

제 6장 복수 클래스를 사용한 개발

소스 파일의 분리

대규모 개발에서는 혼자서 개발이 어렵고

분담을 하여 소스를 모듈화 해야 한다.

1개의 소스파일로는 개발의 한계를 만날 것이다.

```
1 public class Calc {
2     public static void main(String[] args) {
3         int a = 10;
4         int b = 2;
5         int total = add(a, b);
6         int delta = minus(a, b);
7         System.out.println("더하면 " + total + ", 빼면 " + delta);
8     }
9
10    private static int add(int a, int b) {
11        return a + b;
12    }
13
14    private static int minus(int a, int b) {
15        return a - b;
16    }
17 }
18
```

```
1 public class CalcLogic {  
2  
3 @      static int add(int a, int b) {  
4         return a + b;  
5     }  
6  
7 @      static int minus(int a, int b) {  
8         return a - b;  
9     }  
10 }
```

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         int a = 10;  
4         int b = 2;  
5         int total = add(a, b);  
6         int delta = minus(a, b);  
7         System.out.println("더하면 " + total + ", 빼면 " + delta);  
8     }  
9 }
```

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         int a = 10;  
4         int b = 2;  
5         int total = CalcLogic.add(a, b);  
6         int delta = CalcLogic.minus(a, b);  
7         System.out.println("더하면 " + total + ", 빼면 " + delta);  
8     }  
9 }
```


패키지 (package)

Java에서는 각 클래스를 **패키지** (package) 라는 그룹에 소속시켜, 분류, 관리가 되도록 하는 장치가 준비 되어 있다.

main() 메소드의 라인수가 늘어나면 복수의 메소드로 분리 ->

메소드 수가 늘어나면 복수의 클래스로 분리 ->

클래스 수가 늘어나면 복수의 패키지로 분리

```
package calcapp.main;  
     public class Calc {
```



```
package calcapp.logics;  
  
public class CalcLogic {
```

```
1  package calcapp.main;
2
3  public class Calc {
4      public static void main(String[] args) {
5          int a = 10;
6          int b = 2;
7          int total = calcapp.logics.CalcLogic.add(a, b);
8          int delta = calcapp.logics.CalcLogic.minus(a, b);
9          System.out.println("더하면 " + total + ", 빼면 " + delta);
10     }
11 }
```

이름 공간 (name space)

- 자신이 작성한 클래스에 대해, 개발자가 자유로운 이름을 지을 수 있다
- 패키지명 자체의 충돌을 피하려면 보유한 도메인의 앞뒤를 바꿔서 패키지명으로 사용

```
1  package calcapp.main;
2
3  import calcapp.logics.CalcLogic;
4
5  public class Calc {
6      public static void main(String[] args) {
7          int a = 10;
8          int b = 2;
9          int total = CalcLogic.add(a, b);
10         int delta = CalcLogic.minus(a, b);
11         System.out.println("더하면 " + total + ", 빼면 " + delta);
12     }
13 }
```

Java API 에 대해

여러 사람이 합심하여 완성한 Java

약 200개 이상의 패키지, 3500개 이상의 클래스가 준비되어 있고

이것들을 **API** (Application Programming Interface) 라고 한다.

```
1  package calcapp.main;
2
3  import java.util.Arrays;
4
5  public class Calc {
6      public static void main(String[] args) {
7          int[] heights = {172, 149, 152, 191, 155};
8          Arrays.sort(heights);
9          for (int h : heights) {
10             System.out.println(h);
11         }
12     }
13 }
```

java.lang	Java 에서 가장 중요한 클래스군 (자동 import)
java.util	프로그래밍을 편리하게 해 주는 유용한 클래스군
java.math	수학에 관한 클래스군
java.net	네트워크 통신등에 필요한 클래스군
java.io	파일 입출력 등에 필요한 클래스군

Java™ Platform
Standard Ed. 8[All Classes](#) [All Profiles](#)

Packages

[java.applet](#)
[java.awt](#)
[java.awt.color](#)
[java.awt.datatransfer](#)
[java.awt.dnd](#)

All Classes

[AbstractAction](#)
[AbstractAnnotationValueVisitor6](#)
[AbstractAnnotationValueVisitor7](#)
[AbstractAnnotationValueVisitor8](#)
[AbstractBorder](#)
[AbstractButton](#)
[AbstractCellEditor](#)
[AbstractChronology](#)
[AbstractCollection](#)
[AbstractColorChooserPanel](#)
[AbstractDocument](#)
[AbstractDocument.AttributeContext](#)
[AbstractDocument.Content](#)
[AbstractDocument.ElementEdit](#)
[AbstractElementVisitor6](#)
[AbstractElementVisitor7](#)
[AbstractElementVisitor8](#)
[AbstractExecutorService](#)
[AbstractInterruptibleChannel](#)
[AbstractLayoutCache](#)
[AbstractLayoutCache.NodeDimension](#)
[AbstractList](#)
[AbstractListModel](#)
[AbstractMap](#)
[AbstractMap.SimpleEntry](#)
[AbstractMap.SimpleImmutableEntry](#)

OVERVIEW

[PACKAGE](#) [CLASS](#) [USE](#) [TREE](#) [DEPRECATED](#) [INDEX](#) [HELP](#)[PREV](#) [NEXT](#)[FRAMES](#) [NO FRAMES](#)

Java™ Platform, Standard Edition 8

API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: [Description](#)

Profiles

- [compact1](#)
- [compact2](#)
- [compact3](#)

Packages

Package	Description
java.applet	Provides the classes necessary to create applets and uses to communicate with its applet container.
java.awt	Contains all of the classes for creating windows and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between applications.
java.awt.dnd	Drag and Drop is a direct manipulation interface system that provides a mechanism for associating two entities logically associated with each other.


```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.printf("%d 숫자를 나타냄\n", 1);  
4         System.out.printf("%s 을 표시\n", "문자열");  
5         System.out.printf("%3.2f 정수부분이 3자리, 소수점 아래가 2자리\n", 3.5f);  
6         System.out.printf("%2d 2자리의 10진수 숫자 숫자이외의 부분은 공백\n", 1);  
7         System.out.printf("%02d 2자리의 10진수 숫자 숫자이외의 부분은 0\n", 1);  
8         System.out.printf("가\t나\t다 : %s의 예제\n", "탭");  
9         System.out.printf("제 이름은 %s입니다. 나이는 %d살 키는 %3.1fcm 입니다", "오준석", 36, 173.3);  
10    }  
11 }
```

클래스의 분리

- 복수의 클래스로 1개의 프로그램을 구성하는 것이 가능
- 별도의 클래스의 메소드를 호출할 때는 **클래스명.메소드명** 으로 호출
- **Java** 프로그램의 이상적인 완성은 복수의 클래스 파일의 집합체이다.

패키지

- **package** 문을 이용하여 클래스를 패키지에 속하게 할 수 있다
- **import** 문을 사용하여 코드 안에 **full package** 지정을 생략 할 수 있다

API

- **Java** 에 이미 준비되어 있는 다수의 클래스군을 **API**라고 한다
- **API**는 보통 **java.** 이나 **javax.** 으로 시작하는 패키지명을 이용하고 있다
- **java.lang** 패키지에 소속하는 클래스는 자동적으로 **import** 된다
- **API**로 준비되어 있는 클래스는 **API** 레퍼런스에서 찾아 볼수 있다.



연습문제 6-1

```
1 public class Main {  
2     public static void main(String[] args) throws Exception {  
3         System.out.println("3초간 기다림!");  
4  
5         // 3초간 기다림  
6  
7         System.out.println("끝");  
8     }  
9 }
```

java.lang.Thread 클래스를 조사하여, 프로그램을 3초간 멈추게 하는 프로그램을 완성하시오. throws Exception 에 대해서는 나중에 배우니 일단 무시하시오.

연습문제 6-2

구구단을 작성하시오

2 * 1 = 2	3 * 1 = 3	4 * 1 = 4	5 * 1 = 5	6 * 1 = 6	7 * 1 = 7	8 * 1 = 8	9 * 1 = 9
2 * 2 = 4	3 * 2 = 6	4 * 2 = 8	5 * 2 = 10	6 * 2 = 12	7 * 2 = 14	8 * 2 = 16	9 * 2 = 18
2 * 3 = 6	3 * 3 = 9	4 * 3 = 12	5 * 3 = 15	6 * 3 = 18	7 * 3 = 21	8 * 3 = 24	9 * 3 = 27
2 * 4 = 8	3 * 4 = 12	4 * 4 = 16	5 * 4 = 20	6 * 4 = 24	7 * 4 = 28	8 * 4 = 32	9 * 4 = 36
2 * 5 = 10	3 * 5 = 15	4 * 5 = 20	5 * 5 = 25	6 * 5 = 30	7 * 5 = 35	8 * 5 = 40	9 * 5 = 45
2 * 6 = 12	3 * 6 = 18	4 * 6 = 24	5 * 6 = 30	6 * 6 = 36	7 * 6 = 42	8 * 6 = 48	9 * 6 = 54
2 * 7 = 14	3 * 7 = 21	4 * 7 = 28	5 * 7 = 35	6 * 7 = 42	7 * 7 = 49	8 * 7 = 56	9 * 7 = 63
2 * 8 = 16	3 * 8 = 24	4 * 8 = 32	5 * 8 = 40	6 * 8 = 48	7 * 8 = 56	8 * 8 = 64	9 * 8 = 72
2 * 9 = 18	3 * 9 = 27	4 * 9 = 36	5 * 9 = 45	6 * 9 = 54	7 * 9 = 63	8 * 9 = 72	9 * 9 = 81

연습문제 6-3

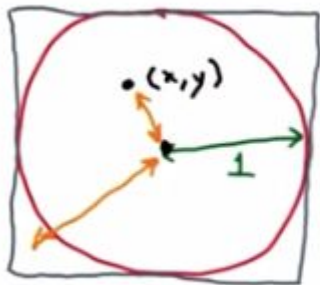
전자시계 프로그램을 작성하시오

```
1:00 2:00 3:00 4:00 5:00 6:00 7:00 8:00 9:00 10:00 11:00 12:00
1:01 2:01 3:01 4:01 5:01 6:01 7:01 8:01 9:01 10:01 11:01 12:01
1:02 2:02 3:02 4:02 5:02 6:02 7:02 8:02 9:02 10:02 11:02 12:02
1:03 2:03 3:03 4:03 5:03 6:03 7:03 8:03 9:03 10:03 11:03 12:03
1:04 2:04 3:04 4:04 5:04 6:04 7:04 8:04 9:04 10:04 11:04 12:04
1:05 2:05 3:05 4:05 5:05 6:05 7:05 8:05 9:05 10:05 11:05 12:05
...
1:59 2:59 3:59 4:59 5:59 6:59 7:59 8:59 9:59 10:59 11:59 12:59
```



연습문제 6-4

반지름이 1인 원 안에 다트를 던져서 원주율 구하기. 설명은 다음장에



참고 : <http://polymer.bu.edu/java/java/montepi/MontePi.html>

1. 던질 횟수를 입력해주세요 를 출력한다
2. 키보드로부터 long값을 변수 tries 에 입력 받는다
3. 정수형 hits 변수를 0으로 초기화 한다
4. 입력 받은 tries 의 수 만큼 for 문을 반복하며 아래 a, b 를 수행한다
 - a. 다트가 꽂히는 좌표 x, y 를 랜덤한 값으로 정하되 범위는 -1 ~ 1 사이의 실수(double) 로 한다
 - i. 힌트 : new Random().nextDouble() 는 0 ~ 1 사이의 실수를 랜덤하게 리턴 해 준다
 - b. 다트가 꽂힌 좌표가 원 안에 있을 경우 hits 를 증가연산자를 사용하여 1 증가 시킨다
 - i. 힌트 : 두 점 (x1, y1), (x2, y2) 의 거리는 $\text{Math.sqrt}((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))$ 로 구할 수 있음
5. 반복이 끝나면 실수형 변수 piEstimate 를 선언과 동시에 PI 값을 계산하여 대입하여 초기화 한다
 - a. 힌트 : hits / tries 의 비율은 원의 면적 / 사각형의 면적과 같고 이는 원주율(PI) / 4 와도 같다. 이 관계를 이용하여 PI를 구하면 됨
식은 이겁니다 : $\text{piEstimate} = 4 * \text{hits} / \text{tries}$
6. 마지막에 PI 값의 예상값 piEstimate 를 출력한다.
7. 3.141592.... 에 가까운 값이 나오는지 확인한다.