

Effective Java

Item 25. 톱레벨 클래스는 한 파일에 하나만 담으
라

목차

- 톱레벨 클래스란?
- 톱레벨 클래스를 한 파일에 여러 개 담는다면?
 - 배경지식) 컴파일러
 - 배경지식) 클래스 로더
- 문제 발생에 따른 해결책
- 요약 정리

톱 레벨 클래스란?

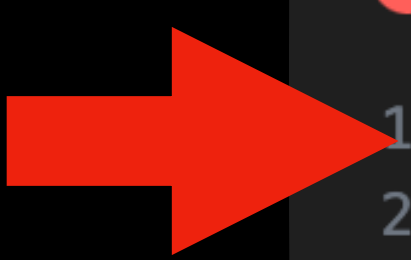
톱레벨 클래스란?

어떤 것이 톱 레벨 클래스일까?

```
1 public class OuterClass {
2     void method() {
3         Runnable r = new Runnable() {
4             @Override
5             public void run() {
6                 System.out.println("익명 클래스");
7             }
8         };
9     }
10 }
```

```
1 public class OuterClass {
2     void method() {
3         class LocalClass {
4             // 지역 클래스
5         }
6     }
7 }
```

```
1 public class OuterClass {
2     class InnerClass {
3         // 내부 클래스
4     }
5 }
```



```
1 public class TopLevelClass {
2     // 클래스 내용
3 }
```

톱레벨 클래스란?

톱레벨 클래스(Top-level class)는
다른 클래스 내부에 선언되지 않은 클래스를 의미한다.



```
1 public class TopLevelClass {  
    // 클래스 내용  
}
```

<톱레벨 클래스 특징>

1. **독립적인 클래스**로 선언 → 다른 클래스 내부에 정의 ❌
2. **public** 또는 **패키지 프라이빗**(기본 접근 제어자)로 선언 가능
 - public → 어디서든 접근 가능
 - 패키지 프라이빗 → 접근 제한자를 생략하면 같은 패키지에서만 접근 가능
3. **static** 키워드를 사용 불가 → static class는 내부 클래스에서만 가능

한 파일에 ^톱레벨 클래스가
여러개 존재한다면..?

톱레벨 클래스가 한 파일에 여러개 존재한다면 ..?

Main.java

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Utensil.NAME + Dessert.NAME);  
4     }  
5 }
```

pan cake

1 pancake

Utensil.java

```
1 class Utensil {  
2     static final String NAME = "pan";  
3 }  
4  
5 class Dessert {  
6     static final String NAME = "cake";  
7 }  
8
```

톱레벨 클래스가 한 파일에 여러개 존재한다면 ..?



```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Utensil.NAME + Dessert.NAME);  
4     }  
5 }
```



```
1 ????
```



Utensil.java

```
1 class Utensil {  
2     static final String NAME = "pan";  
3 }  
4  
5 class Dessert {  
6     static final String NAME = "cake";  
7 }
```

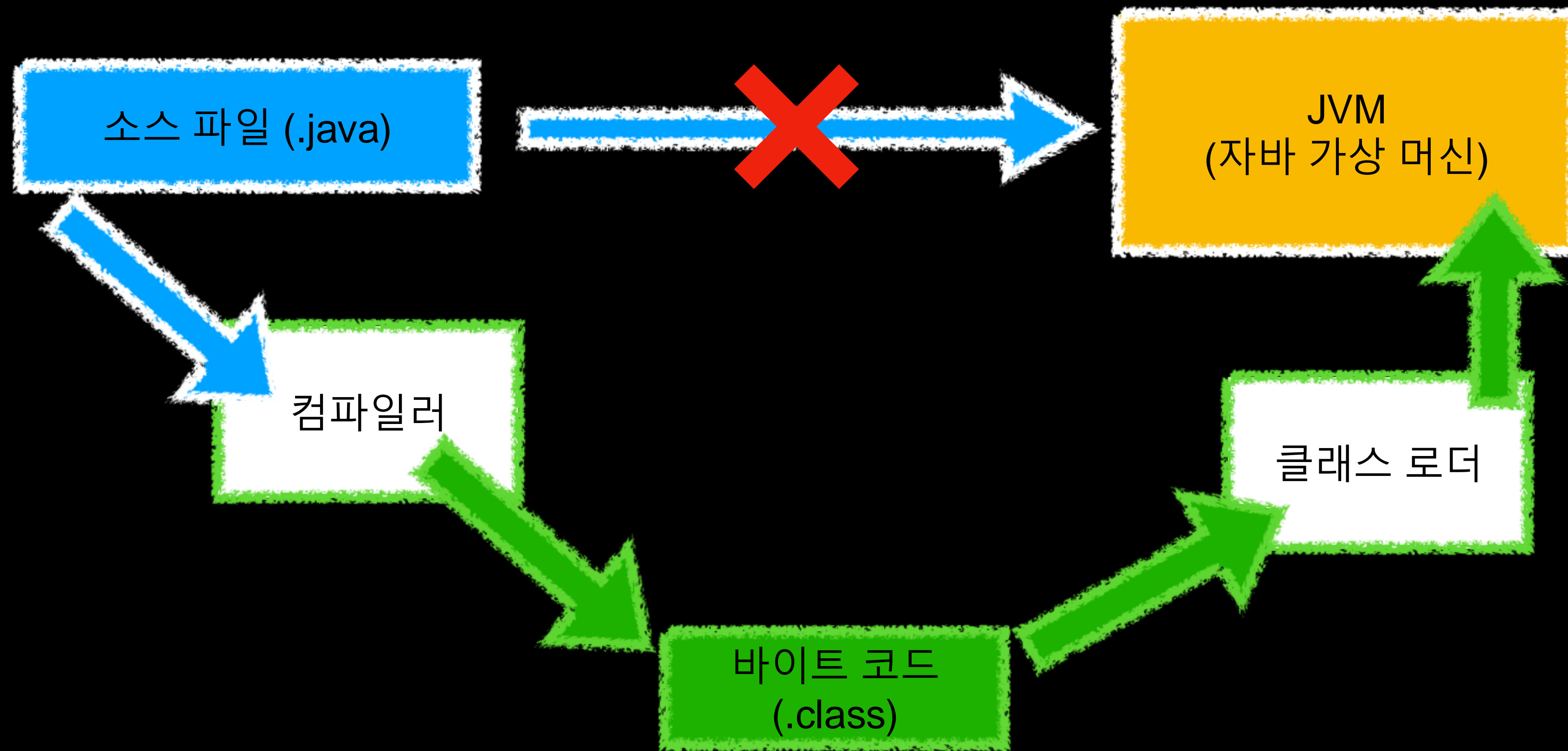


Dessert.java

```
1 class Utensil {  
2     static final String NAME = "pot";  
3 }  
4  
5 class Dessert {  
6     static final String NAME = "pie";  
7 }
```


토피레벨 클래스가 한 파일에 여러개 존재한다면

?
배경 지식) 소스 파일의 실행 순서



토피레벨 클래스가 한 파일에 여러개 존재한다면 배경지식) 컴파일러

자바의 컴파일러는 각각의 토피레벨 클래스를 개별적인 .class 파일로 변환한다.



```
1 // File: MultipleClasses.java
2 class A {
3     static final String MESSAGE = "Hello from A!";
4 }
5
6 class B {
7     static final String MESSAGE = "Hello from B!";
8 }
```

```
ui-MacBookPro item25 % cd example
ui-MacBookPro example % javac MultipleClasses.java
ui-MacBookPro example %
```

javac: 소스 코드를 컴파일하는 명령어

example

- J A.class 클래스 A의 바이트 코드
- J B.class 클래스 B의 바이트 코드
- J MultipleClasses.java 원본 코드

토퍼벨 클래스가 한 파일에 여러개 존재한다면 배경지식) 클래스 로더

클래스 로더(Class Loader)는 **클래스 파일(.class)**을 읽어와 **JVM에 로드**한다.

클래스 로더는 **필요할 때 클래스 파일을 로드**하며, 요청이 있을 때만 개별적으로 로드한다.

클래스 로더의 동작 방식

1. 클래스가 처음 사용될 때 클래스 로더가 .class 파일을 찾는다.
2. JVM이 해당 클래스를 메모리에 로드하여 사용 가능하게 만든다.
3. 이미 로드된 클래스는 재사용됩니다.

톱레벨 클래스가 한 파일에 여러개 존재한다면 ..?

- razel@lajel-ui-MacBookPro vsCode % `javac Main.java Utensil.java`

- razel@lajel-ui-MacBookPro vsCode % `java Main.java`

pancake

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println(Utensil.NAME + Dessert.NAME);
4     }
5 }
```

- razel@lajel-ui-MacBookPro vsCode % `javac Main.java`

- razel@lajel-ui-MacBookPro vsCode % `java Main.java`

pancake

Utensil.java

```
1 class Utensil {
2     static final String NAME = "pan";
3 }
4
5 class Desert {
6     static final String NAME = "cake";
7 }
```

Dessert.java

```
1 class Utensil {
2     static final String NAME = "pot";
3 }
4
5 class Dessert {
6     static final String NAME = "pie";
7 }
```

토퍼벨 클래스가 한 파일에 여러개 존재한다면 ..?

```
● razel@lajel-ui-MacBookPro vscode % javac Dessert.java Main.java
● razel@lajel-ui-MacBookPro vscode % java Main.java
potpie
```

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println(Utensil.NAME + Dessert.NAME);
4     }
5 }
```

Utensil.java

```
1 class Utensil {
2     static final String NAME = "pan";
3 }
4
5 class Desert {
6     static final String NAME = "cake";
7 }
```

Dessert.java

```
1 class Utensil {
2     static final String NAME = "pot";
3 }
4
5 class Dessert {
6     static final String NAME = "pie";
7 }
```

톱레벨 클래스가 한 파일에 여러개 존재한다면 ..?

● razel@lajel-ui-MacBookPro item25 % `javac Main.java Dessert.java`



```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Utensil.NAME + Dessert.NAME);  
4     }  
5 }
```

Utensil.java:3: error: **duplicate class:** item25.Utensil
class Utensil {
^
Utensil.java:7: error: **duplicate class:** item25.Dessert
class Dessert {
^
2 errors

컴파일오류!!
클래스 중복 정의



Utensil.java

```
1 class Utensil { 클래스 정의 완료  
2     static final String NAME = "pan";  
3 }  
4  
5 class Desert { 클래스 정의 완료  
6     static final String NAME = "cake";  
7 }
```



Dessert.java

```
1 class Utensil { 이미 정의됨  
2     static final String NAME = "pot";  
3 }  
4  
5 class Dessert { 이미 정의됨  
6     static final String NAME = "pie";  
7 }
```

토퍼벨 클래스가 한 파일에 여러개 존재한다면

??

- 컴파일러에 어느 소스 파일을 먼저 건네느냐에 따라 동작이 달라진다.

→ 코드를 수정하지 않았는데 실행 결과가 달라지는 것은 문제가 된다.

- 컴파일된 .class 파일과 원본 .java 파일 간의 대응 관계가 불명확 해진다.

→ 클래스를 수정하고자 할 때, 어떤 소스 파일을 수정해야하는지 찾기 어려워진다.

- 컴파일된 .class 파일들이 서로 다른 시점에 생성된 경우, 버전 불일치로 오류가 발생할 수 있다.

→ 한 클래스만 수정하게 된다면, 수정한 클래스는 덮어 써지고, 기존 클래스는 유지가 되면서 버전의 차이가 생긴다.

문제 발생에 따른 해결책

문제 발생에 따른 해결책

- 톱레벨 클래스들을 서로 다른 소스 파일로 분리한다.
- 다른 클래스에 딸린 부차적인 클래스라면 정적 멤버 클래스로 만드는 쪽이 일반적으로 더 낫다.

```
1 public class Test {
2     public static void main(String[] args) {
3         System.out.println(Utensil.NAME + Dessert.NAME);
4     }
5
6     private static class Utensil {
7         static final String NAME = "pan";
8     }
9     private static class Dessert {
10        static final String NAME = "cake";
11    }
12 }
```

→ 가독성이 좋고, private으로 선언
하면
접근 범위도 치수로 관리 가능하다!

요약 정리

요약 정리

- 톱 레벨 클래스는 다른 클래스 내부에 정의되지 않는 독립적인 클래스이다.
- 한 소스 파일 안에 여러 개의 톱레벨 클래스가 존재한다면, 컴파일러와 클래스 로더의 동작 방식으로 인해 프로그램 실행 결과가 달라질 수 있다.



- 톱 레벨 클래스는 한 파일에 하나만 담아야 한다.
- 한 파일에 여러 클래스를 사용할 경우, 정적 멤버 클래스로 만드는 것이 일반적으로 더 좋다.

- The End -