

자바 제어자 (modifier) #1

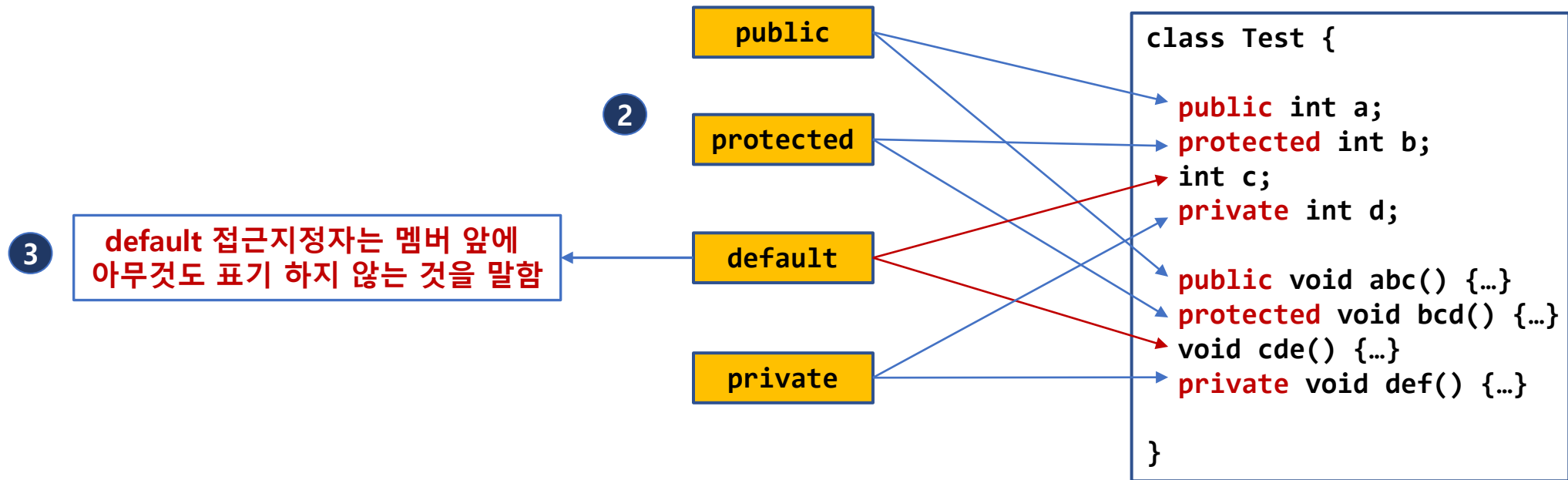
접근제어자, static

자바 제어자 (modifier) #1 - 접근지정자

자바 제어자 (modifier) #1 - 접근지정자

☞ 멤버 및 생성자의 접근지정자

- ① - 멤버 및 생성자에서 사용되는 4가지 접근지정자 (public, protected, default, private)



자바 제어자 (modifier) #1 - 접근지정자

☞ 멤버 및 생성자의 접근지정자

① - 멤버 및 생성자에서 사용되는 4가지 접근지정자의 사용가능 범위

②

접근범위 넓음 ↑	public	동일 패키지 + 다른 패키지의 모든 클래스에서 사용 가능
	protected	동일 패키지 + 다른 패키지의 <u>자식</u> 클래스에서 사용 가능
	default	동일 패키지 + 같은 패키지 내의 모든 클래스에서 사용 가능
↓ 접근범위 좁음	private	동일 클래스 내에서 사용 가능

자바 제어자 (modifier) #1 - 접근지정자

☞ 멤버 및 생성자의 접근지정자

① - 멤버에서 사용되는 4가지 접근지정자의 사용가능 범위

2

패키지 abc

```
package abc;
public class A {

    public int a;
    protected int b;
    int c;
    private int d;

    void abc(){
        a, b, c, d 사용가능
    }
}
```

```
package abc;
class B {
    a, b, c 사용가능
}
```

3

패키지 bcd

```
package bcd;
class C {

    a 사용가능

}
```

```
package bcd;

class D extends A {

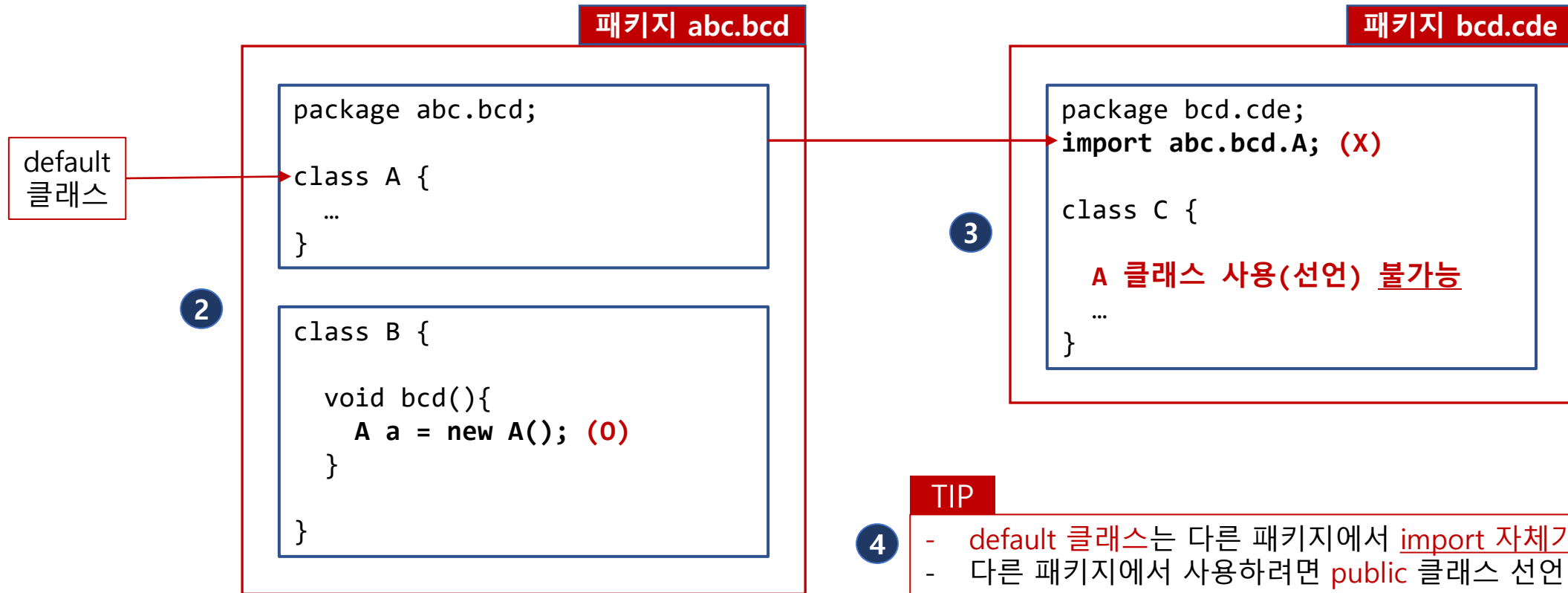
    a, b 사용가능

}
```

자바 제어자 (modifier) #1 - 접근지정자

👉 클래스의 접근지정자

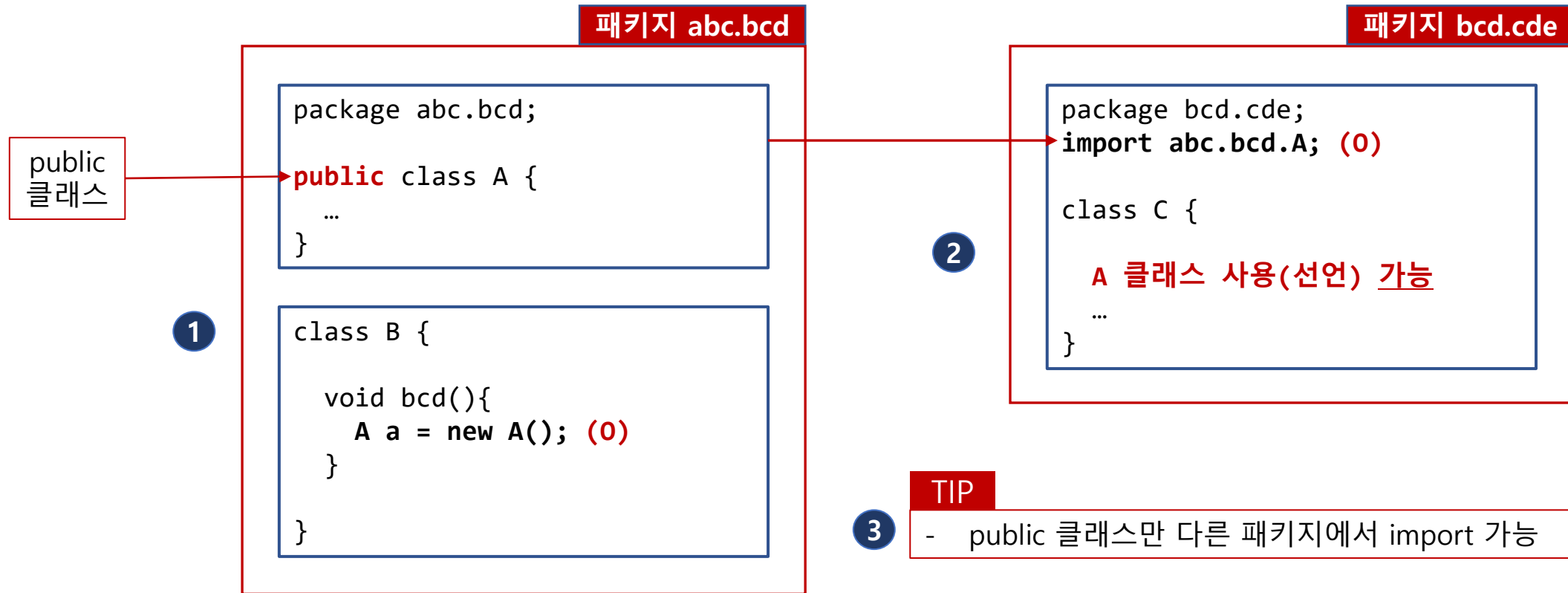
- 1 - 클래스에서 사용되는 2가지 접근지정자 (public, default)



자바 제어자 (modifier) #1 - 접근지정자

👉 클래스의 접근지정자

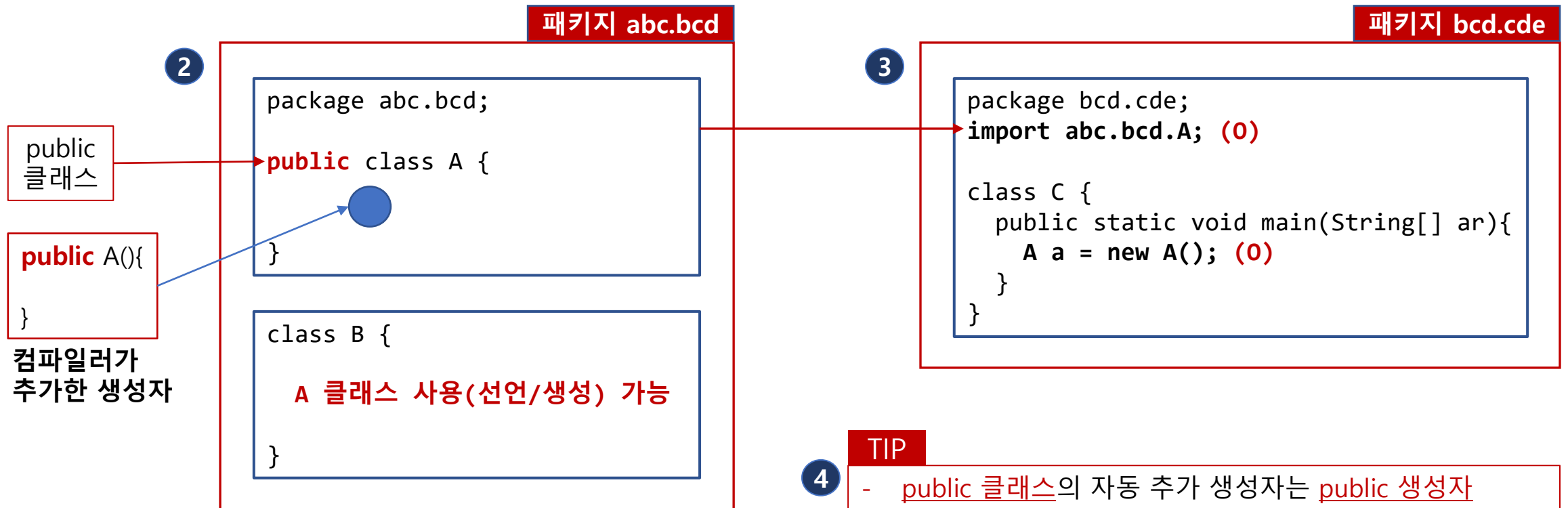
- 클래스에서 사용되는 2가지 접근지정자 (public, default)



자바 제어자 (modifier) #1 - 접근지정자

👉 클래스의 접근지정자 vs. 생성자의 접근지정자

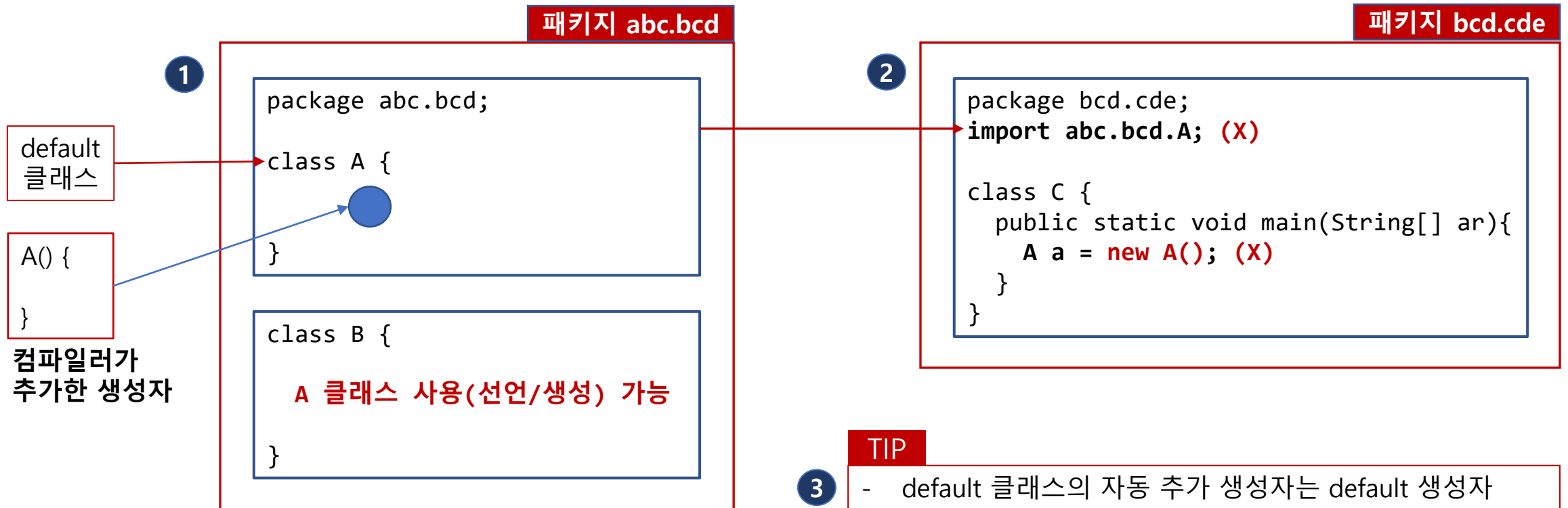
- 1 - 컴파일러가 자동 생성하는 기본생성자의 접근지정자는 클래스의 접근지정자와 동일



자바 제어자 (modifier) #1 - 접근지정자

☞ 클래스의 접근지정자 vs. 생성자의 접근지정자

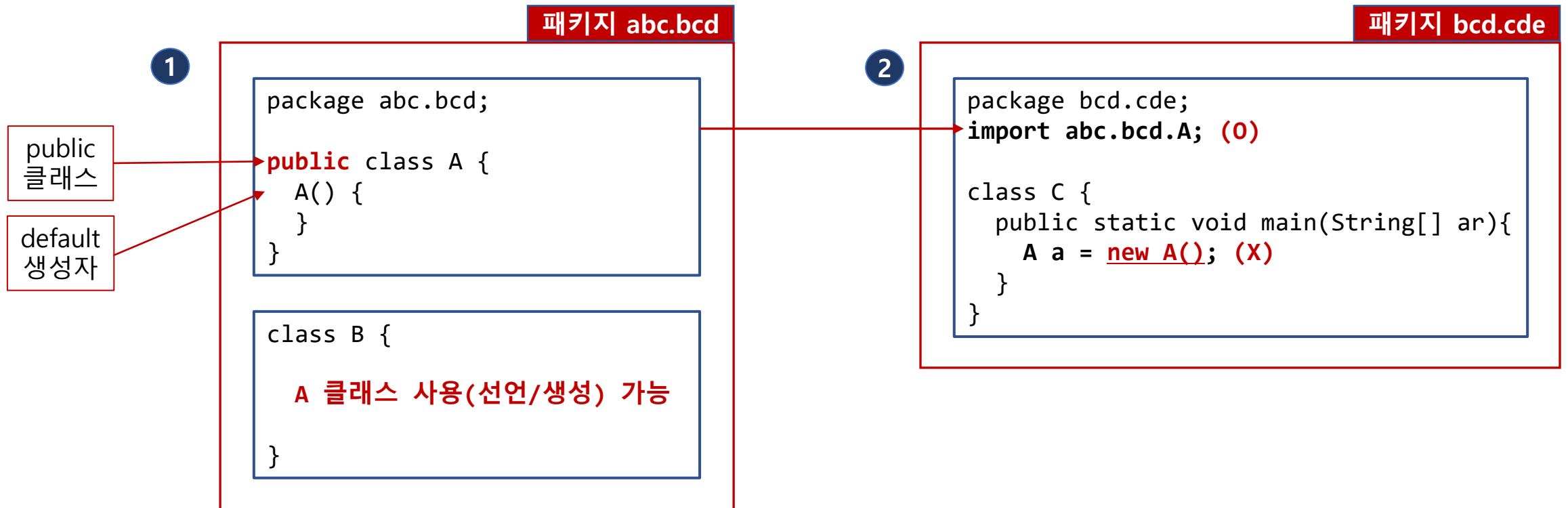
- 컴파일러가 자동 생성하는 기본생성자의 접근지정자는 **클래스의 접근지정자와 동일**



자바 제어자 (modifier) #1 - 접근지정자

☞ 클래스의 접근지정자 vs. 생성자의 접근지정자

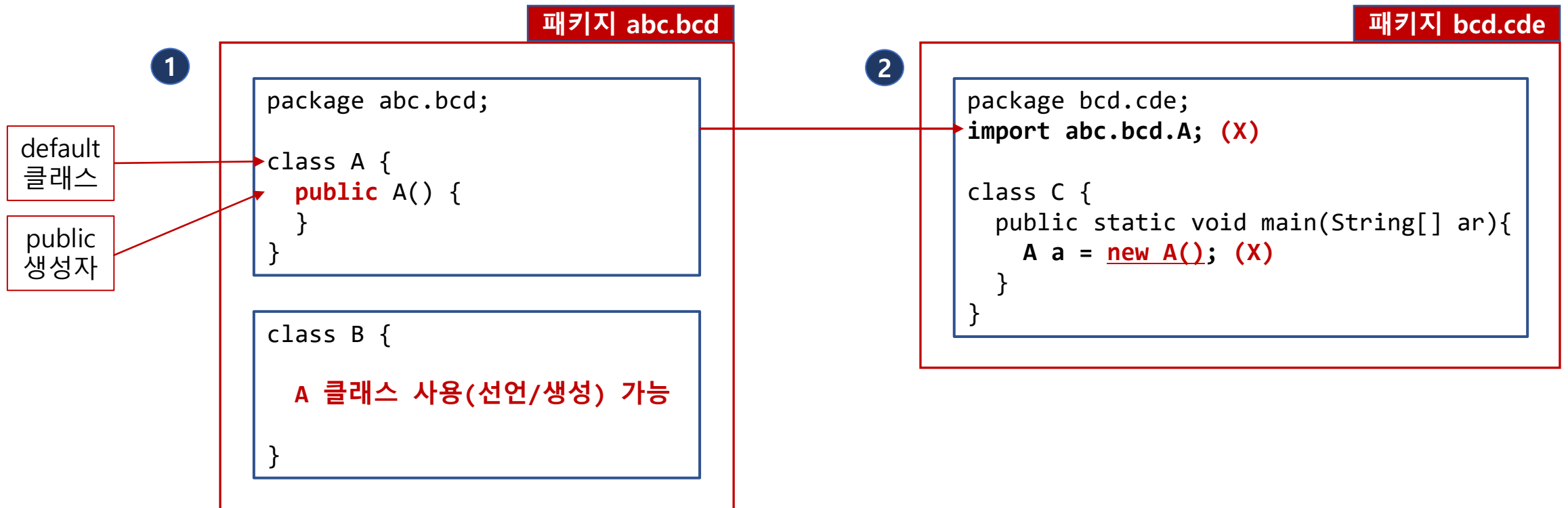
- public 클래스 + default 생성자



자바 제어자 (modifier) #1 - 접근지정자

☞ 클래스의 접근지정자 vs. 생성자의 접근지정자

- default 클래스 + public 생성자



The End

자바 제어자 (modifier) #1 - static

자바 제어자 (modifier) #1- static

- 1 🖱 static 키워드 (필드) → 객체생성 없이 바로 사용 가능

2

```
class A{  
    int m = 3;  
    static int n = 5;  
}
```

인스턴스 필드는 객체를 생성한 후 사용 가능

static 필드는 객체 생성 없이 사용 가능

인스턴스 필드의 활용 방법 #1

3

```
A a = new A();  
System.out.println(a.m); //3
```

→ 객체생성 후에만 사용 가능

static 필드의 활용 방법 #1

```
System.out.println(A.n);
```

→ 객체생성 없이 바로 사용

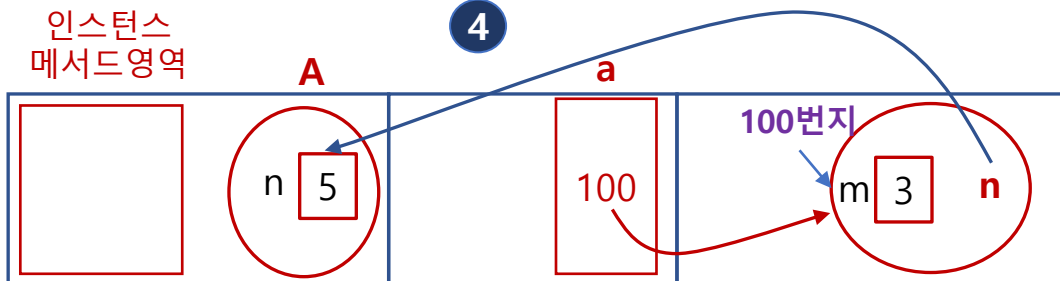
5

static 필드의 활용 방법 #2

```
A a = new A();  
System.out.println(a.n);
```

→ 객체생성 후 사용 (지양할 것)

4



→ static 필드는 힙메모리 객체에는 저장공간이 없음
→ 저장공간은 static 영역에 있음

자바 제어자 (modifier) #1- static

☞ static 키워드 (필드) → 객체생성 없이 바로 사용 가능

1 - static 필드의 객체간 공유기능

```
A a1 = new A();
A a2 = new A();

a1.m=5;
a2.m=6;

System.out.println(a1.m);
System.out.println(a2.m);

a1.n=7;
a2.n=8;

System.out.println(a1.n);
System.out.println(a2.n);

A.n=9;
System.out.println(a1.n);
System.out.println(a2.n);
```

5
6

8
8

9
9

2

```
class A{
```

```
int m = 3;
```

```
static int n = 5;
```

```
}
```

인스턴스 필드는 객체를
생성한 후 사용 가능

static 필드는 객체
생성 없이 사용 가능

4

인스턴스
메서드영역

A

a2

a1

100번지

200번지

n

7

8

9

200

100

m

5

n

m

6

n

static 필드는 힙메모리
객체에는 저장공간이 없음
저장공간은 static 영역에 있음

자바 제어자 (modifier) #1- static

1 🖱 static 키워드 (메서드) → 객체생성 없이 바로 사용 가능

2

```
class A{  
    void abc(){  
        System.out.println("인스턴스메서드");  
    }  
    static void bcd(){  
        System.out.println("static메서드");  
    }  
}
```

인스턴스 메서드는 객체를 생성한 후 사용 가능

static 메서드는 객체 생성 없이 사용 가능

3

인스턴스 메서드의 활용 방법 #1

```
A a = new A();  
a.abc(); //인스턴스메서드
```

→ 객체생성 후에만 사용 가능

4

static 필드의 활용 방법 #1

```
A.bcd(); //static메서드
```

→ 객체생성 없이 바로 사용

static 필드의 활용 방법 #2

```
A a = new A();  
a.bcd(); //static메서드
```

→ 객체생성 후 사용 (지양할 것)

자바 제어자 (modifier) #1- static

☞ static 키워드 (메서드) → 객체생성 없이 바로 사용 가능

2

```
A.bcd();
```

```
A a = new A();  
a.abc();  
a.bcd();
```

static 메서드

인스턴스 메서드
static 메서드

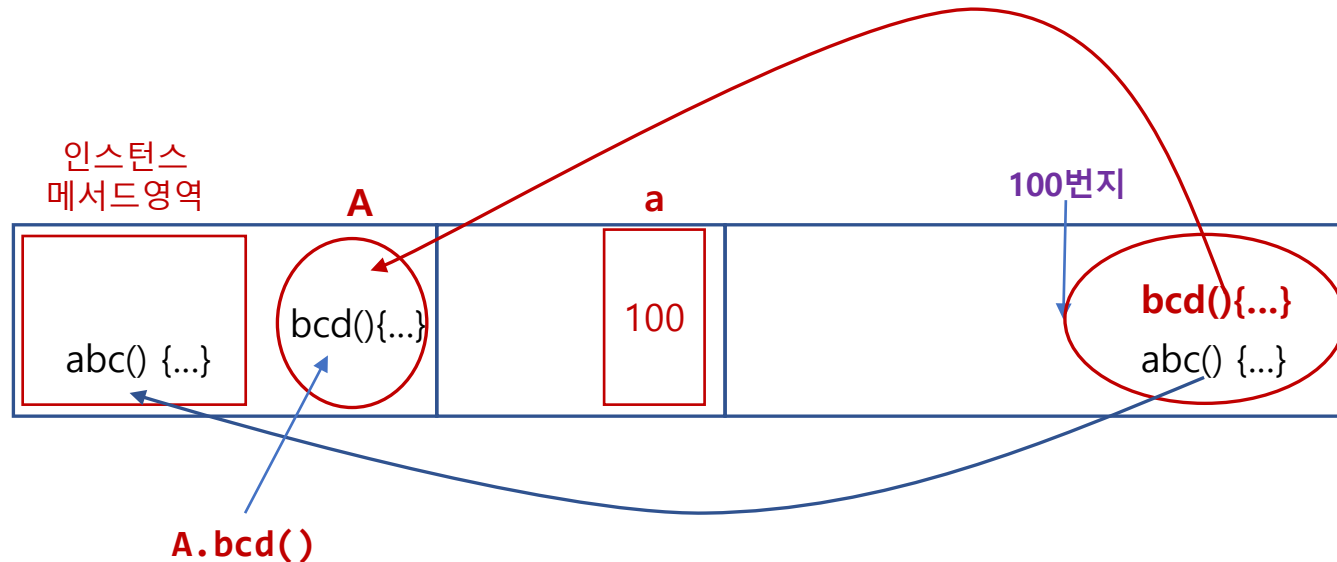
1

```
class A{  
    void abc(){  
        System.out.println("인스턴스 메서드");  
    }  
    static void bcd(){  
        System.out.println("static 메서드");  
    }  
}
```

일반 메서드는 객체를 생성한 후 사용 가능

static 메서드는 객체 생성 없이 사용 가능

3



자바 제어자 (modifier) #1- static

☞ static 메서드에서 사용가능한 필드 및 메서드

1 → static 메서드내에서는 static 멤버만 사용 가능함 (객체 생성이전에 사용가능해야 하기 때문)

객체 생성후
사용 가능

```
class A{
    int a;
    static int b; → 객체 생성전 사용 가능
    void abc(){
        a, b, bcd(), cde() 사용가능
    }

    static void bcd(){ → 객체 생성전 사용 가능
        b, cde() 사용가능
    }

    static void cde(){ → 객체 생성전 사용 가능
        b, bcd () 사용가능
    }
}
```

2

정적메서드 내부에서는 클래스 객체 참조변수인 this 키워드를 사용할 수 없음

↓ 3

this 객체를 사용할 수 없으면 객체를 통해서만 접근할 수 있는 인스턴스 멤버를 사용할 수 없음

↓ 4

- 인스턴스 메서드 내부:
→ 인스턴스 멤버 + 정적 멤버 모두 사용 가능

- 정적 메서드 내부
→ 정적 멤버만 사용 가능

자바 제어자 (modifier) #1- static

☞ static 초기화 블록

- instance 필드의 초기화 위치 → 생성자

1

- static 필드의 초기화 위치 → static { } (생성자 호출없이 사용가능하기 위해)

```
class A{  
    int a;  
    static int b;  
  
    static {  
        b=5;  
        System.out.println("클래스가 로딩될 때 static block 실행");  
    }  
  
    A() {  
        a=3; //instance 필드의 초기화 위치  
    }  
}
```

2

Static 초기화 블록



3

```
System.out.println(A.b);
```

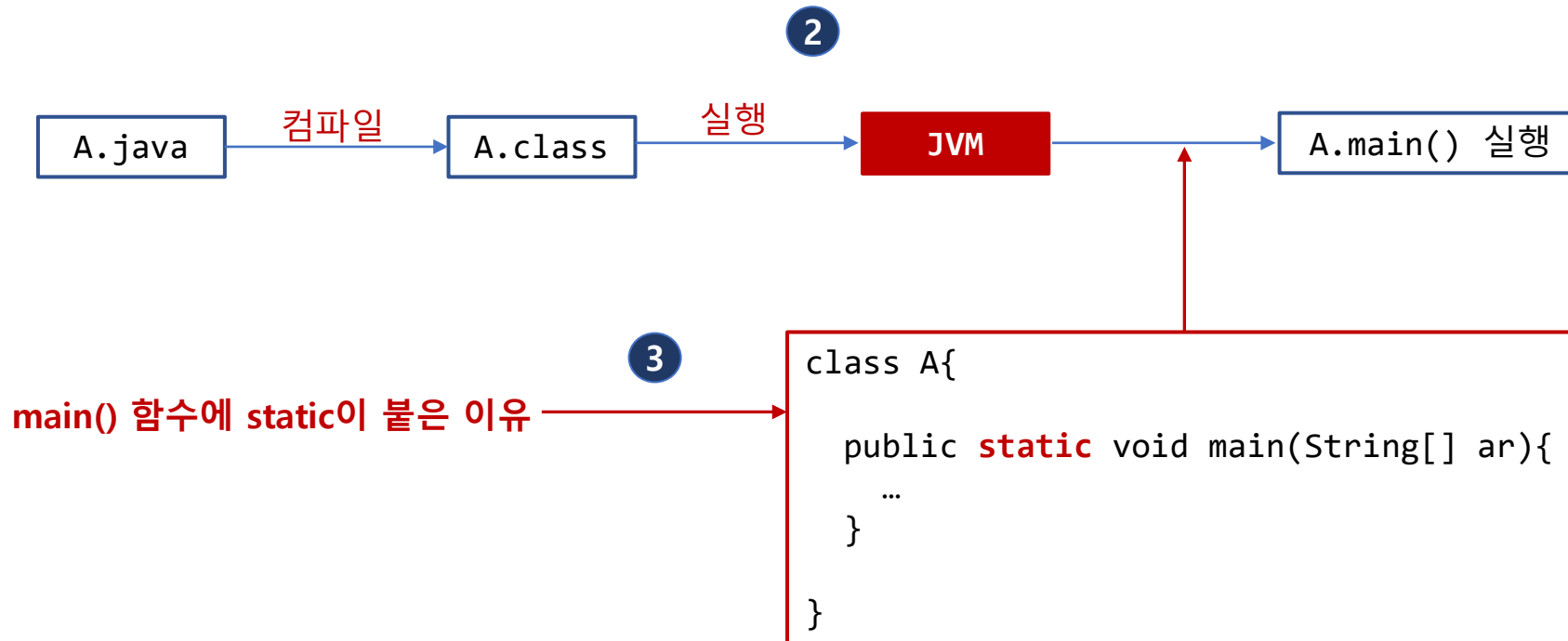


클래스가 로딩될때 static block 실행
5

자바 제어자 (modifier) #1- static

☞ static 키워드 (메서드)

① - JVM에서의 main 메서드 실행



The End