13 Lambda Expression

Object Oriented Programming

Terminologies

- Function (타 언어 programming): 기능과 동작을 정의

```
void abc(){
 //기능 및 동작
}
```

- Method
  - Java 에서의 function
  - Class 또는 Interface 내에 정의
  - 사용하려면 object 생성 후, object.method 형태로 call

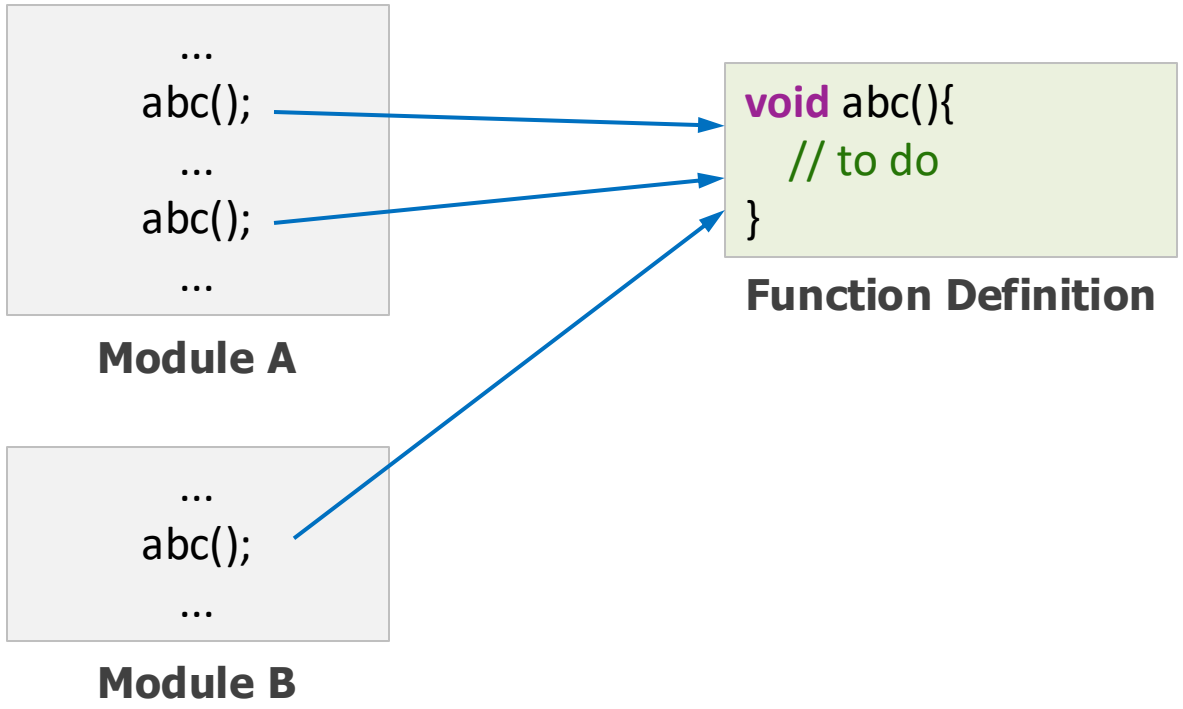
```
class A{
 void methodABC(){
 //기능 및 동작
 }
}
```

- Functional Interface
  - abstract method를 1개만 가지는 interface

```
interface A{
  public abstract void abc();
}
```

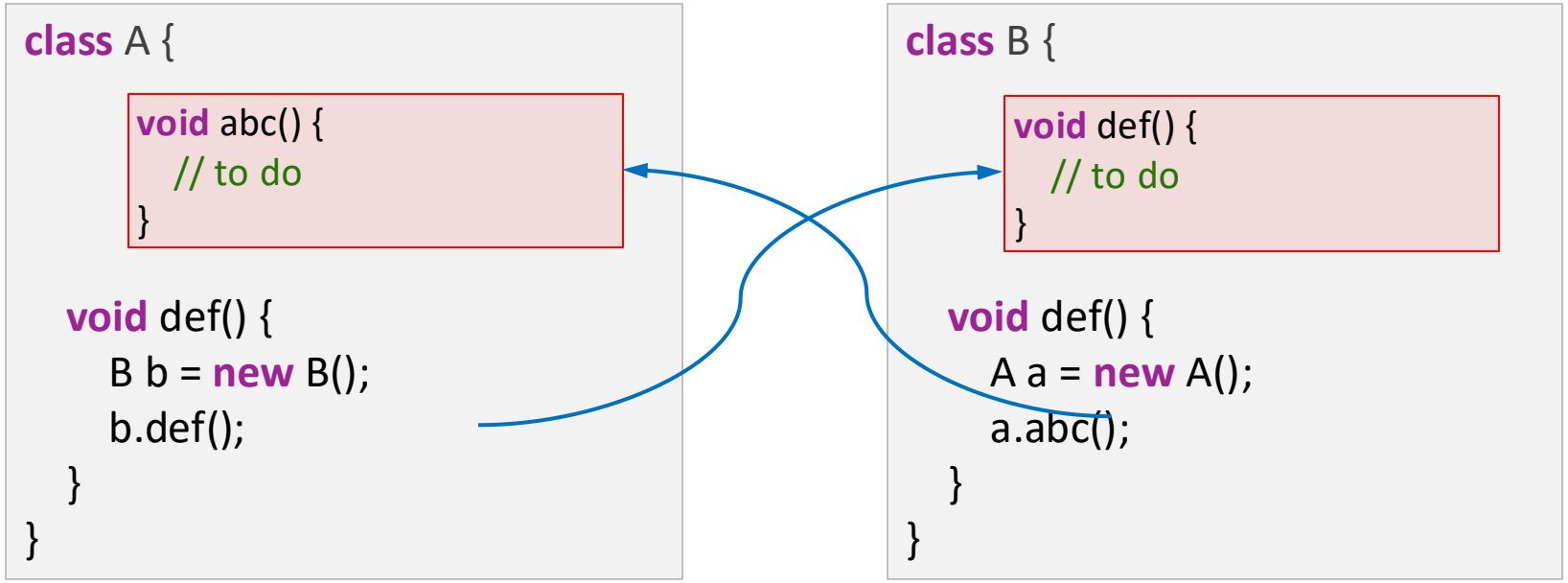
Functional Programming

- Function definition outside the execution module
- Function call

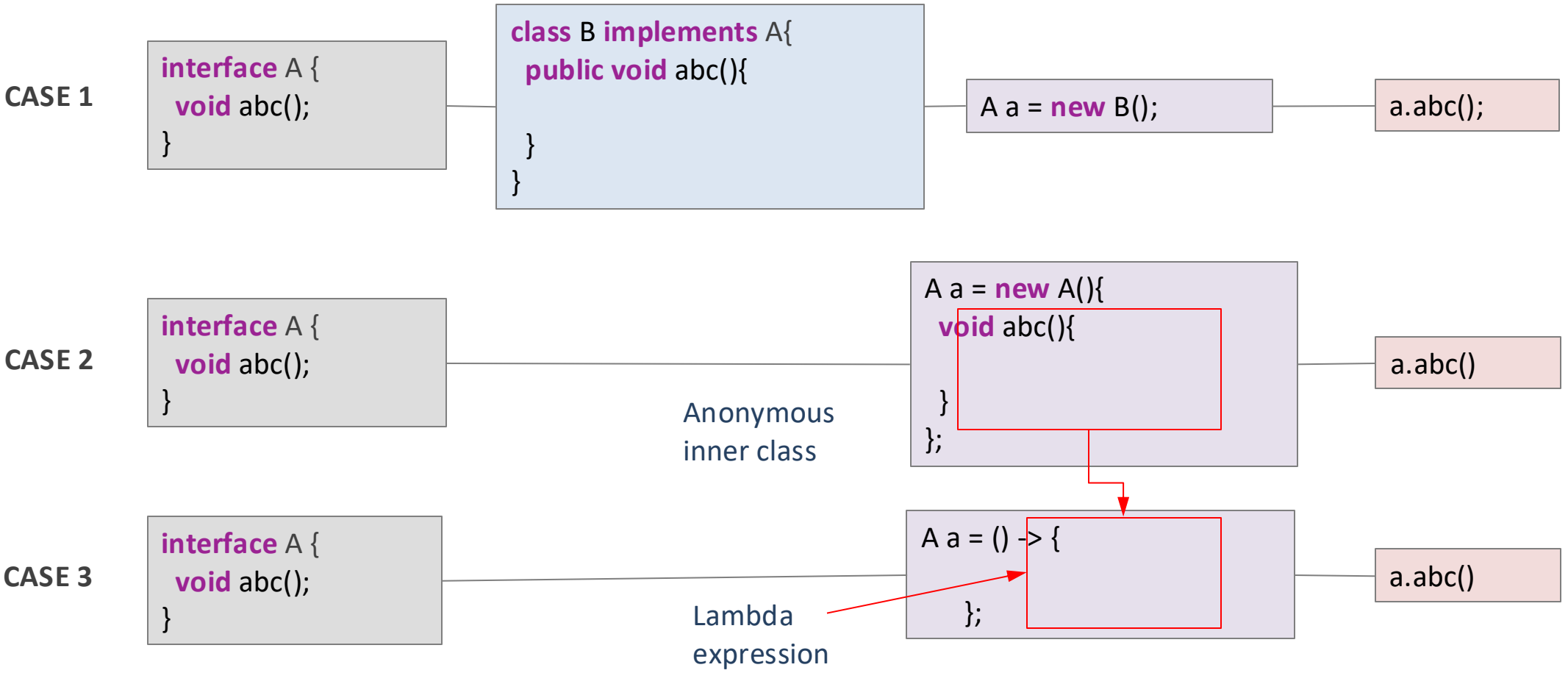


Object Oriented Programming

- Method definition inside the class
- Method call via class object



Abstract Method Implementation and Call



Example: OOPvsFP (1/2)

```
interface A {
   void abc();
}

class B implements A {
   @Override
   public void abc() {
      System.out.println(”method 1");
   }
}

public class OOPvsFP {
   public static void main(String[] args) {
      // CASE 1: object oriented 문법 1 (class implementation 사용)
      A a1 = new B();
      a1.abc();
```

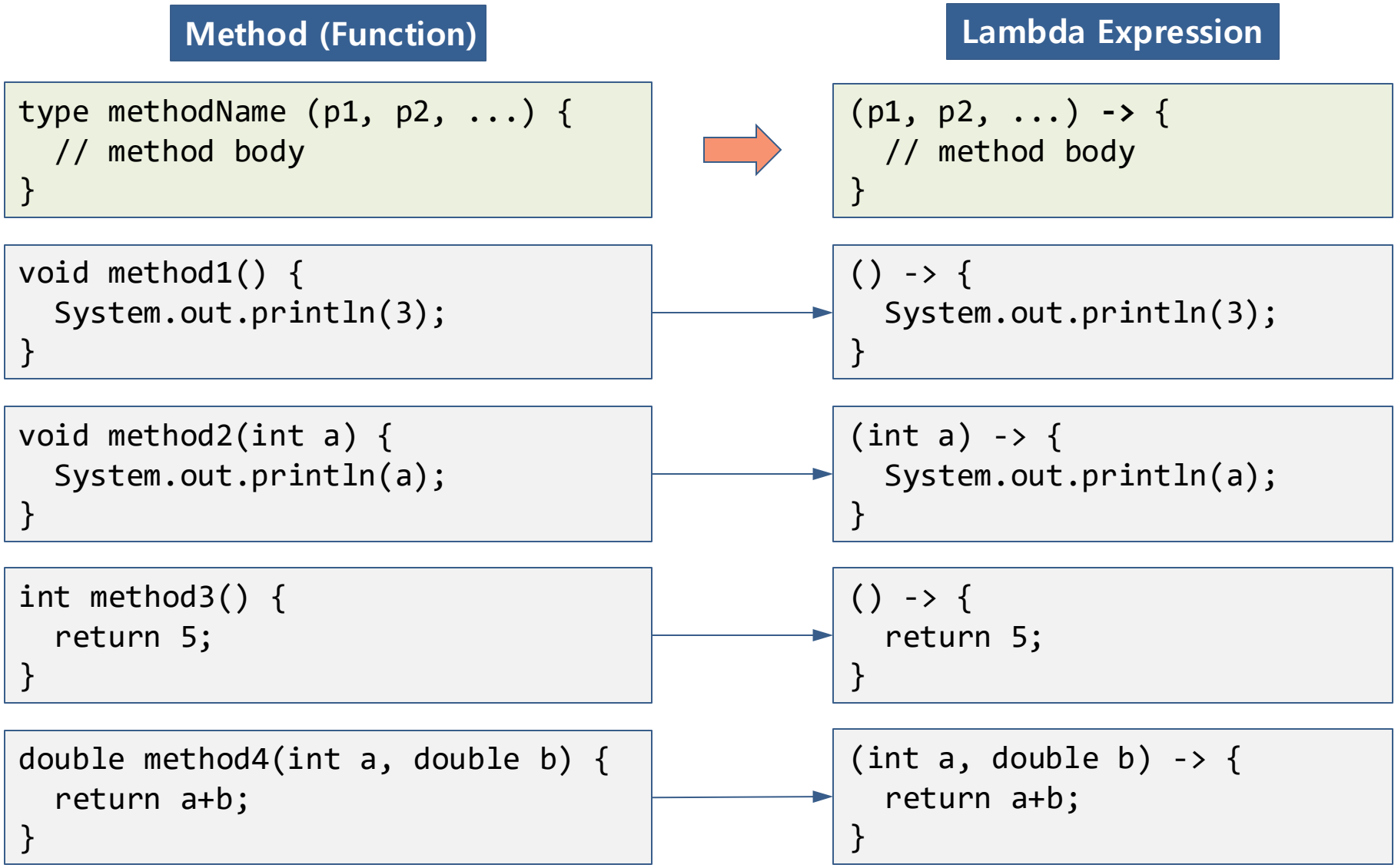
Example: OOPvsFP (2/2)

```
        // CASE 2: object oriented 문법 2 (Anonymous class 사용)
        A a2 = new A() {
            @Override
            public void abc() {
                System.out.println(”method 2");
            }
        };
        a2.abc();

        // CASE 3: Functional Programming 문법 (Lambda expression 사용)
        A a3 =  () -> {System.out.println(”method 3");};
        a3.abc();
    }
}
```

OUTPUT:
method 1
method 2
method 3

From Method to Lambda Expression



Shorthand Representation

Statement가 하나인 경우 중괄호 생략가능

```
A a = ()  -> {System.out.println("test");};
```

→

```
A a = () -> System.out.println("test");
```

Parameter's type 생략 가능

```
A a = (int a) -> { ... };
```

→

```
A a = (a) -> { ... };
```

Parameter 가 한 개인 경우 ( ) 생략 (type은 반드시 함께 생략)

```
A a = (int a) -> { ... };
```

→

```
A a = a → { ... };
```

Statement로 return만 있는 경우 return 생략 가능 (중괄호도 반드시 함께 생략)

```
A a = (int a, int b) → { return a + b; };
```

→

```
A a = (a, b) -> a + b;
```

Three Uses for Lambda Expressions

- Type 1: Shorthand representations of anonymous inner class implementation methods
- Type 2: Method reference
  - Importing functionality that already exists, rather than new implementation of the method
- Type 3: Constructor reference
  - The implementation method is fixed by the object creation code alone

Type 1: Shorthand Implementation of Method

```
interface A {
  double method4(int a, double b);
}
```

Functional Interface

```
A a = new A(){
  public double method4(int a, double b){
    return ...
  }
}
```

Anonymous Inner Class

```
A a = (int a, double b)->{ return ... };
```

Lambda Expression

Examples) FunctionToLambdaExpression2

```
interface Bftl2 {
   void method2(int a);
}
interface Dftl2 {
   double method4(int a, double b);
}

public class FunctionToLambdaExpression2 {
   public static void main(String[] args) {

      Bftl2 b2 = (int a)->{System.out.println("a = " + a);};
      b2.method2(3);

      Dftl2 d4 = (a, b)-> a + b;
      System.out.println(d4.method4(2,3));

   }
}
```

OUTPUT:
a = 3
5.0

Type 2: Instance Method Reference

```
interface A {
    void abc();
}
class B {
    void bcd() {
        System.out.println("method");
    }
}
```

```
// Lambda Expression
A a2 = () -> {
    B b = new B();
    b.bcd();
};

// Instance Method Reference
B b = new B();
A a3 = b::bcd;
```

- a2를 생성하는 Lambda expression의 역할은 b.bcd() 를 call하는 것 뿐
- 이 경우, A a3 = b::bcd; 와 같이 축약된 method reference를 사용 가능
- a3.abc() call을 b.bcd() call로 대체하라는 뜻

Type 2: Static Method Reference

```
interface A {
   void abc();
}
class B {
   static void bcd() {
      System.out.println("method");
   }
}
```

```
// Lambda Expression
A a2 = () -> {
   B.bcd();
};

// Static Method Reference
A a3 = B::bcd;
```

- a2를 생성하는 Lambda expression의 역할은 class B의 static method인 B.bcd()를 call하는 것 뿐
- 이 경우, A a3 = B::bcd; 와 같이 축약된 method reference를 사용 가능
- a3.abc() call을 B.bcd() call로 대체하라는 뜻

Type 3: Array Constructor Reference

```
interface A {
   int[] abc(int len);
}

// Lambda Expression
A a = (len) -> {  new int[len];  };

// Array Constructor Reference
A a = int[]::new;
```

- Lambda expression이 하는 역할은 length가 len인 int array를 생성하는 것 뿐
- 이런 경우 A a = int[]::new; 로 축약하여 표현 가능함

Type 3: Class Constructor Reference

```
interface A {
   B abc(int);
}
class B {
   B() { }
   B(int k) { }
}
```

```
// Lambda Expression
A a = (k) -> new B(k);

// Class Constructor Reference
A a = B::new;
```

- 여기서 lambda expression의 역할은 B의 constructor B(int)를 call하여 reference a의 object를 생성하는 것
- Class constructor reference의 축약표현 A a = B::new; 는 B의 default constructor B() 를 call하는 것이 아니라 B(int) 를 call 하는 것임 (interface A의 정의로 부터 유추 가능함)