

5. Collections

Object-Oriented Design & Programming
2016년도 봄학기

시작하기 전에

- 어떤 과목이든,
'단어(keyword)'를 정확히 사용하는 것은
공부할 때나 시험 볼 때 몹시 중요합니다.
 - 이름에는, 그 이름을 붙인 사람의 의도가 들어 있음
 - 여러분의 의도를 표현하는 가장 좋은 방법은,
그 의도가 함축된 단어를 사용하는 것임
 - 일상적인 대화에서도 그렇겠지만,
공대 공부에서는 특히나 더 중요함

시작하기 전에

- 그러므로, 앞으로 모든 실습자료에서는 Java의 특정 개념을 지칭하는 단어들은 모두 '원래 명칭'을 써서 적으려 합니다.
 - 인스턴스 → instance
 - 필드 → field
 - 메서드 → method
 - 기타 등등
 - 다만 '클래스'나 '생성자'같이 Java / C++ / C# 등에서 공통적으로 사용되는 OOP 개념은 종종 한글로 적을 수 있음

시작하기 전에

- 이렇게 적으면 또
Java code에서 붙인 이름과
Java 자체의 keyword가 헷갈릴 수 있으니
Java code의 이름을 적을 때는
항상 '서로 다른 글자체'로 적으려 합니다.
 - 새로운 클래스와 Java class와 Program
 - static field와 numberOfHeads
 - public method와 main()

시작하기 전에

- 물론, 조교가 이렇게 한다고 해서 여러분도 이렇게 해야 하는 것은 아닙니다.
 - 사실 정말 귀찮은(까먹기 쉬운) 작업임
- 그럼에도 불구하고
우리는 지금 처음 배우는 입장이니
각종 기본 개념들, 기본 keyword들을
차근차근 정확히 익혀 보도록 합시다.

이번 시간에는

- 지난 시간에 사용해 본 개념들 돌아보기
- Collections
 - Java에서 기본으로 제공하는 collection class들
 - 가장 자주 쓰이는 ArrayList<> class
 - 실습과제#4 진행
- 전반기 개인프로젝트 시작

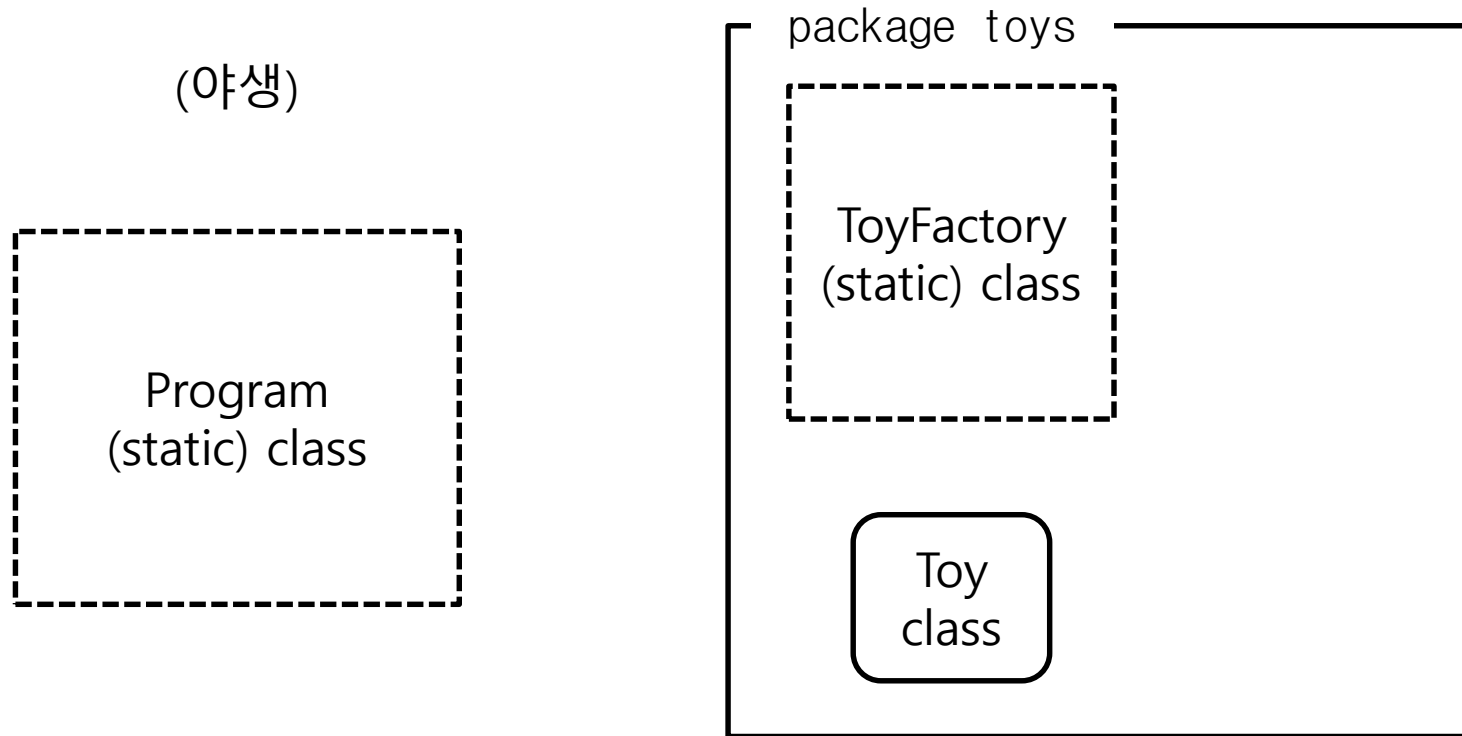
실습과제#4

- 아이템, 인벤토리, 그리고 보물상자
 - Item class 만들기(지난 주 복습) – 64점
 - Inventory class 만들기 – 32점
 - Chest class와 ItemMaker class 만들기 – 4점
- 이번 실습과제#4-3은
상상력과 충분한 복습을 필요로 하며,
이번 주에 도전해 보기 딱 좋은 부분과제입니다.

지난 주 내용 돌아보기

- HY-in 학생자료실에서
OODP_HW_3_3_by_TA.zip을 받아서
import해 봅시다.

돌아본 것 정리

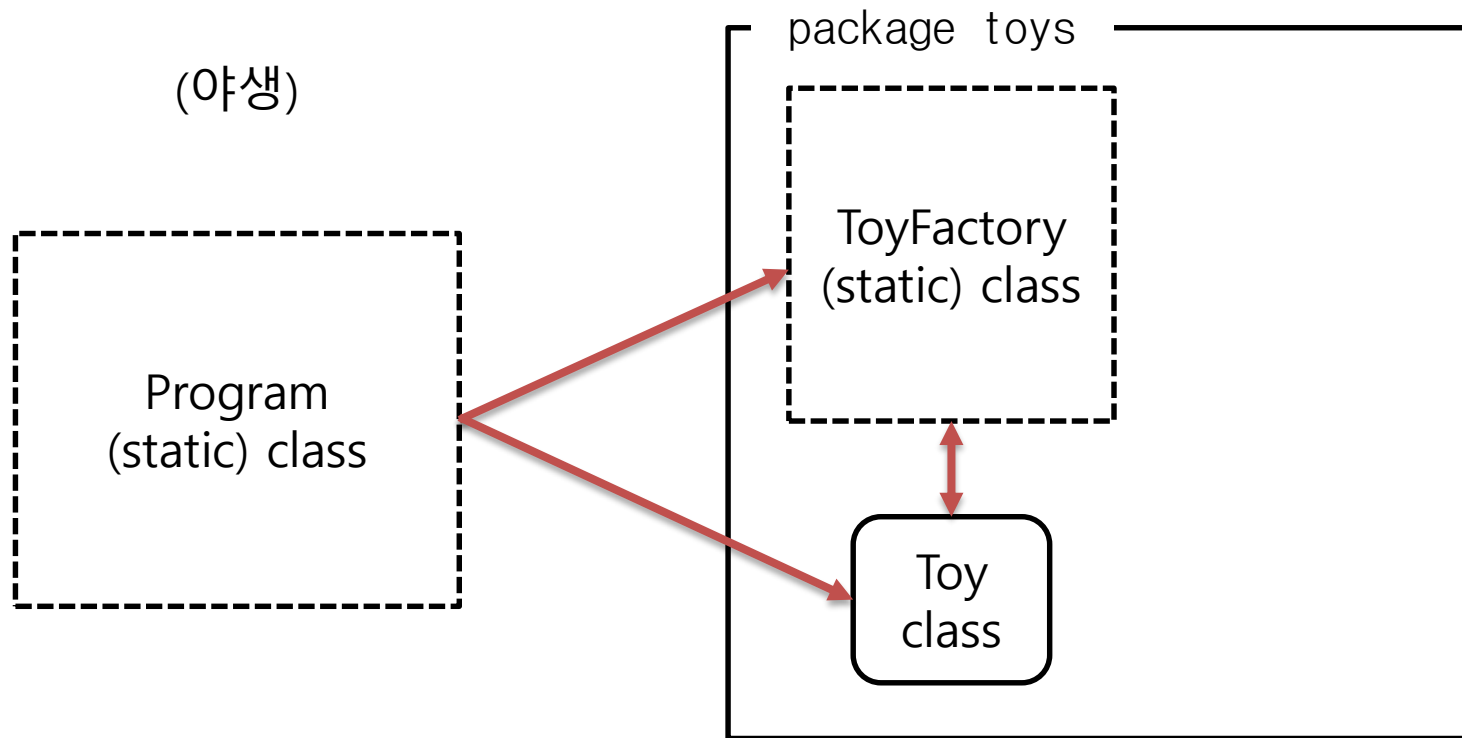


돌아본 것 정리

- import 키워드의 진짜 의미
 - 원래 class는,
중세 시대에 '어느 마을의 누구'로 사람을 부르듯
`java.util.Scanner`, `toys.ToyFactory`처럼
각자의 '풀 네임'을 불러 주는 것이 원칙임
 - 하지만 import 키워드를 쓰면,
자주 사용할 이름들에 대해
'그냥 그 이름'만으로 부를 수 있게 해 줌
 - 여기서 단 한 가지 예외로,
`java.lang` package는 항상 import된 것으로 간주함

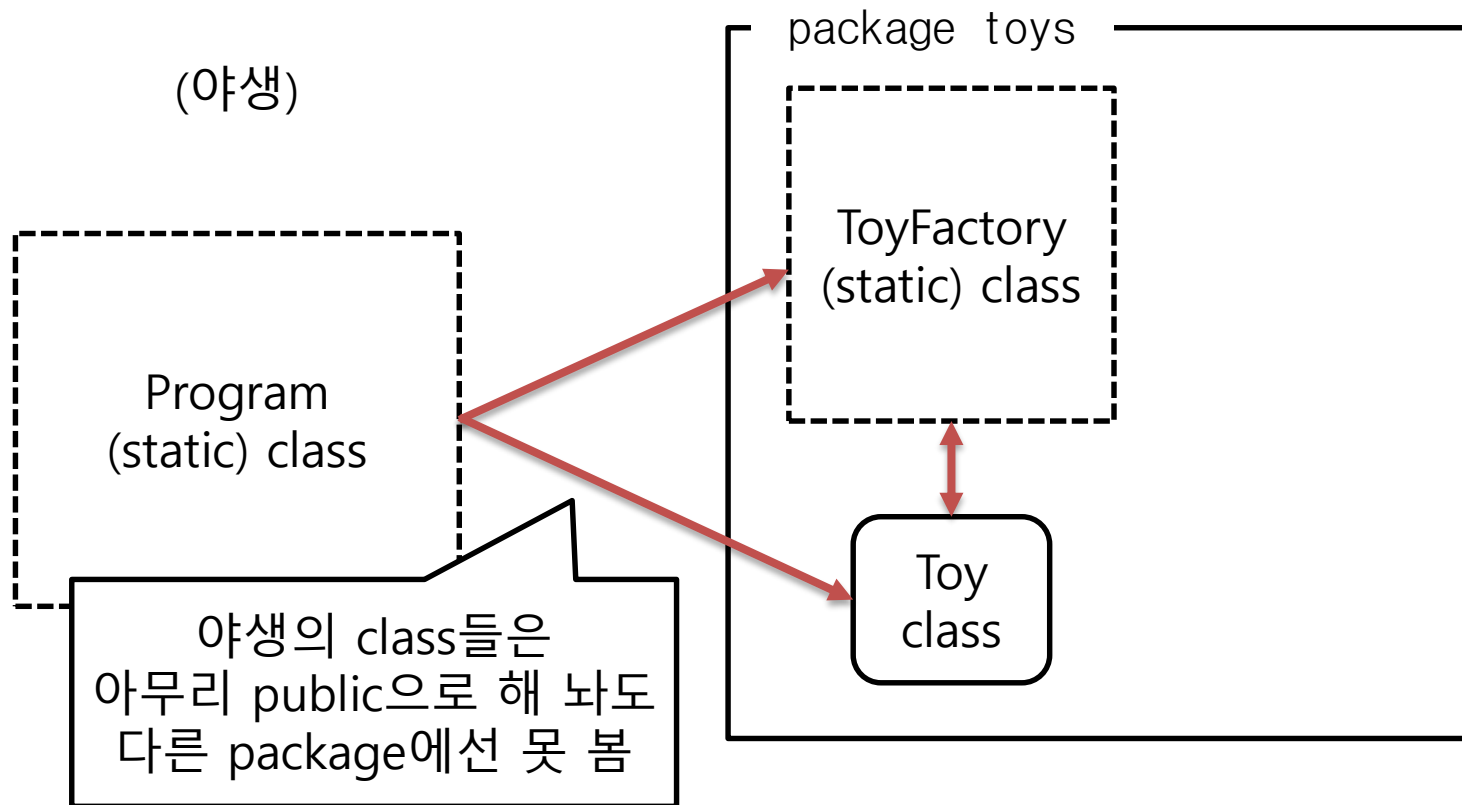
돌아본 것 정리

- 1. 저 class를 볼 수 있나?



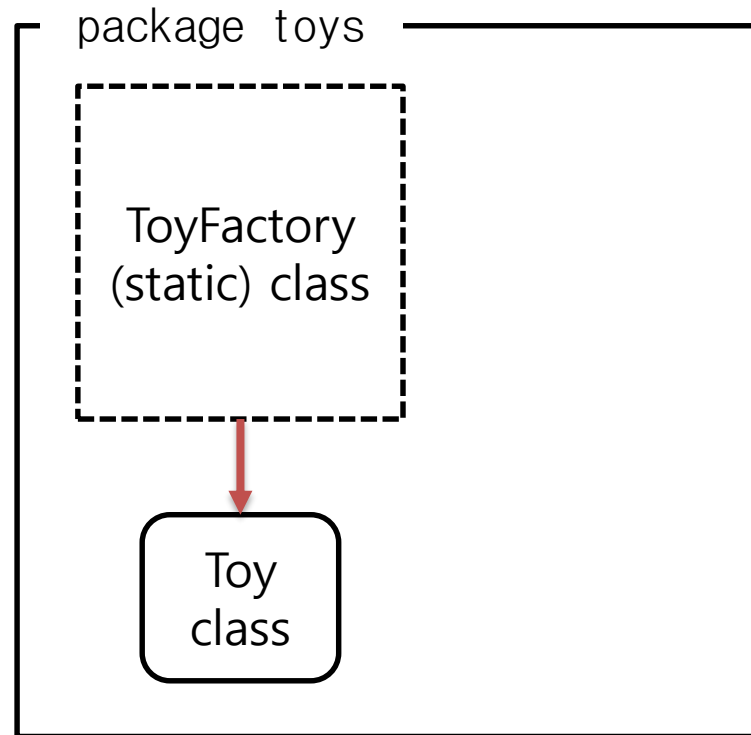
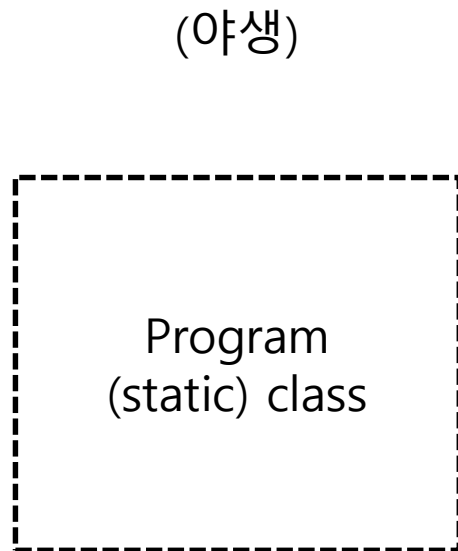
돌아본 것 정리

- 1. 저 class를 볼 수 있나?



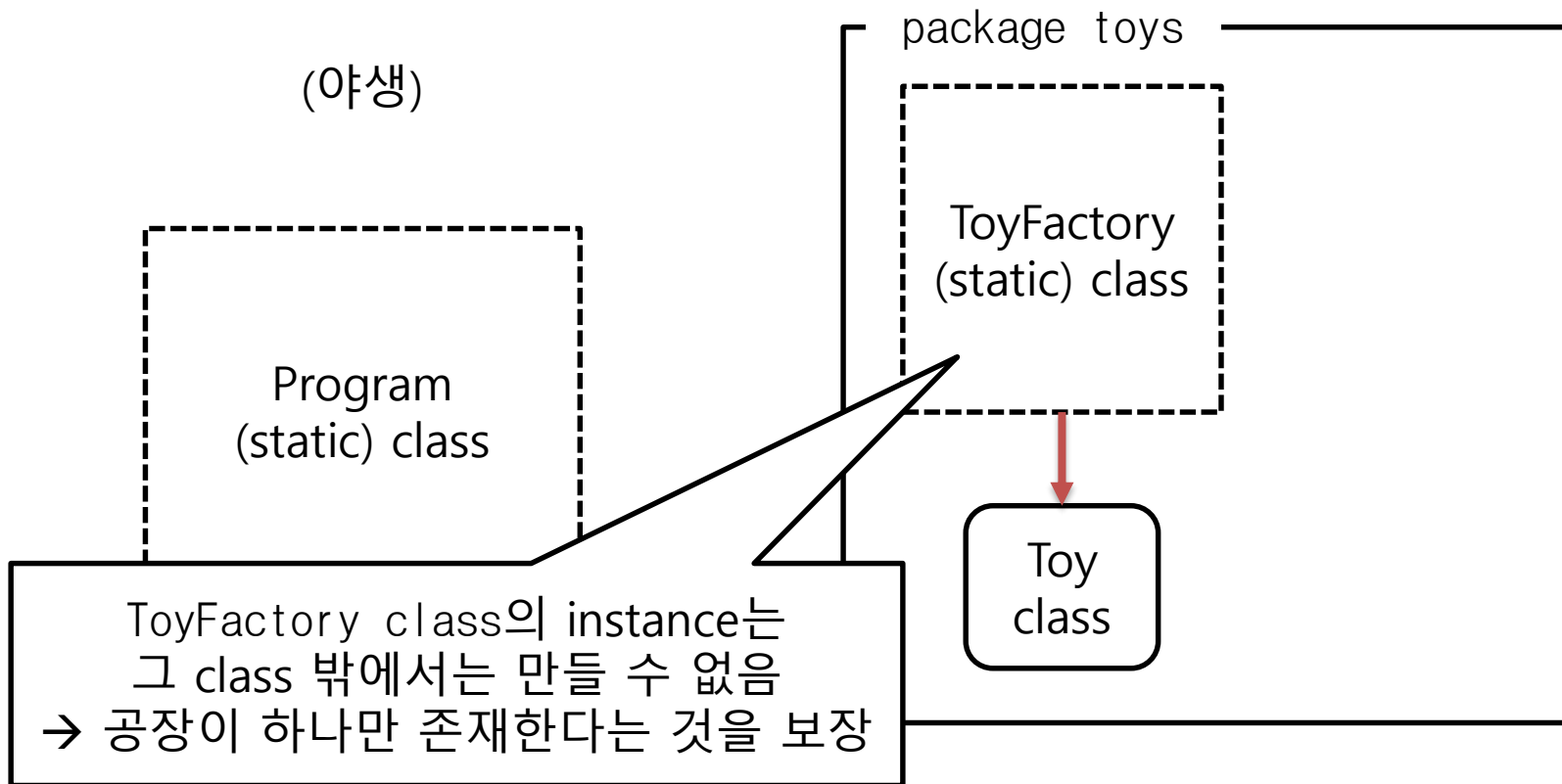
돌아본 것 정리

- 2. 저 class를 instance화할 수 있나?



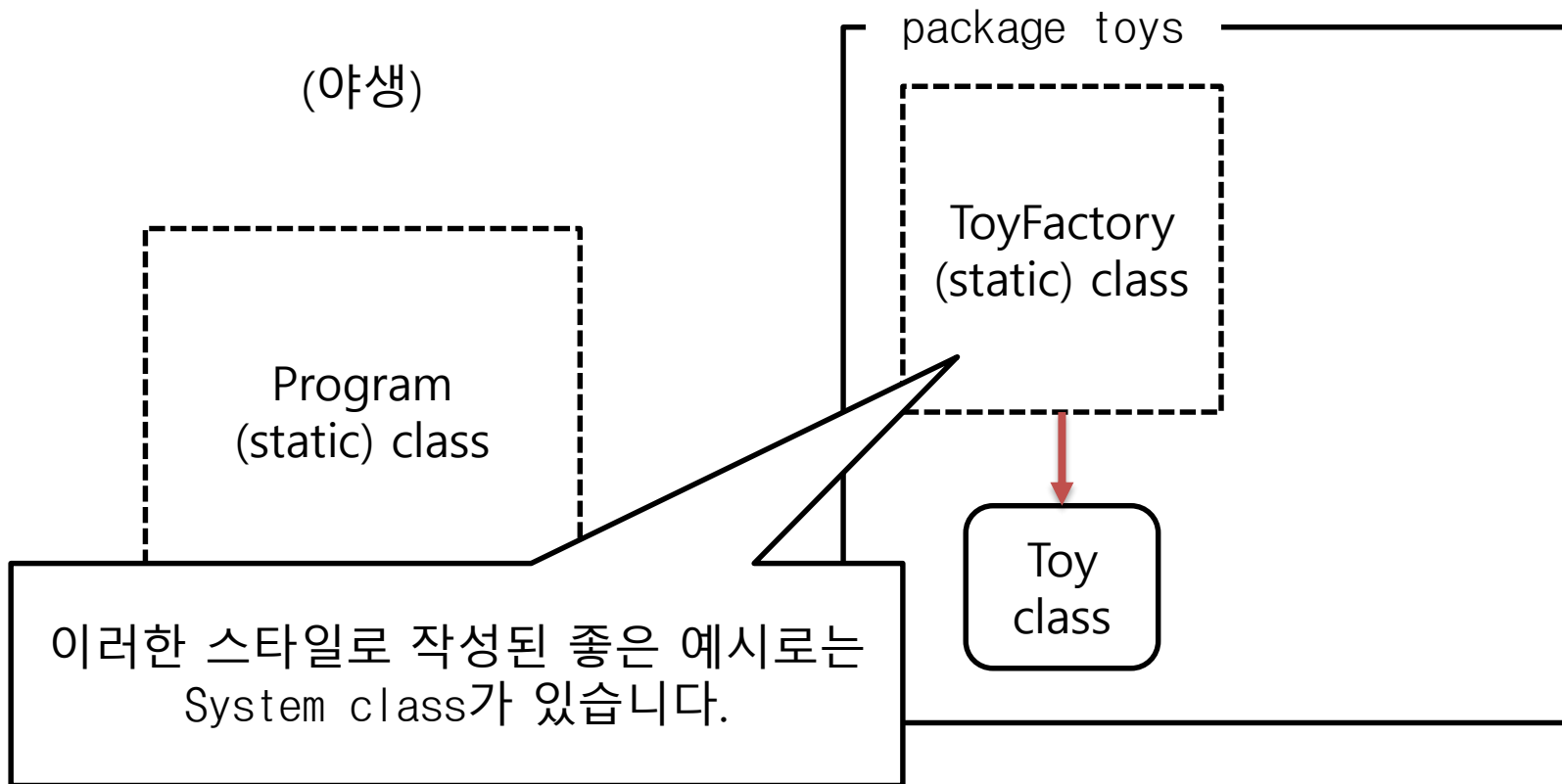
돌아본 것 정리

- 2. 저 class를 instance화할 수 있나?



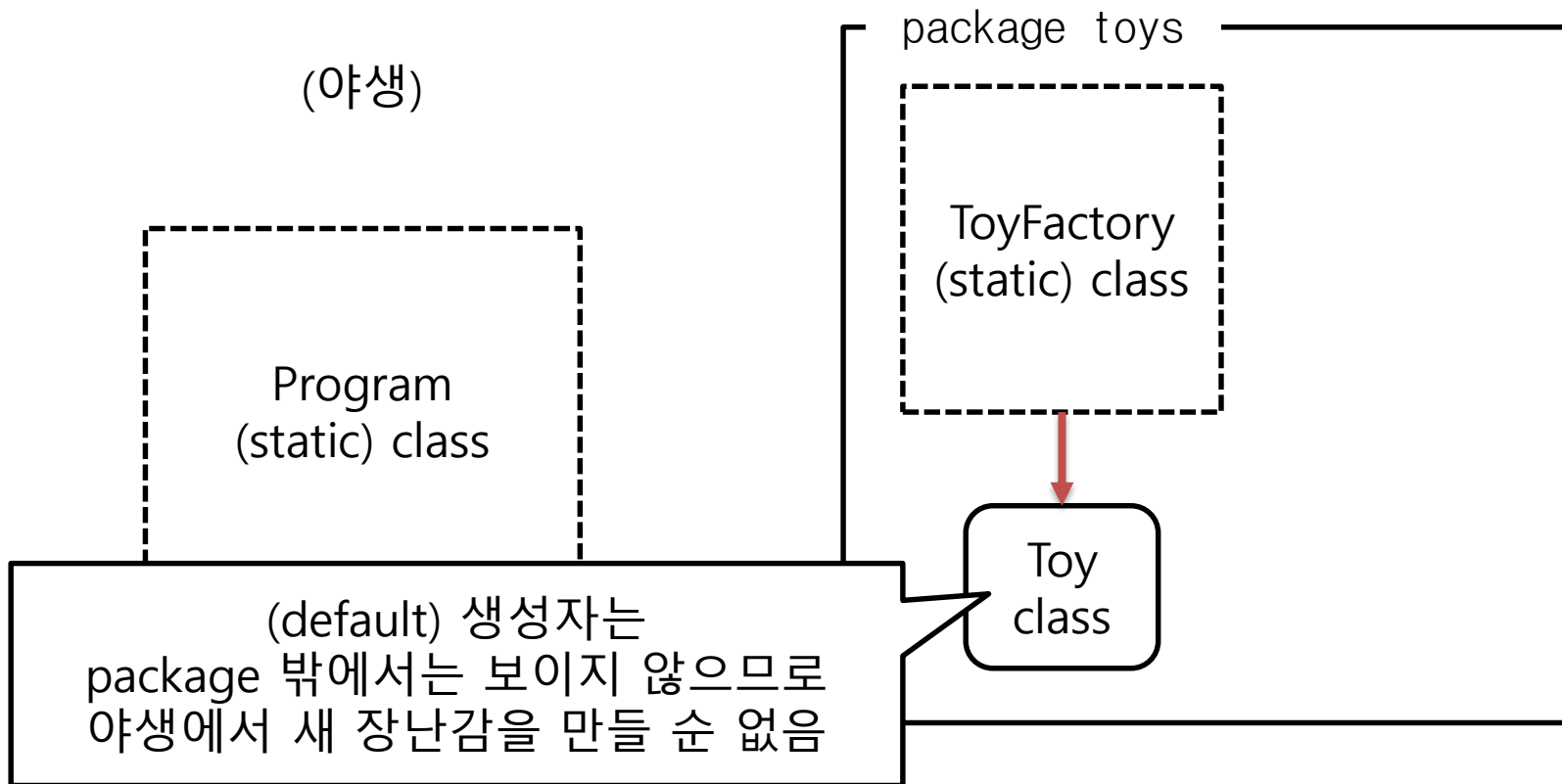
돌아본 것 정리

- 2. 저 class를 instance화할 수 있나?



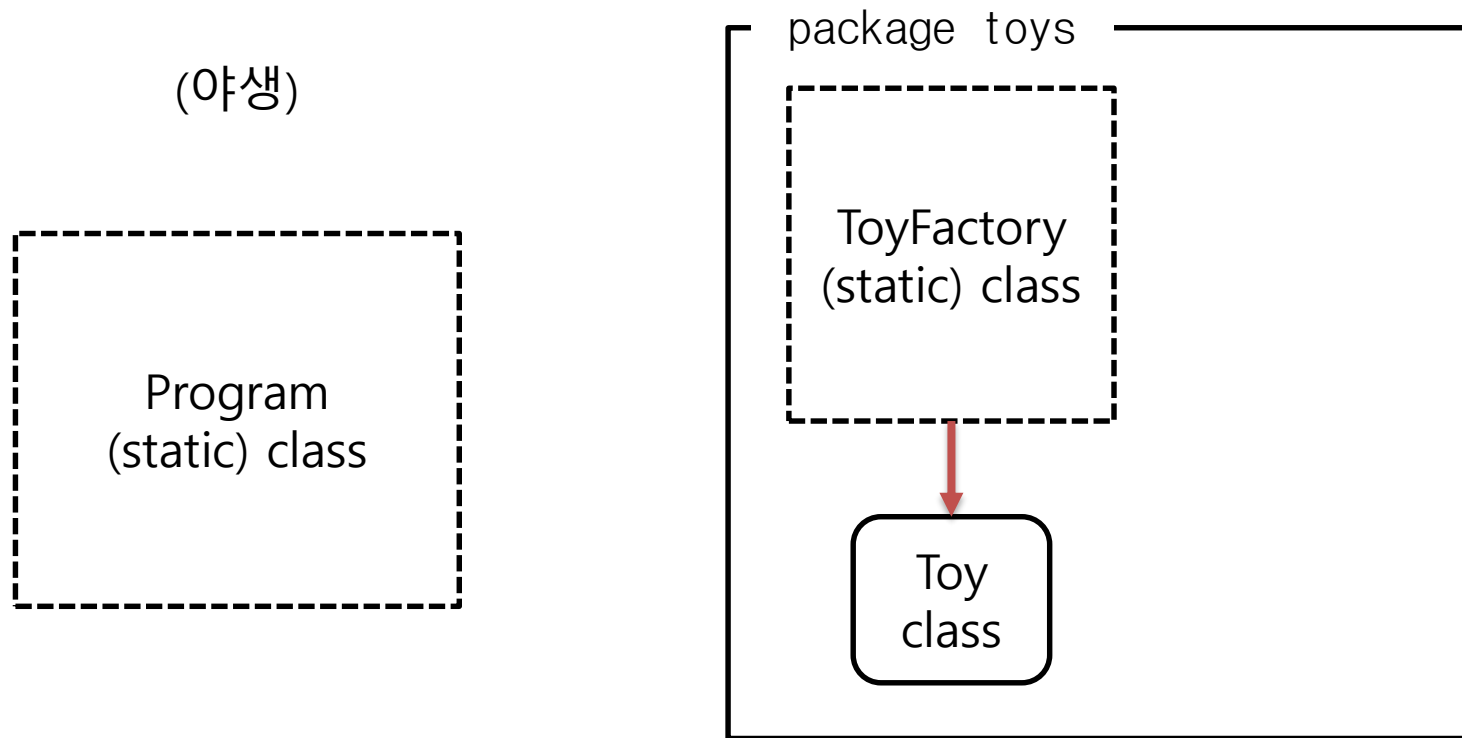
돌아본 것 정리

- 2. 저 class를 instance화할 수 있나?



돌아본 것 정리

- 2. 저 class를 instance화할 수 있나?



돌아본 것 정리

- 3. 저 class의 instance를 사용할 수 있나?
 - 그 class의 field, method를 어떻게 설정했는지에 따라 각자 다 다름

돌아본 것 정리

- '어디서 볼 수 있나'를 정해 주는 키워드들

돌아본 것 정리

- '어디서 볼 수 있나'를 정해 주는 키워드들
 - public: 누구나 볼 수 있음
 - 다만, 야생의 class는 다른 package에서는 볼 수 없음
 - private: '이 class 중괄호 안'에서만 볼 수 있음
 - (default): '이 package 폴더 안'에서만 볼 수 있음
 - 기본적으로, 볼 수 없다면 사용할 수 없음!

돌아본 것 정리

- 이 키워드들을 붙일 수 있는 곳들
 - Class 자체
 - 이 class가 어디까지 보일 수 있는지 결정함
 - 당연하겠지만, private class는 허용되지 않음
 - Class의 생성자
 - 이 class의 instance를 어디서 만들 수 있는지 결정함
 - 아무 생성자도 없을 때는 Java가 암시적으로 만들어 주며, 이 때 만들어진 생성자는 public임

돌아본 것 정리

- 이 키워드들을 붙일 수 있는 곳들
 - Class의 각 field들, method들
 - 해당 field들, method들을 어디서 사용할 수 있는지 결정함
 - field의 값을 읽거나 바꿀 수 있는지
 - method를 호출할 수 있는지
 - 당연히, field의 type, method의 argument 및 return type은 '적어도 그 field 또는 method만큼은' 보일 수 있어야 함
 - (default) class type의 field를 만들 때...
 - » public은 불가 → 밖에서는 이 class의 존재 자체를 모름
 - » (default)나 private은 가능

돌아본 것 정리

- static: '누구에게 귀속되나'를 정해 줌

돌아본 것 정리

- static: '누구에게 귀속되나'를 정해 줌
 - static을 붙이면...
 - 해당 field / method는 해당 **class**에 귀속됨
 - static field: 해당 class에 단 하나 존재하는 field
 - » 각 instance들이 '공유'하는 field라 볼 수도 있음
 - static method: Instance와 무관하게 호출되는 method
 - » 따라서 static이 아닌 field를 static method에서 볼 순 없음
 - 사용할 때는, class 이름에 점을 찍으면 됨
 - ex) System.out, ToyFactory.MakeOne() 등

돌아본 것 정리

- static: '누구에게 귀속되나'를 정해 줌
 - static을 안 붙이면...
 - 해당 field / method는 각 **instance**에 귀속됨
 - non-static field: 각 instance마다 존재하는 field
 - » 서로 다른 instance들은 서로 다른 field 값을 가질 수 있음
 - non-static method: 각 instance'에 대해' 호출되는 method
 - » 해당 instance의 non-static field도 사용 가능
 - 사용할 때는, instance 이름에 점을 찍으면 됨
 - ex) `System.out.println()`, `toy.Fix()` 등

돌아본 것 정리

- Java package: '장소'를 나누는 방법
- import: '장소'를 생략하고 부르는 방법
- '어디서 볼 수 있나'를 정해 주는 키워드들
- '누구에게 귀속되나'를 정해 주는 키워드

돌아본 것 정리

- 이 많은 것들을
지금 완벽히 이해할 필요는 없습니다.
 - 앞으로 꾸준히 써 볼 예정이니,
어 까먹었다 싶으면 다시 이 슬라이드를 보면 됨
- 그러니 너무 걱정하지는 말고,
잠시 한 숨 돌린 다음
오늘 실습을 시작해 봅시다.

Collections

- 오늘 배울 새 class 스타일은,
다른 class의 instance들을 담기 위한
collection class입니다.

Collections

- 오늘 배울 새 class 스타일은, 다른 class의 instance들을 담기 위한 collection class입니다.
 - 네, Java에선 배열도 collection class의 한 종류입니다.
 - Java에는, 배열 이외에도, 미리 만들어진 다양한 collection class들이 있습니다.

Collections

- Java에 들어 있는 기본 collection들:
 - `java.util.ArrayList<>` class
 - `java.util.LinkedList<>` class
 - `java.util.TreeSet<>` class
 - `java.util.HashMap<>` class
 - 그 외 다수

Collections

- Java에 들어 있는 기본 collection들:
 - `java.util.ArrayList<>` class
 - `java.util.LinkedList<>` class
 - `java.util.TreeSet<>` class
 - `java.util.HashMap<>` class
 - 그 외 다수



배열이랑 유사합니다.

Collections

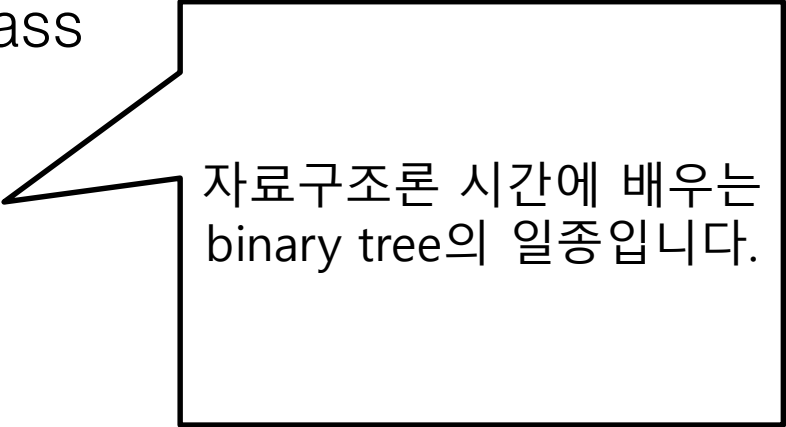
- Java에 들어 있는 기본 collection들:
 - `java.util.ArrayList<>` class
 - `java.util.LinkedList<>` class
 - `java.util.TreeSet<>` class
 - `java.util.HashMap<>` class
 - 그 외 다수



자료구조론 시간에 배운
linked list입니다.

Collections

- Java에 들어 있는 기본 collection들:
 - `java.util.ArrayList<>` class
 - `java.util.LinkedList<>` class
 - `java.util.TreeSet<>` class
 - `java.util.HashMap<>` class
 - 그 외 다수



자료구조론 시간에 배우는
binary tree의 일종입니다.

Collections

- Java에 들어 있는 기본 collection들:
 - `java.util.ArrayList<>` class
 - `java.util.LinkedList<>` class
 - `java.util.TreeSet<>` class
 - `java.util.HashMap<>` class
 - 그 외 다수

알고리즘 시간에 배울
hash table의 일종입니다.

Collections

- Java에 들어 있는 기본 collection들:

- `java.util.ArrayList<>` class

- `java.util.LinkedList<>` class

- `java.util.Vector<>` class

이들 모두는 각자 장/단점이 있습니다.
(주로 '어떨 때 더 빠르지'가 다릅니다)

물론 우리가 그걸 이해하고 쓰는 것은 불가능하며,
그 장/단점 또한, data가 몇 만 개 정도는 되어야 눈에 보이는 차이가 생기므로,
보통 Java 프로그래머들은 그냥 배열 또는 `ArrayList<>`중 하나를 골라 씁니다.

Collections

- `java.util.ArrayList<>` class
 - 구조는 배열과 유사합니다(사실 같습니다).
- 차이점은...
 - 배열의 길이는 생성 순간 고정되지만, 이 친구는 길이가 가변적입니다.
 - 따라서 보통은, 어떤 data의 수가 고정되어 있으면 배열을, 그렇지 않으면 `ArrayList<>`를 쓴다 생각하면 됩니다.

Collections

- 배열 사용 예:

```
int[] arr = new int[10];  
  
for ( int idx = 0; idx < 10; ++idx )  
    arr[idx] = idx;  
  
for ( int number : arr )  
    System.out.println(number);
```

Collections

- ArrayList<> 사용 예:

```
ArrayList<Integer> arr = new ArrayList<>();  
  
for ( int idx = 0; idx < 10; ++idx )  
    arr.add(idx);  
  
for ( int number : arr )  
    System.out.println(number);
```

Collections

- ArrayList<> 사용 예:

```
ArrayList<Integer> arr = new ArrayList<>();
```

```
for (int idx = 0; idx < 10; ++idx) {
```

Class 이름 옆에 붙는 <>는 '어떤 type에 대한'을 의미합니다.
여기서는 '어떤 type을 담는 collection인지'를 정하기 위해 쓰입니다.

그리고, Java에서 <> 안에는 반드시 class 이름을 넣어야 합니다.
int 값을 담고 싶을 땐 int에 해당하는 class인 Integer를 넣어야 합니다.
아마 여러분은 Toy class같은 data element class들을 만들어 쓸 테니
이 문법 때문에 크게 불편을 겪게 되진 않을 것입니다.

Collections

- ArrayList<> 사용 예:

```
ArrayList<Integer> arr = new ArrayList<>();  
  
for ( int idx = 0; idx < 10; ++idx )  
    arr.add(idx);
```

add()는 현재 collection의 맨 뒤에 새 instance를 추가합니다.
이외에도 다양한 추가 method들이 존재하지만,
보통은 add()를 가장 많이 사용하게 됩니다.

Collections

- ArrayList<> 사용 예:

foreach문을 쓰면
배열이든 뭐든 동일한 방법으로 사용할 수 있습니다.
이 Java code는, 우리가 기대하는 대로,
arr에 있는 모든 instance들을 순서대로 출력해 줍니다.

```
for ( int number : arr )  
    System.out.println(number);
```

Collections

- 이제 잠시 쉬었다가
전반기 개인 프로젝트를 시작해 보고,
실습과제#4도 수행해 보도록 합시다!

실습과제#4

- 아이템, 인벤토리, 그리고 보물상자
 - Item class 만들기(지난 주 복습) – 64점
 - Inventory class 만들기 – 32점
 - Chest class와 ItemMaker class 만들기 – 4점
- 이번 실습과제#4-3은
상상력과 충분한 복습을 필요로 하며,
이번 주에 도전해 보기 딱 좋은 부분과제입니다.