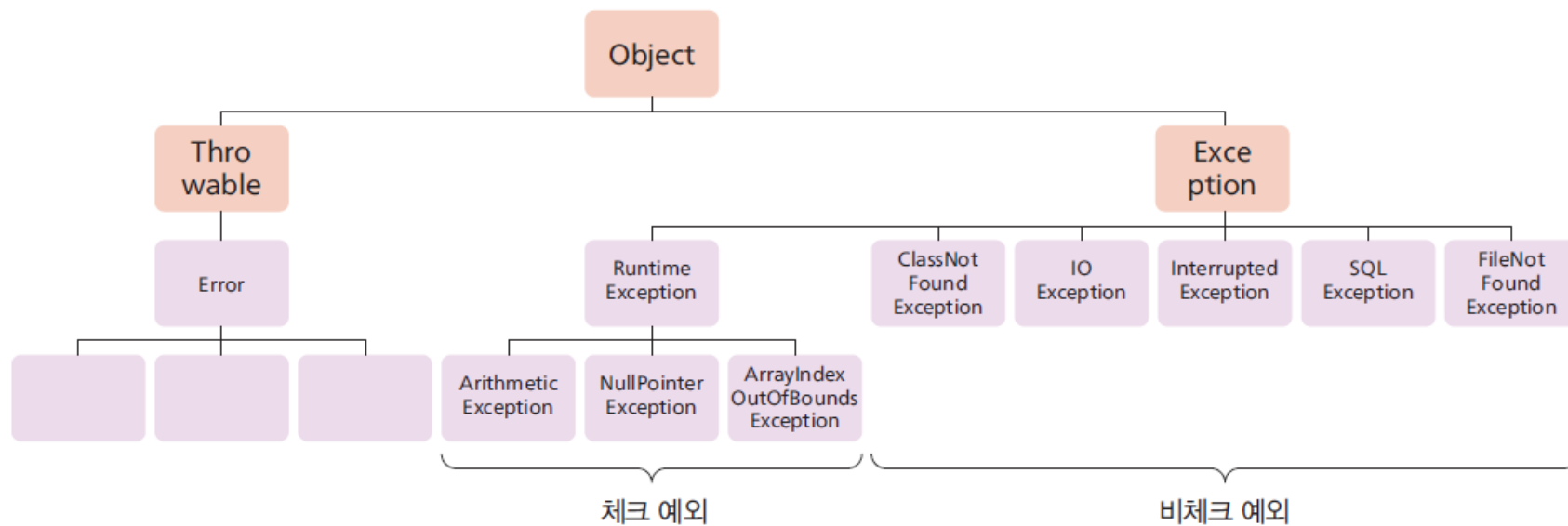
프로그래밍 버그이므로
스스로 고쳐야 함
→ 통과

Error나

RuntimeException이 아닌 예외
반드시 처리해야 함!!
→ 검사



예외의 종류





RuntimeException

분류	예외	설명
Runtime-Exception	ArithmeticException	어떤 수를 0으로 나누는 경우
	NullPointerException	널 객체를 참조할 때 발생하는 경우
	ClassCastException	적절치 못하게 클래스를 형변환하는 경우
	NegativeArraySizeException	배열의 크기가 음수값인 경우
	OutOfMemoryException	사용 가능한 메모리가 없는 경우
	NoClassDefFoundException	원하는 클래스를 찾지 못하였을 경우
	ArrayIndexOutOfBoundsException	배열을 참조하는 인덱스가 잘못된 경우



CheckedException

분류	예외	설명
Checked-Exception	ClassNotFoundException	클래스가 발견되지 않을 때
	IOException	입출력 오류
	illegalAccessException	클래스의 접근이 금지되었을 때
	NoSuchMethodException	메소드가 발견되지 않을 때
	NoSuchFieldException	필드가 발견되지 않을 때
	InterruptedException	스레드가 다른 스레드에 의하여 중단되었을 때
	FileNotFoundException	파일을 찾지 못했을 때



예제: 배열 인덱스 예외 처리

이번에는 배열에서 인덱스가 배열의 크기를 벗어나는 경우에 발생하는 예외를 처리하여 보자. 예외의 이름은 `ArrayIndexOutOfBoundsException`이다. 이 예외를 처리해보자.

1 2 3 4 5 인덱스 5는 사용할 수 없네요!

ArrayError.java

```
01 public class ArrayError {
02     public static void main(String[] args) {
03         int[] array = { 1, 2, 3, 4, 5 };
04         int i = 0;
05
06         try {
07             for (i = 0; i <= array.length; i++)
08                 System.out.print(array[i] + " ");
09         } catch (ArrayIndexOutOfBoundsException e) {
10             System.out.println("인덱스 " + i + "는 사용할 수 없네요!");
11         }
12     }
13 }
```




예제: 입력 예외 처리

다음과 같은 프로그램에서는 NumberFormat 예외가 발생할 수 있다. 이것을 처리하여 보자.

```
int num = Integer.parseInt("ABC");  
System.out.println(num);
```

Exception in thread "main" java.lang.NumberFormatException:

다음과 같이 try-catch 구조를 사용하여 예외를 처리할 수 있다.

ExceptionTest3.java

```
01 public class ExceptionTest3 {  
02     public static void main(String args[]) {  
03         try {  
04             int num = Integer.parseInt("ABC");  
05             System.out.println(num);  
06         } catch (NumberFormatException e) {  
07             System.out.println("NumberFormat 예외 발생");  
08         }  
09     }  
10 }
```



try-with-resources 문장

- try-with-resources 문장은 문장의 끝에서 리소스들이 자동으로 닫혀지게 한다. try-with-resources 문장은 Java SE 7버전부터 추가되었다.

TryTest.java

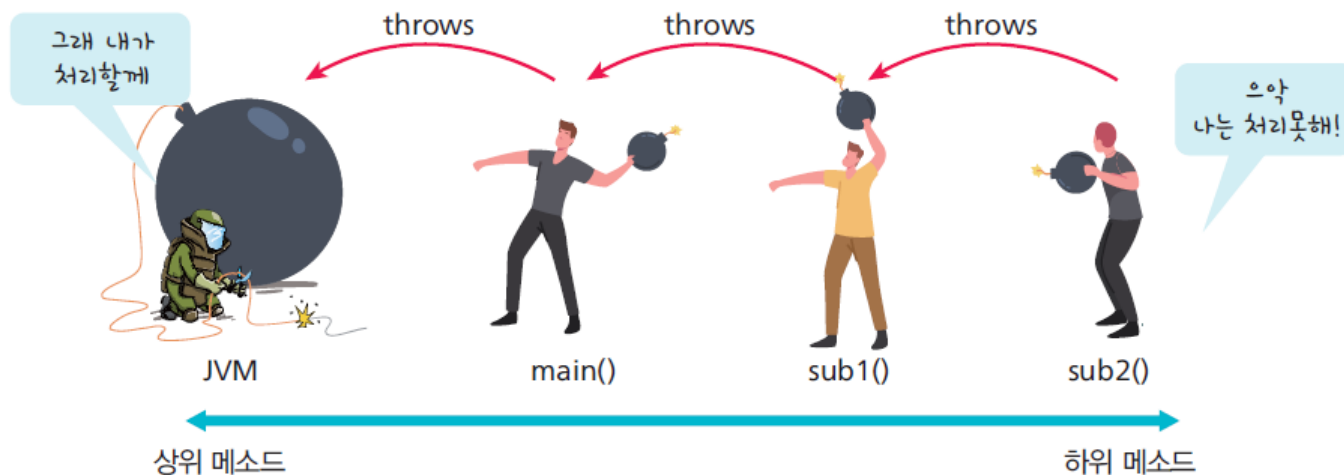
```
01  import java.io.*;
02
03  public class TryTest {
04      public static void main(String args[]) {
05          try (FileReader fr = new FileReader("test.txt")) {
06              char[] a = new char[50];
07              fr.read(a);
08              for (char c : a)
09                  System.out.print(c);
10          } catch (IOException e) {
11              e.printStackTrace();
12          }
13      }
14  }
```

괄호가 있으면 자원으로
취급한다.



예외를 떠넘기기

- 가끔은 메소드가 발생하는 예외를 그 자리에서 처리하지 않고, 자신을 호출한 상위 메소드로 예외를 전달하는 편이 더 적절할 수도 있다.





예외를 던지기

ExceptionTest.java

```
01  import java.io.*;
02
03  public class ExceptionTest {
04
05      public static void main(String args[]) throws IOException {
06          FileReader fr = new FileReader("test.txt");
07          char[] a = new char[50];
08          fr.read(a);
09          for (char c : a)
10              System.out.print(c);
11      }
12  }
```

입출력 예외가 발생하면 상위 메소드로 예외를 던져서 처리하겠다는 의미이다.



중간점검

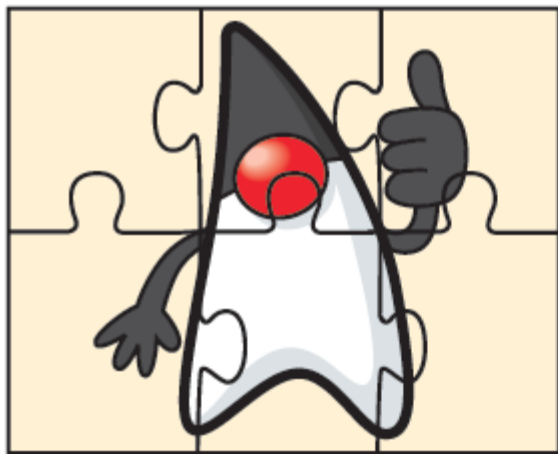


중간점검

1. 예외는 어떤 경우에 발생하는가?
2. 예외를 처리하는 경우와 처리하지 않는 경우를 비교하여 보라. 장점은 무엇인가?
3. 예외가 발생했을 때 바로 종료하는 것과 try-catch 예외 처리 구조를 사용하는 것을 비교해보자. 어떤 방법이 더 장점이 많은가?
4. try-catch 예외 처리 구조에서 예외를 처리하는 문장은 어디에 놓아야 하는가? 또 예외를 발생하는 문장은 어디에 두어야 하는가?
5. 정수를 입력받을 때, 사용자가 문자 형태로 입력하면 InputMismatchException 예외가 발생한다. 이 예외를 처리하는 코드를 작성해보자.
6. 배열에서 인덱스가 범위를 벗어나도 예외가 발생한다. 크기가 10인 배열을 생성하고 11번째 원소에 0을 대입하여 보라. 이 예외를 처리하는 try-catch 블록을 만들어 보라.
7. Error와 RuntimeException은 언제 발생하는가?
8. 자바 코드에서 반드시 처리하여야 하는 예외는 어떤 것들인가?
9. RuntimeException을 처리했는지를 왜 컴파일러에서는 검사하지 않는가?
10. 만약 호출한 메소드가 예외를 발생할 가능성이 있다면 어떻게 하여야 하는가?
11. 예외를 발생할 가능성이 있는 메소드는 어떻게 정의되는가?
12. throws가 의미하는 것은 무엇인가?

모듈

- 자바 모듈(module)은 여러 가지 자바 패키지들을 하나의 단위(모듈)에 포장할 수 있는 메커니즘이다. 자바 플랫폼 모듈 시스템(JPMS: Java Platform Module System)은 직소 프로젝트의 결과물로서 **Java 9** 버전의 주요 변경 사항이다. 이전 버전의 자바에는 모듈 개념이 없었기 때문에 애플리케이션의 크기가 커지고 이동하기 어려웠다. 자바 **API** 자체도 크기가 너무 무거워서 **Java 8**에서 **rt.jar** 파일 크기는 약 **64MB**나 되었다.



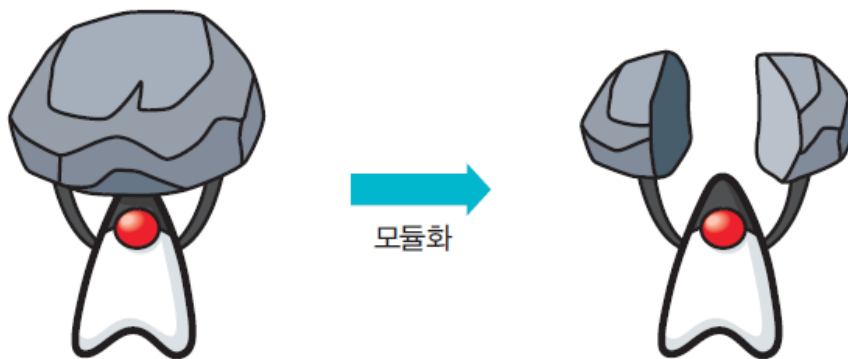
필요한 모듈만 패키징하여서
애플리케이션의 크기를 줄일 수 있습니다.





직소 프로젝트

- 직소(Jigsaw) 프로젝트는 2008년에 시작되어서 2017년에 완료된 OpenJDK의 프로젝트이다.
- 구체적으로 자바 **API**를 모듈로 분리하여서 소형 컴퓨팅 장치에서도 사용하도록 축소하는 것이었다. 그동안에는 기존에는 자바 **API**의 일부분만 배포할 수가 없었다. 즉 애플리케이션이 사용하지 않더라도 무조건 **rt.jar** 파일(XML, SQL, Swing 등이 포함되어있다)이 항상 같이 배포되었다. 따라서 메모리가 작은 임베디드 장치의 경우, 자바를 실행하기가 쉽지 않았다.





모듈화의 장점



자신이 필요한 모듈만 골라서 실행 파일로 묶을 수 있다.



자바 모듈은 외부로 노출되는 패키지와 노출되지 않는 패키지를 지정할 수 있다. 노출되지 않은 패키지는 다른 자바 모듈에서 사용할 수 없다. 이러한 패키지는 자바 모듈에서 내부적으로만 사용할 수 있다. 패키지 레벨에서도 캡슐화가 가능하다.

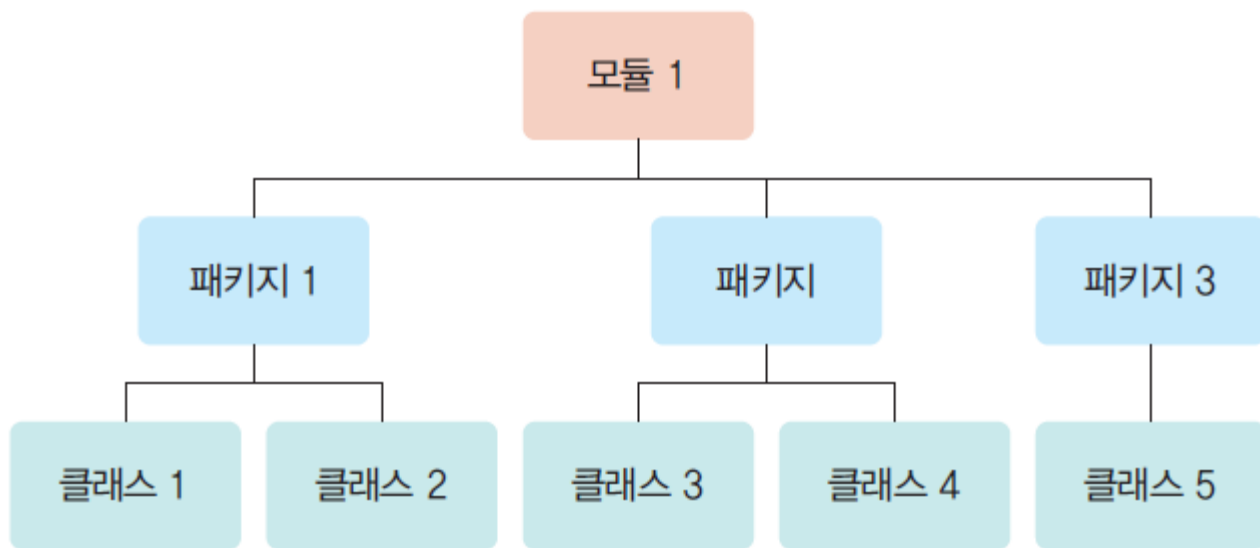


Java 9 이상부터는 자바 애플리케이션은 모듈로 패키징해야 한다. 애플리케이션은 자신이 필요한 모듈을 미리 지정하여야 한다. 따라서 자바 가상 머신(JVM)은 시작할 때 애플리케이션 모듈에서 전체 모듈 종속성 그래프를 확인할 수 있다. 시작 시 필요한 모듈이 없으면 JVM은 누락된 모듈을 보고하고 종료한다. Java 9 이전에는 애플리케이션이 실제로 누락된 클래스를 사용하려고 시도할 때까지, 누락된 클래스가 감지되지 않았다.



모듈의 개념

- 모듈은 하나 이상의 자바 패키지로 이루어진다. 모듈은 패키지의 상위 개념이다.





모듈의 개념

- 모듈에서는 지정된 패키지를 요구할 수도 있고, 자신이 가진 패키지를 남들이 사용할 수 있도록 허가할 수도 있다. 이것은 모듈마다 하나씩 있어야 하는 `module-info.java`에서 지정한다.

module-info.java

```
module com.example {  
    exports com.example;  
    requires java.base;  
}
```





자바 모듈 컴파일

```
d:\> javac -d mods --module-source-path src/ --module com.example
```

컴파일 후 다음과 같은 구조의 디렉터리가 생성된다.

```
d:\mods
  ---> com.example
    ---> com
      ---> example
        ---> Hello.class
        ---> module-info.class
```



자바 모듈 실행

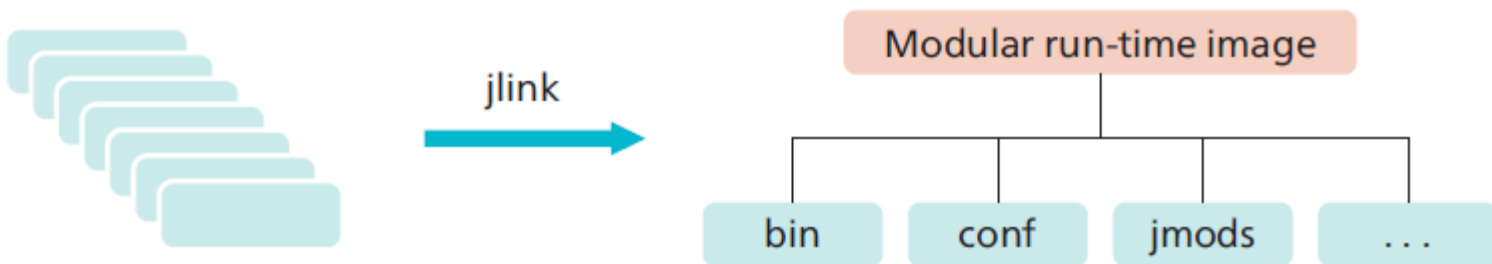
자바 모듈 안의 클래스를 실행하려면 다음과 같은 명령의 java를 사용한다.

```
d:\> java --module-path mods/ --module com.example/com.example.Hello
```



독립 실행형 애플리케이션으로 자바 모듈 패키징

- 자신이 만든 자바 모듈을 필요한 자바 **API**와 함께 독립 실행형 애플리케이션으로 패키징할 수 있다. 이러한 독립 실행형 애플리케이션은 애플리케이션에 자바가 포함되어 제공되므로 애플리케이션을 실행하기 위해 자바 **JRE**를 미리 설치할 필요가 없다. 즉, 애플리케이션이 실제로 사용하는 만큼의 자바 플랫폼 **API**를 포함한다.





Jlink 사용하기

```
d:\> jlink --module-path "mods;C:\Program Files\Java\jdk-16.0.1\jmods" --add-modules  
com.example --output out-standalone
```

- --module-path 인수는 모듈을 찾기 위한 모듈 경로를 지정한다. 예제에서는 모듈이 컴파일되는 디렉터리 mods와 JDK 설치 디렉터리의 jmods를 지정한다.
- --add-modules 인수는 독립 실행형 애플리케이션으로 패키지에 자바 모듈을 지정한다. 위의 예에는 com.powerjava 모듈만 포함되어 있다.
- --output 인수는 독립형 자바 애플리케이션이 생성되는 디렉터를 지정한다. 디렉터리가 존재하지 않아야 한다.



독립 실행형 애플리케이션 실행

```
d:\> cd out-standalone
```

```
d:\out-standalone> bin\java --module com.example/com.example.Hello
```

```
Hello from module
```



참고

이름 없는 모듈

Java 9 이상부터 모든 자바 클래스는 모듈에 있어야 한다. 그러나 이전 버전의 자바 클래스라면 어떻게 해야 하는가? Java 9 이후 버전에서도 애플리케이션을 실행할 때 가상 머신에 대한 인수 `-classpath`를 계속 사용할 수 있다. 클래스 경로에는 Java 9 이전과 마찬가지로 이전 자바 클래스를 모두 포함할 수 있다. 클래스 경로에 있는 모든 클래스는 자바가 “이름 없는 모듈”이라고 하는 클래스에 포함된다.



- • •





Summary

- 패키지(package)는 관련있는 클래스들을 하나로 묶은 것이다. 패키지를 사용하여 이름 충돌을 피하고 유지 관리가 쉬운 코드를 작성할 수 있다.
- 패키지를 선언하는 문장 “package 패키지이름;”은 소스 파일의 첫 번째 줄로 추가한다.
- JAR 파일을 사용하면 자바 파일들을 압축하여 하나의 파일로 만들 수 있다.
- 클래스를 찾기 위하여 가상 머신이 둘러보는 디렉터리들을 모아둔 경로가 클래스 경로(class path)이다.
- 클래스를 정의할 때 명시적으로 부모 클래스를 선언하지 않으면 Object 클래스가 부모 클래스가 된다.
- 기초 자료형을 객체로 포장할 때 사용되는 클래스가 랩퍼 클래스(Wrapper Class)이다.
- 2개의 문자열이 동일한지 검사하려면 String 클래스의 equals()을 호출하여야 한다.
- 예외 처리를 사용하면 오류가 발생했을 때, 우아하게 종료하거나 계속 실행할 수 있다.
- 예외는 try-catch 블록을 이용하여 예외를 잡아서 처리할 수도 있고 상위 메소드로 예외를 전달할 수도 있다.





Q & A

