

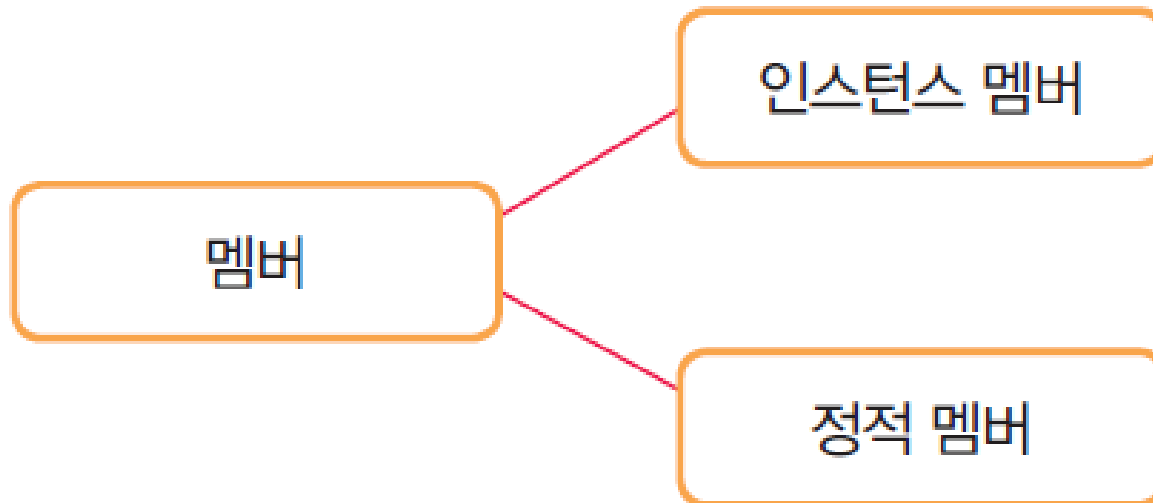
## 08. 정적 키워드



강동기

# 정적(static) 키워드

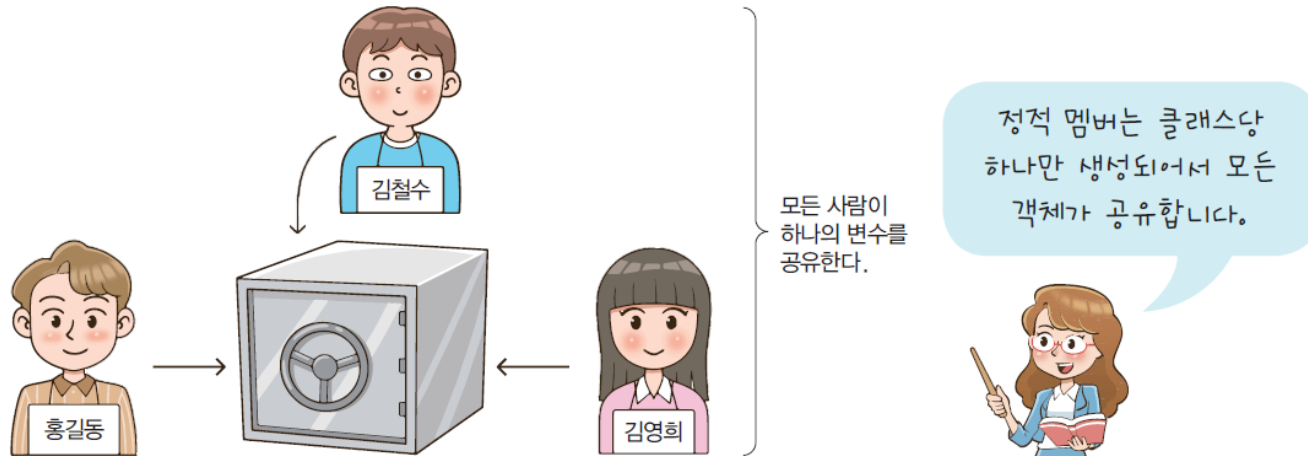
- static
- 클래스의 멤버(필드, 메서드, 이너 클래스)에 사용하는 제어자
- 인스턴스 멤버: static 이 붙어있지 않은 멤버
  - 인스턴스 필드, 인스턴스 메서드, 인스턴스 이너 클래스
- 정적 멤버: static 이 붙어있는 멤버
  - 정적 필드, 정적 메서드, 정적 이너 클래스



# 정적 멤버

- 정적(static) 멤버의 특징

- 객체의 생성 없이, '클래스명.멤버명'만으로 사용 가능
- 정적 필드는 모든 객체가 공통으로(공유하여) 가지는 변수
- 정적 필드는 하나의 클래스에 하나만 존재함



# 인스턴스 필드 vs. 정적 필드

- 인스턴스 필드
  - 객체마다 별도로 생성

```
class Television {  
    int channel;  
    int volume;  
    boolean onOff;  
}
```

channel	7
volume	9
onOff	<b>true</b>

객체 A

channel	9
volume	10
onOff	<b>true</b>

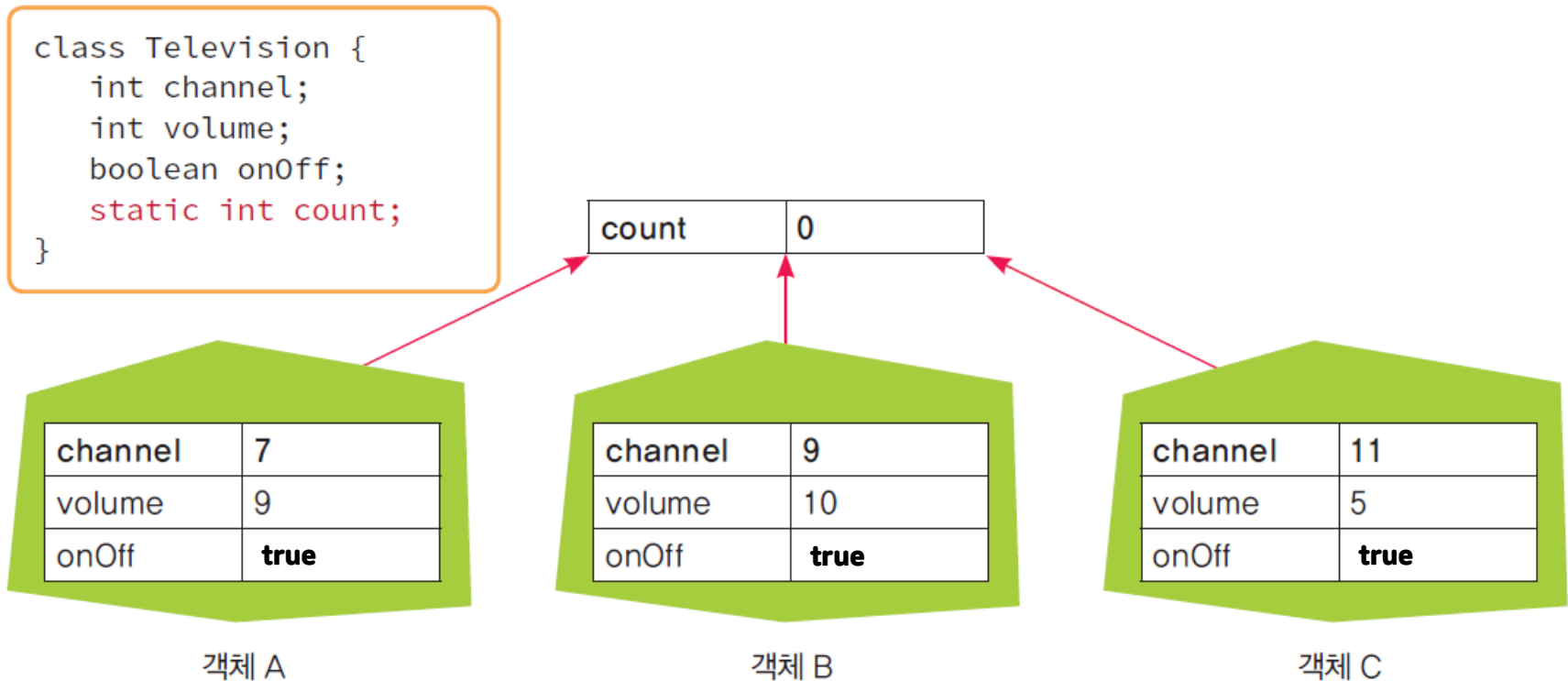
객체 B

channel	11
volume	5
onOff	<b>true</b>

객체 C

# 인스턴스 필드 vs. 정적 필드

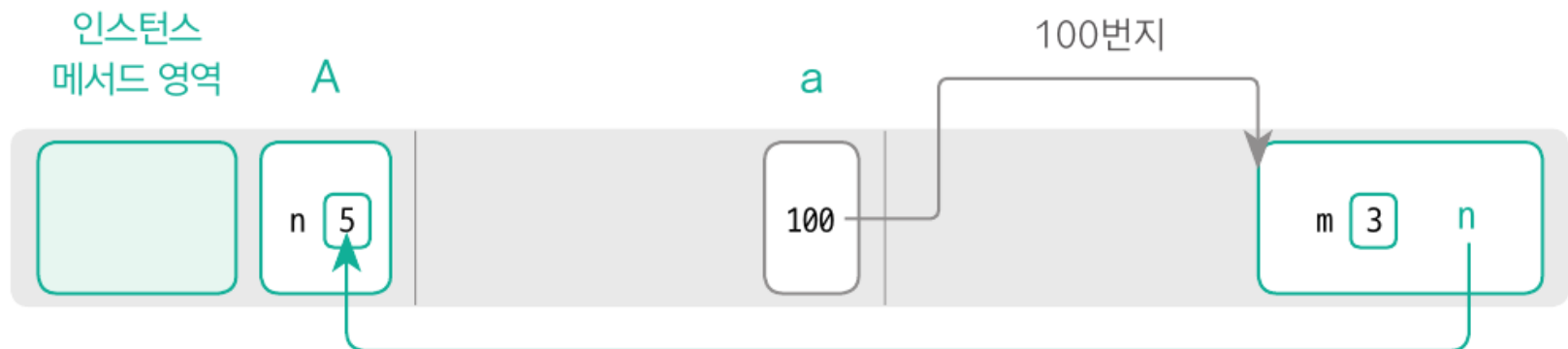
- 정적 필드
  - 모든 객체에 공통으로 존재함
  - 하나의 클래스에 하나만 존재함
  - 객체 생성을 하지 않고 접근 및 사용 가능



# 인스턴스 필드와 정적 필드의 메모리 구조

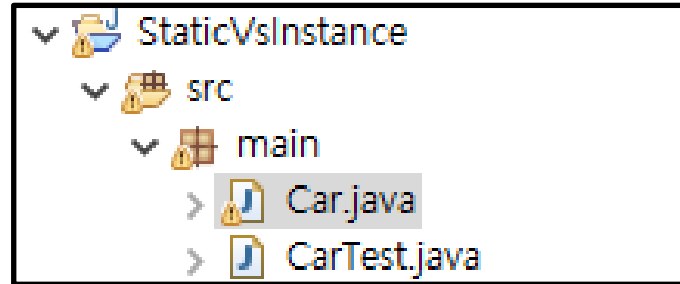
- 인스턴스 필드
  - 객체 내부에 생성, 즉 메모리 힙 영역에 저장됨
  - 값을 읽기 위해서는 참조 객체명을 사용하여야 함
- 정적 필드
  - 객체 외부에 생성, 데이터 영역(정적 영역)에 저장됨
  - 객체 내부의 참조 변수는 정적 필드 저장 위치를 가리킴
  - 값을 읽기 위해 참조 객체명을 사용하거나 혹은 사용하지 않아도 상관없음

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



# 인스턴스 필드와 정적 필드 예제

- 프로젝트 구조



# 인스턴스 필드와 정적 필드 예제

- Car.java

```
package main;

public class Car {
    private String model;
    private String color;
    private int speed;

    private int id;
    private static int numbers = 0;

    public Car(String m, String c, int s) {
        model = m;
        color = c;
        speed = s;
        id = ++numbers;
    }

    public void printCarId() {
        System.out.println(this.model + "'s id : " + this.id);
    }

    public void printCarNumbers() {
        System.out.println("The number of generated cars = " + numbers);
    }
}
```



# 인스턴스 필드와 정적 필드 예제

- CarTest.java

```
package main;

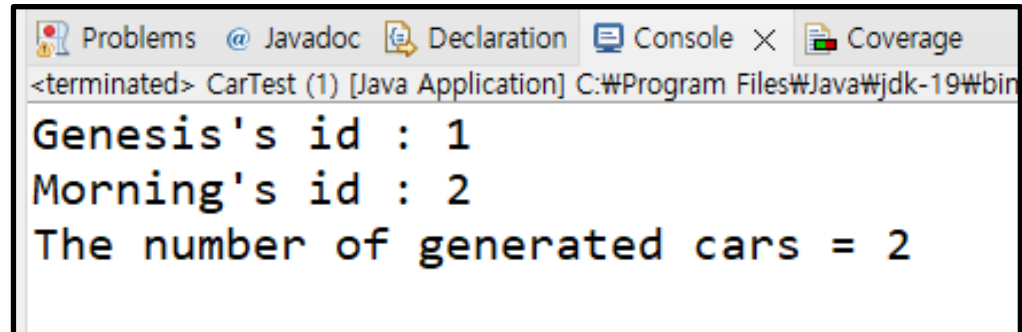
public class CarTest {
    public static void main(String args[]) {
        Car c1 = new Car("Genesis", "white", 80);
        Car c2 = new Car("Morning", "blue", 20);

        c1.printCarId();
        c2.printCarId();

        c1.printCarNumbers();
    }
}
```

# 인스턴스 필드와 정적 필드 예제

- CarTest.java 실행결과



The screenshot shows an IDE console window with the following tabs: Problems, Javadoc, Declaration, Console, and Coverage. The console output is as follows:

```
<terminated> CarTest (1) [Java Application] C:\Program Files\Java\jdk-19\bin  
Genesis's id : 1  
Morning's id : 2  
The number of generated cars = 2
```

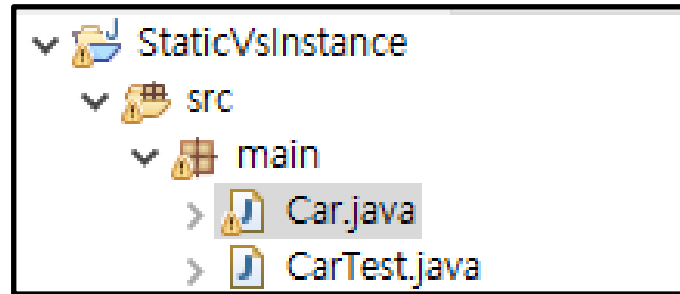
# 정적(static) 메서드

---

- 정적 메서드(static method)
  - static 키워드를 메소드 앞에 붙임
  - 정적 필드와 마찬가지로 객체 생성 없이, 클래스 이름을 통하여 호출 가능
  - (예) `double value = Math.sqrt(9.0);`

# 정적 메서드 예제

- 프로젝트 구조



# 정적 메서드 예제

- Car.java

```
package main;

public class Car {
    private String model;
    private String color;
    private int speed;

    private int id;
    private static int numbers = 0;

    public Car(String m, String c, int s) {
        model = m;
        color = c;
        speed = s;
        id = ++numbers;
    }

    public void printCarId() {
        System.out.println(this.model + "'s id : " + this.id);
    }

    public static void printCarNumbers() {
        System.out.println("The number of generated cars = " + numbers);
    }
}
```

# 정적 메서드 예제

- CarTest.java

```
package main;

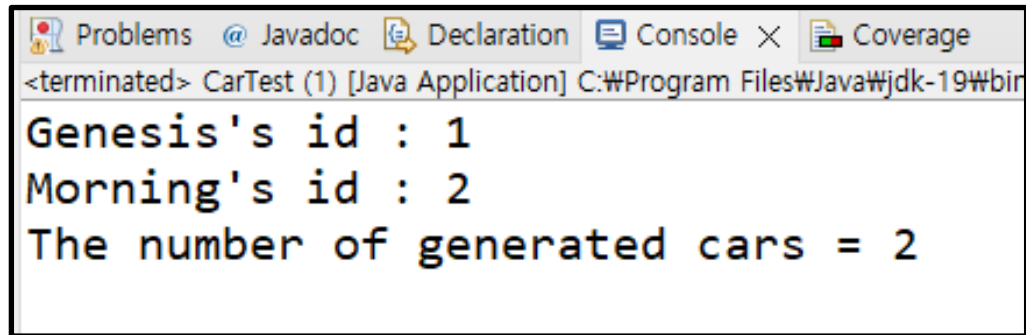
public class CarTest {
    public static void main(String args[]) {
        Car c1 = new Car("Genesis", "white", 80);
        Car c2 = new Car("Morning", "blue", 20);

        c1.printCarId();
        c2.printCarId();

        Car.printCarNumbers();
    }
}
```

# 정적 메서드 예제

- CarTest.java 실행결과



The screenshot shows an IDE console window with the following tabs: Problems, Javadoc, Declaration, Console, and Coverage. The console output is as follows:

```
<terminated> CarTest (1) [Java Application] C:\Program Files\Java\jdk-19\bin  
Genesis's id : 1  
Morning's id : 2  
The number of generated cars = 2
```

# 정적 메서드 사용시 주의점

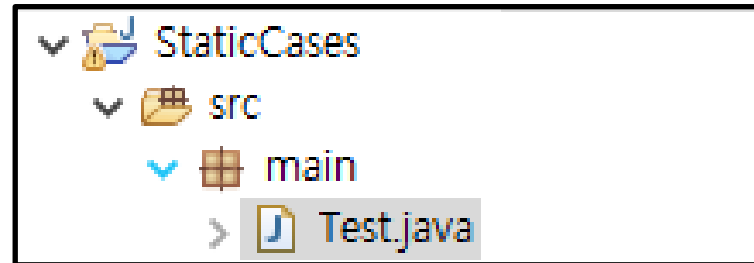
---

- 정적 메서드 안에서는...
  - 정적 멤버만 사용 가능
  - 인스턴스 필드를 사용할 수 없음 (객체 안에서만 존재하므로)
  - 다른 인스턴스 메서드 호출 불가 (객체가 생성되어야만 사용 가능하므로)
  - 정적 필드 및 (자기 안에 속한) 지역 변수만 사용 가능
  - 정적 메서드만 호출 가능
  - this 키워드 사용 불가
- 인스턴스 메서드 안에서는...
  - 제약 없음
  - 인스턴스 필드를 사용할 수 있음
  - 다른 인스턴스 메서드 호출 가능 (객체가 생성되어야만 사용 가능하므로)
  - 정적 필드 및 (자기 안에 속한) 지역 변수 사용 가능
  - 정적 메서드 호출 가능
  - this 키워드 사용 가능



# 정적 메서드 사용시 주의점 예제

- 프로젝트 구조



# 정적 메서드 사용시 주의점 예제

- Test.java

```
package main;

public class Test {
    int fieldA;
    int fieldB;
    static int sFieldA;

    public static void main(String[] args) {
        fieldA = 10; // 에러!
        fieldB = 20; // 에러!
        sFieldA = 30; // OK
    }
}
```

# 정적 메서드 사용시 주의점 예제

- 확장된 Test.java

```
package main;

public class Test {
    int fieldA;
    int fieldB;
    static int sFieldA;

    public static void main(String[] args) {
        fieldA = 10; // 에러!
        fieldB = 20; // 에러!
        sFieldA = 30; // OK

        add(10, 20); // 에러!
        mul(30, 40); // OK
    }

    int add(int x, int y) {
        return x + y;
    }

    static int mul(int x, int y) {
        return x * y;
    }
}
```

# 정적 초기화 블록

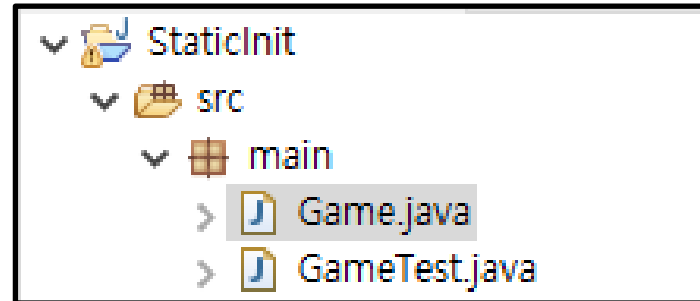
- 정적 초기화 블록
  - 정적 필드를 초기화하기 위한 별도의 문법
  - 정적 필드는 객체의 생성 이전에도 사용 가능해야 하므로 생성자 호출 전에 초기화 되어야 함
  - 클래스가 메모리(메서드 영역)에 로딩되는 시점에 가장 먼저 실행됨

## 정적 초기화 블록

```
static {  
    // 클래스가 메모리에 로딩될 때 실행되는 내용  
}
```

# 정적 초기화 블록 예제

- 프로젝트 구조



# 정적 초기화 블록 예제

- Game.java (1/2)

```
package main;

public class Game {

    Town firstTown = new Town();

    public void printTown() {
        System.out.print("FirstTown's name: " + firstTown.name);
        System.out.print(", the size: " + firstTown.size);
        System.out.println(", the number of people: " + firstTown.numOfPeople);
    }

    public void printProtagonistInfo() {
        System.out.print("Protagonist's name: " + Protagonist.name);
        System.out.print(", the age: " + Protagonist.age);
        System.out.print(", HP: " + Protagonist.HP + ", MP: " + Protagonist.MP);
        System.out.println(", Job: " + Protagonist.job);
    }
}
```

# 정적 초기화 블록 예제

- Game.java (2/2)

```
class Town{
    String name;
    double size;
    int numOfPeople;

    Town(){
        name = "Jeonju";
        size = 206.2; //km^2
        numOfPeople = 500000;
    }
}

class Protagonist{
    static String name;
    static int age; static int HP; static int MP;
    static String job; static String[] skills;

    static {
        name = "KDK";
        age = 60; HP = 30; MP = 50;
        job = "magician";
    }
}
```

# 정적 초기화 블록 예제

- GameTest.java

```
package main;

public class GameTest {
    public static void main(String[] args) {
        Game game = new Game();
        game.printTown();
        game.printProtagonistInfo();
    }
}
```



# 정적 초기화 블록 예제

---

- GameTest.java 실행 결과

```
FirstTown's name: Jeonju, the size: 206.2, the number of people: 500000  
Protagonist's name: KDK, the age: 60, HP: 30, MP: 50, Job: magician
```

# 연습문제 1

- 페이지19의 Test.java 소스 코드를 수정하여 에러가 발생하지 않게 하자.

```
package main;

public class Test {
    int fieldA;
    int fieldB;
    static int sFieldA;

    public static void main(String[] args) {
        fieldA = 10; // 에러!
        fieldB = 20; // 에러!
        sFieldA = 30; // OK

        add(10, 20); // 에러!
        mul(30, 40); // OK
    }

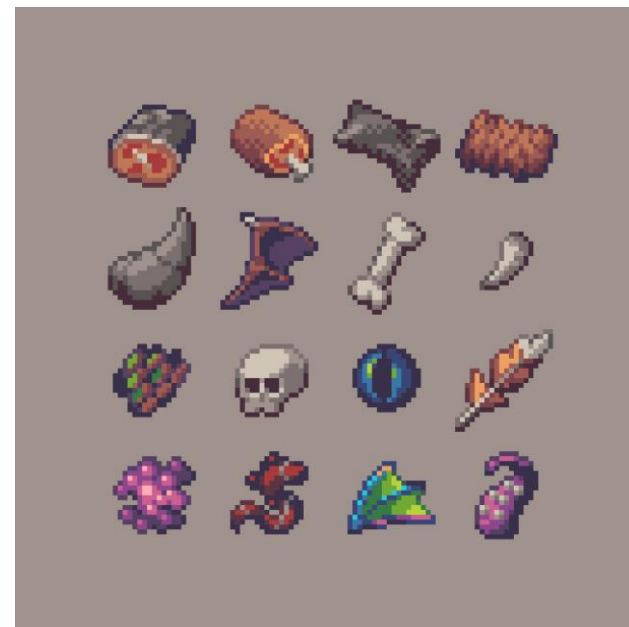
    int add(int x, int y) {
        return x + y;
    }

    static int mul(int x, int y) {
        return x * y;
    }
}
```

## 연습문제 2

- Game 클래스에 [아이템] 과 [몬스터] 클래스를 추가해보자
  - 인스턴스 필드와 정적 필드를 나누어서 구현하자
  - 생성자, static 초기화 블록, 인스턴스 메서드 등을 적절히 구현하자

```
...  
  
class Protagonist{  
    static String name;  
    static int age; static int HP; static int MP;  
    static String job; static String[] skills;  
  
    static {  
        name = "KDK";  
        age = 60; HP = 30; MP = 50;  
        job = "magician";  
    }  
}  
  
class Item{  
    ??????  
}  
  
class Monster{  
    ??????  
}
```



**감사합니다! XD**

