Default Method

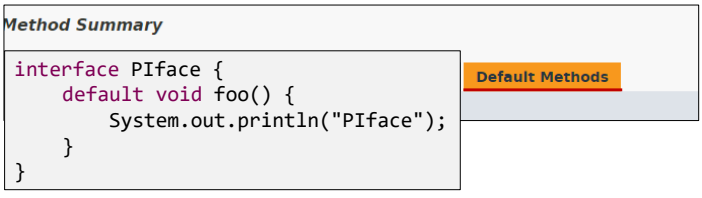
sanghyuck.na@lge.com

JAVA Initializer
JAVA for-loop
Collections helper method

1

Default method[8]

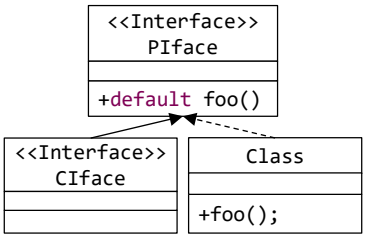
- interface Method
  - default 키워드로 default method 정의하고 실제 구현 추가
  - Legacy code와 호환성 보장
- 호환성을 위한 조치
  - default method는 상속됩니다. sub-interface는 abstract method로 재정의 가능
  - sub interface는 default method를 재정의 할 수 있습니다



2

Child class overriding

- Method 재정의
  - abstract, general method로 메소드 속성을 바꿀 수 있습니다.
  - Parent.super.methodName() 부모 default method호출



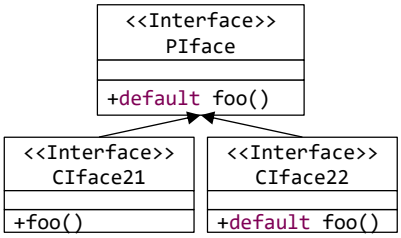

```
interface CIface extends PIface {
}

class Class implements Piface {
    public void foo() {
        Piface.super.foo();
        System.out.println("Class");
}}
```

3

Child interface overriding

- Method 재정의
  - 재정의동작은 상위 class로 인한 의존성 없음
  - interface CIface21는 abstract method 재정의
  - interface CIface22는 default method 재정의

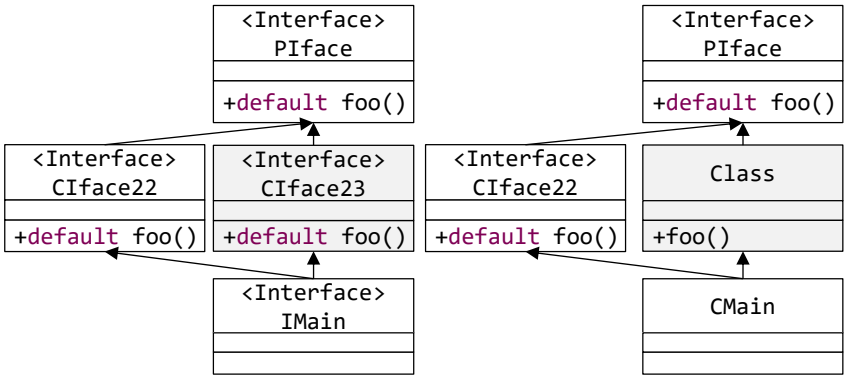
```
interface CIface21 extends PIface {
    void foo();
}
interface CIface22 extends PIface {
    default void foo() { System.out.println("C22"); }
}
```



4

Confliction, Priority

- 충돌 재정의 제약
  - 양쪽이 동일한 class type이라면 반드시 default method를 재정의 해야합니다
  - Duplicate default methods named foo with the parameters () and () are inherited from the types CIface23 and CIface22
- Class method 우선정책
  - interface default method 보다 class가 우선됩니다.
  - Class method가 하나라면 기본 선택. 만약 둘 다 존재한다면 재정의 필요



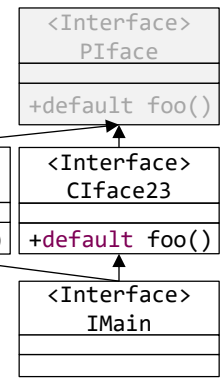
5

Close super call

- 인접 부모호출만 가능
  - 상속관계의 인접부모의 default metho만 호출 가능
  - Illegal reference to super type PIface, cannot bypass the more specific direct super type co.ajava.defaultmethod.CIface23

```
interface IMain extends CIface22, CIface23 {
    default void foo() {
        PIface.super.foo();
    }}
```

```
class Class implements PIface {
    public void foo() {
        PIface.super.foo();
    }}
```



6

JAVA Initializer

- 생성과 초기화 히스토리
  - JAVA5: Generics
  - JAVA7: Double brace initialization
  - JAVA8: Stream Factory Methods
  - JAVA9: Collections Factory Methods

```
List lst;
lst = new ArrayList();

lst = new ArrayList<Integer>();
lst = new ArrayList<>();

lst = new ArrayList<>() {{ add(1);}};

lst = IntStream.range(0, 3)
    .collect(ArrayList::new, ArrayList::add,  ArrayList::addAll);
list = List.of(0, 1, 2);
```

https://goo.gl/BKnMUa foreach

7

JAVA for-loop

- For-Looper 히스토리
  - JAVA: for
  - JAVA 1.2: Iterator
  - JAVA5: Foreach
  - JAVA8: Stream

```
List<Integer> lst = List.of(0, 1, 2);

for (int i = 0; i < list.size(); i++) {list.get(i);}

for (Iterator iterator = list.iterator(); iterator.hasNext();) {
    Integer i = (Integer) iterator.next();}

for (Integer i : list) {}

list.steram().forEach(System.out::println);
```

http://bit.ly/2D8PVdX performance

8

JAVA for-loop

- JAVA8 forEach
  - `void forEach(Consumer<? super T> action)`
  - 모든 요소를 순회합니다. 단 순서는 보장 안 합니다
- JAVA8 forEachOrdered
  - `void forEachOrdered(Consumer<? super T> action)`
  - 자료구조에 정의된 순서(encounter order)에 따라 순회합니다.

```
interface Consumer {
   void accept(T t)
}
```

```
List lst = List.of(0, 1, 2);

list.steram().forEach(System.out::println);
list.steram().forEachOrdered(System.out::println);
```

https://goo.gl/FNxuba Consumer
https://goo.gl/LgLWGD println(String x)

9

Untility Collections

- Empty series
  - emptyList, emptySet, emptyMap, emptyIterator, emptyEnumeration
- Unmodifiable series
  - unmodifiableList, unmodifiableSet, unmodifiableMap…
- Synchronized series
  - synchronizedList, synchronizedSet, synchronizedMap…

```
List<Integer> emp = Collections.EMPTY_LIST;
List<Integer> emp = Collections.emptyList();

List<Integer> lst = Collections.unmodifiableList(lst);

List<Integer> lst = Collections.synchronizedList(lst);

Collection<Integer> col = Collections.checkedCollection(list,
Integer.class);

int r = Collections.binarySearch(list, 2);
```

10

도전하세요!

- Map 만들기
  - Map<Integer, String> map
  - {0: "private", 1: "business", 2: "root"}
- Set으로 변환하기
  - map을 Set<Entry<Integer, String>> set로 변환하고 "수정할 수 없도록" 변경
  - set을 Thread-safe하게 변경
- 화면에 모두 출력

```
Map<Integer, String> map = ?
Set<Entry<Integer, String>> set = ?
set = ?
set.stream().forEach(System.out::println);
```

```
2=root
0=private
1=business
```

11

정리

- initializer
- loop
- collections helper

12