

클래스와 객체 II

01 생성자

02 캡슐화와 접근제어



생성자 개념과 목적

- 생성자
 - 객체가 생성될 때 초기화 목적으로 실행되는 메소드
 - 객체가 생성되는 순간에 자동 호출



기본 객체



생성자



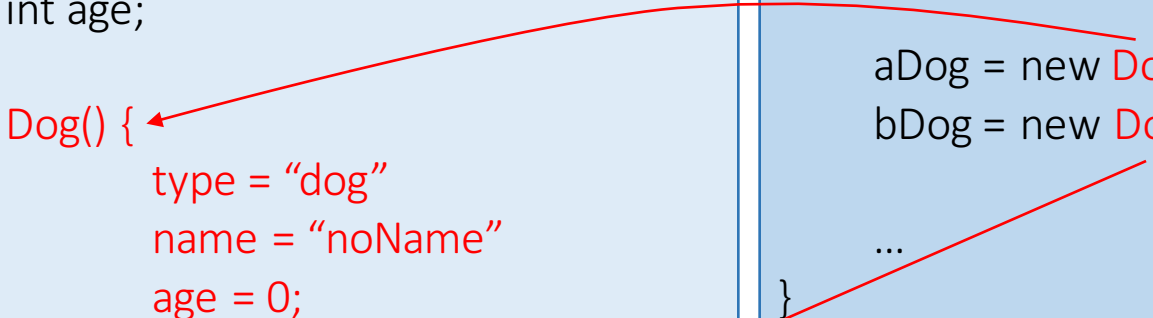
초기화된 객체

생성자

- 객체가 생성될 때 **멤버변수 초기화 목적**으로 실행되는 메소드

```
class Dog {  
    String type;  
    String name;  
    int age;  
  
    Dog() {  
        type = "dog"  
        name = "noName"  
        age = 0;  
    }  
  
    Dog(String t, String n, int a ) {  
        type = t;  
        name = n;  
        age = a;  
    }  
}
```

```
public static void main (String args[]) {  
    Dog aDog;  
    Dog bDog;  
  
    aDog = new Dog();  
    bDog = new Dog("삽살개","레오",2);  
  
    ...  
}
```



생성자의 특징

- 생성자 이름은 클래스 이름과 동일
- 생성자는 여러 개 작성 가능(생성자 중복)

```
public class Dog {  
    public Dog() {...} // 매개 변수 없는 생성자  
    public Dog (String t, String n, int a) {...} // 3개의 매개 변수를 가진 생성자  
}
```

- 생성자는 객체 생성시 한 번만 호출
 - 자바에서 객체 생성은 반드시 new 연산자로 함

```
aDog = new Dog();  
bDog = new Dog("삽살개","레오",2);
```

- 생성자의 목적은 객체 생성 시 초기화
- 생성자는 리턴 타입을 지정할 수 없음



```
public void Dog() {...} // 오류. void도 사용 안 됨
```

기본 생성자

- 기본 생성자(default constructor)
 - 매개 변수 없고, 아무 작업 없이 단순 리턴하는 생성자

```
class Circle {  
    public Circle() {} // 기본 생성자  
}
```

- 디폴트 생성자라고도 불림

기본 생성자가 자동 생성되는 경우

- 클래스에 생성자가 하나도 선언되어 있지 않을 때
 - 컴파일러에 의해 기본 생성자 자동 생성

```
public class Circle {  
    int radius;  
    void set(int r) { radius = r; }  
    double getArea() { return 3.14*radius*radius; }  
  
    public static void main(String[] args) {  
        Circle pizza = new Circle();  
        pizza.set(5);  
        System.out.println(pizza.getArea());  
    }  
}
```

컴파일러에 의해 기본
생성자 자동 삽입

public Circle() { }

호출

기본 생성자가 자동 생성되지 않는 경우

- 클래스에 생성자가 선언되어 있는 경우
 - 컴파일러는 기본 생성자를 자동 생성해 주지 않는다.

```
public class Circle {  
    int radius;  
    void set(int r) { radius = r; }  
    double getArea() { return 3.14*radius*radius; }
```

←-----X----- public Circle() { }

```
    public Circle(int r) {  
        radius = r;  
    }
```

호출

```
    public static void main(String[] args) {  
        Circle pizza = new Circle(10);  
        System.out.println(pizza.getArea());  
    }
```

오류

```
    Circle donut = new Circle();  
    System.out.println(donut.getArea());  
}
```

컴파일 오류.
해당하는 생성자 없음

실습1

- 지난주 실습1에서 정의한 `Rectangle` 클래스에 2개의 생성자를 정의하라
 - 매개변수가 없는 생성자
 - `width`, `height` 멤버변수를 임의의 양의 정수 값으로 설정
 - 매개 변수가 2개 인 생성자
 - `width`, `height` 멤버변수를 매개변수의 값으로 설정
- `main()` 함수는 다음과 같다.

```
public static void main(String [] args) {  
    Rectangle r1 = new Rectangle();  
    Rectangle r2 = new Rectangle(4,5);  
  
    System.out.println("사각형 r1의 면적은 " + r1.getArea());  
    System.out.println("사각형 r2의 면적은 " + r2.getArea());  
}
```


클래스와 객체 II

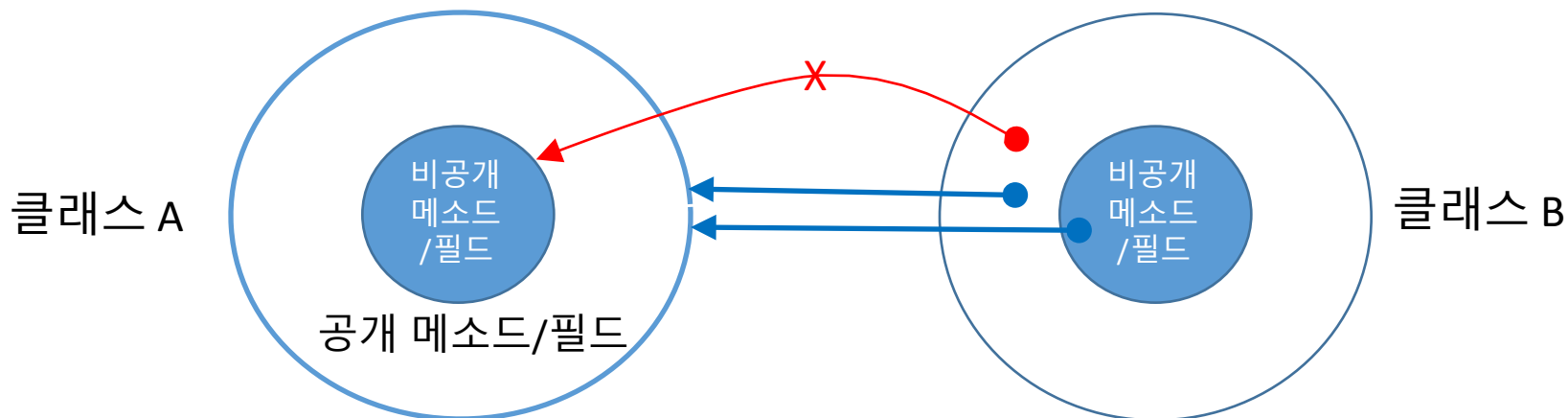
01 생성자

02 캡슐화와 접근제어



캡슐화 (클래스 설계 원리)

- 클래스 내에 메소드(함수)와 필드(변수)를 **멤버**로 정의
- 클래스 외부에서는 클래스의 **공개된 멤버만 접근 가능**
 - 보안, 보호, 외부접근 제한이 필요한 데이터는 비공개
 - 내부에서만 사용되는 함수는 비공개



• 장점

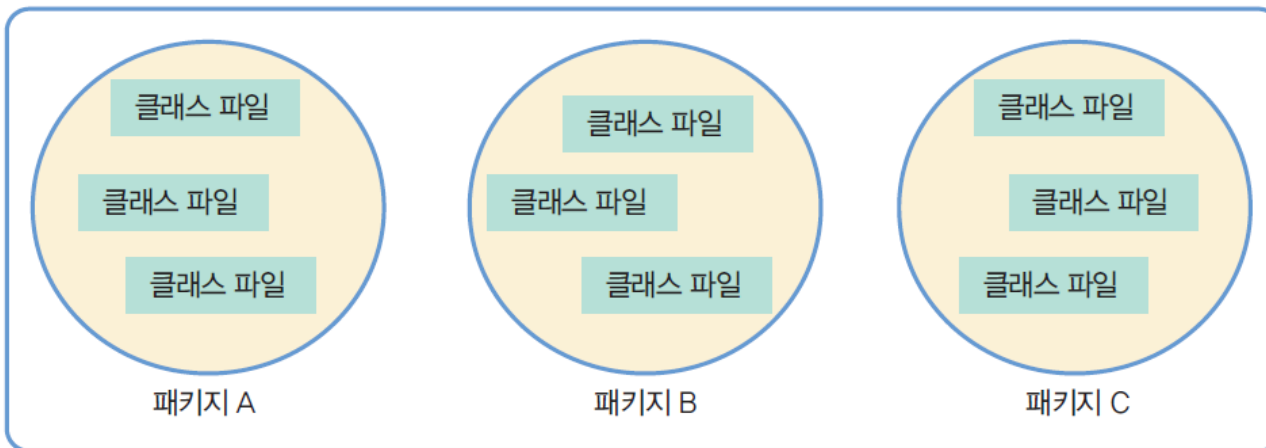
- 비공개 메소드/필드의 변경이 다른 클래스 코드에 영향을 주지 않는다. (유지 보수성 향상)
- 공개된 정보만 알고 사용하면 되므로, 코드의 이해도를 높여준다.

자바의 패키지 개념

- 패키지

- 상호 관련 있는 클래스 파일(컴파일된 .class)을 저장하여 관리하는 디렉터리
- 자바 응용프로그램은 하나 이상의 패키지로 구성

자바 응용프로그램

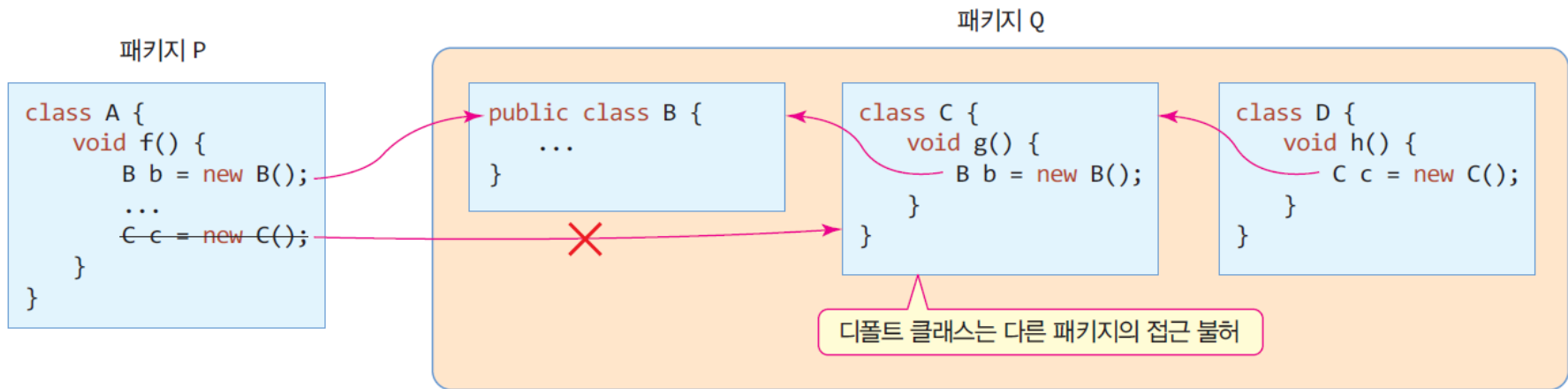


```
package com.kwanwoo.java;  
  
public class DogSimulator {  
    ...  
}
```

DogSimulator 클래스는
com.kwanwoo.java 패키지
내에 존재

접근 제어

- 클래스 수준에서의 접근 제어
 - public**: 다른 모든 클래스에서 접근 가능한 클래스임을 나타냄
 - 접근제어 수식어가 없는 경우**: 같은 패키지 안에 있는 클래스들만이 접근 가능함을 나타냄



* public으로 선언된 클래스는 반드시 같은 이름의 .java 파일에 정의되어 있어야 함.

접근 제어

- 멤버 수준에서의 접근 제어
 - **public**: 다른 모든 클래스에서 접근 가능한 멤버임을 나타냄
 - **접근제어 수식어가 없는 경우**: 같은 패키지 안에 있는 모든 클래스들에서 접근 가능함을 나타냄
 - **private**: 이 멤버를 정의한 클래스에서만 접근 가능한 멤버임을 나타냄

```
public class DogSimulator {  
    public static void main (...) {  
        Dog aDog;  
        aDog = new Dog();  
        ...  
        aDog.type = "진도개";  
        aDog.name = "화랑";  
        aDog.age = 3;  
        ...  
        aDog.bark();  
    }  
}
```

```
class Dog {  
    String type;  
    private String name;  
    private String age;  
    public void bark() {  
        System.out.println(...);  
    }  
    ...  
}
```

X
X

Getter/Setter 메소드

- private 멤버 변수를 다른 클래스에서 사용하기 위해서는 정의되는 메소드
- **Getter 메소드**
 - 필드의 값을 반환하는 메소드
 - getXXX() 형식
- **Setter 메소드**
 - 필드의 값을 설정하는 메소드
 - setXXX() 형식

```
package com.kwanwoo.java;

public class DogSimulator {
    public static void main (...) {
        Dog aDog;
        aDog = new Dog();
        ...
        aDog.type = "진도개";
        aDog.setName("화랑");
        aDog.setAge(3);
        ...
        aDog.bark();
    }
}
```

```
package com.kwanwoo.java;
...
class Dog {
    String type;
    private String name;
    private String age;

    public void setName(String name) {
        this.name = name;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

실습2

- 지난주 실습2를 바탕으로 진행
- 너비(width)와 높이(height) 필드, 그리고 면적 값을 제공하는 getArea() 메소드를 가진 Rectangle 클래스를 정의하라.
 - width와 height 필드는 **private** 멤버로 설정
 - width와 height의 getter/setter 메소드 정의
- Rectangle 클래스를 테스트하기 위한 main() 함수는 다음과 같다.

```
public static void main(String [] args) {  
    Rectangle r1 = new Rectangle();  
    Rectangle r2 = new Rectangle(4,5);  
  
    r1.setWidth(3);  
    r1.setHeight(5);  
  
    System.out.println("사각형 r1의 면적은 " + r1.getArea());  
    System.out.println("사각형 r2의 면적은 " + r2.getArea());  
}
```