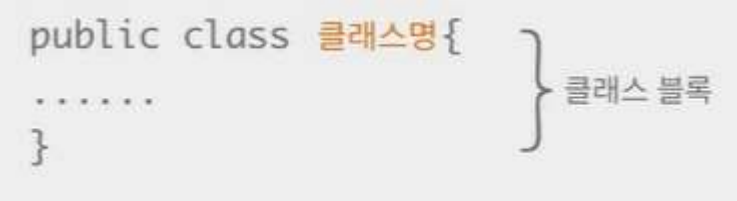
```
int[] iarr = {10,20,30,40,50};

   for(int value:iarr){
       System.out.println(value);
   }
```

class



String 클래스

new 연산자를 이용하지 않고 인스턴스 생성 가능

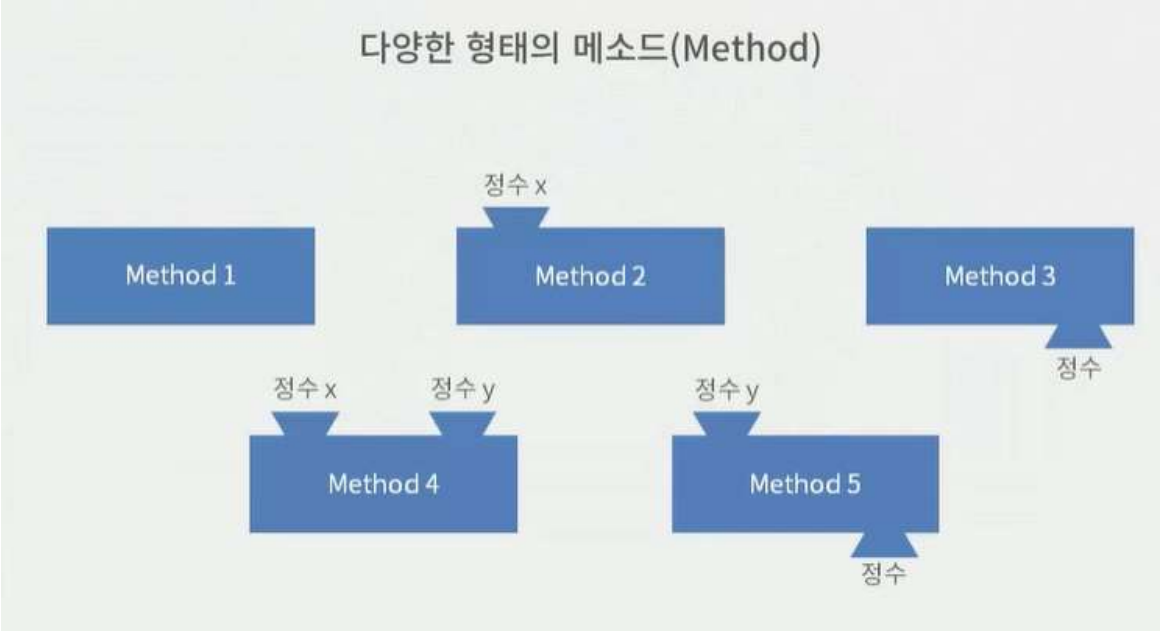
메소드

선언

public 리턴타입 메소드명 (매개변수) {

<필요한 기능 구현>

}



String 클래스가 제공하는 메소드

length() → 문자열의 길이 반환

concat(”<붙이고 싶은 문자열>”) → 문자열 병합

substring(<index>) → index부터 문자열 끝까지 반환

substring(<index1>, <index2>) → index1부터 index2까지 문자열 반환

static

키워드 static을 사용하면 인스턴스화 하지 않아도 사용할 수 있음 (즉, new 키워드 없이 사용 가능)

클래스 변수

static한 변수, 값을 저장할 수 있는 공간이 하나밖에 없어서 값을 공유함

클래스 이름을 직접 사용하는 것이 가능

ex) 클래스이름.클래스변수명

열거형(enum)

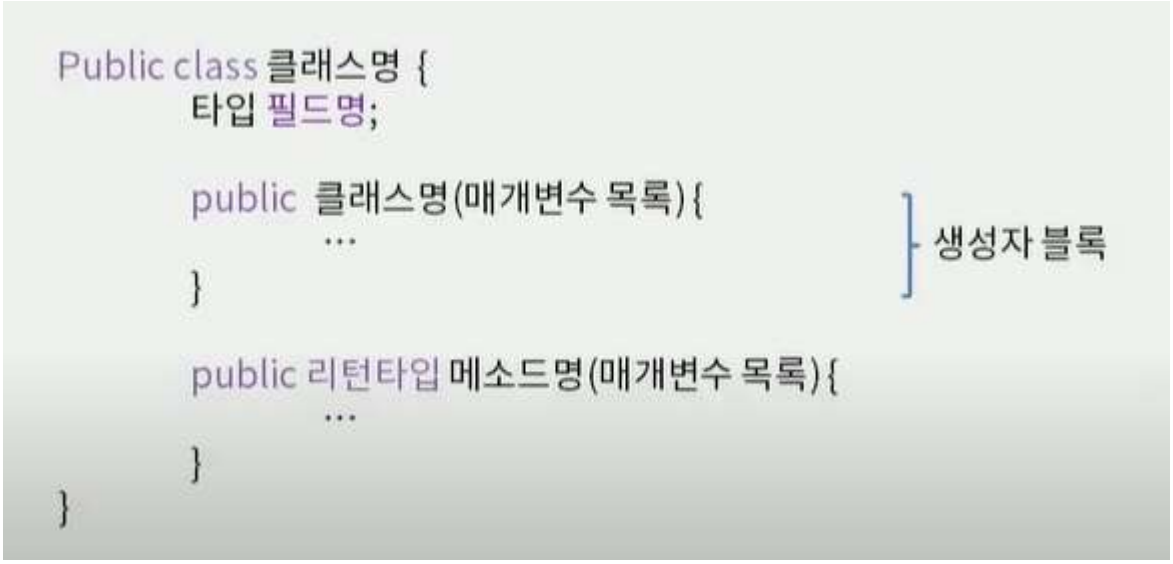
enum Gender{

MALE, FEMALE;

}

Gender.MALE 이런 식으로 접근

생성자



생성자를 하나라도 정의하게 되면 기본 생성자는 만들어지지 않음

메소드 오버로딩

매개변수의 수, 타입이 다른 경우 동일한 이름으로 메소드를 여러개 정의할 수 있음

ex)

```
class MyClass2{
       public int plus(int x, int y){
           return x+y;
       }

       public int plus(int x, int y, int z){
           return x + y + z;
       }

       public String plus(String x, String y){
           return x + y;
```

```
        }
    }
```

생성자 오버로딩

메소드와 마찬가지로 매개변수의 수와 타입이 다르다면 여러개의 생성자를 선언할 수 있음

기본 생성자로 객체를 만들고 싶다면 기본 생성자를 직접 정의해주기

패키지

관련 있는 클래스들끼리 묶음

import

다른 패키지에 들어있는 클래스를 사용하기 위한 키워드

상속

부모 클래스가 자식 클래스에게 물려주는 것

접근제한자

클래스 내에서 멤버의 접근을 제한하는 역할

캡슐화

관련된 내용을 모아서 가지고 있는 것

- public
  - 어떤 클래스든 접근할 수 있다는 것을 의미
- protected
  - 자기 자신, 같은 패키지, 서로 다른 패키지다 하더라도 상속받은 자식 클래스에서는 접근할수 있다는 것을 의미
- private
  - 자기 자신만 접근할 수 있다는 것을 의미
- 접근제한자를 적지 않으면 default접근 지정자
  - 자기자신과 같은 패키지에서만 접근할 수 있다는 것을 의미

public > protected > default > private

추상클래스

구체적이지 않은 클래스

메소드가 하나라도 추상메소드인 경우 해당 클래스는 추상 클래스임

추상클래스는 부모 역할만 가능 직접적으로 객체 생성 불가능

super

클래스가 인스턴스화 될 때 생성자가 실행되면서 객체 초기화

그 때 부모의 생성자부터 실행 후 자신의 생성자 실행

오버라이딩

부모가 가지고 있는 메소드와 똑같은 모양의 메소드를 자식이 가지는 것

메소드를 재정의하는 것

클래스의 형변환

부모타입으로 자식객체를 참조하게 되면 부모가 가지고 있는 메소드만 사용할 수 있음.

자식객체가 가지고 있는 메소드나 속성을 사용하고 싶다면 형변환 해야 함.

```
public class BusExam{
        public static void main(String args[]){
            Car car = new Bus();
            car.run();
//car.ppangppang(); // 컴파일 오류 발생Bus bus = (Bus)car;//
부모타입을 자식타입으로 형변환bus.run();
            bus.ppangppang();
        }
    }
```

인터페이스

서로 관계가 없는 물체들이 상호 작용을 하기 위해서 사용하는 장치나 시스템

인터페이스 자체가 객체를 생성하지는 못함

인터페이스 내에서 선언한 변수는 상수 역할 → 값을 바꿀 수 없으므로 바꾸려면 클래스에서 선언하길

default

이 키워드를 이용하면 인터페이스 내에서 메소드 구현이 가능

static

인터페이스에 static 메소드를 선언하면 참조할 때 인터페이스명으로 참조해야함

내부클래스

익명클래스

예외처리

try-catch-finally

try → 예외가 발생할 수 있는 상황

catch → 예외가 발생하면 수행

finally → 예외 발생 여부에 상관없이 반드시 실행

catch블록 여러개 사용 가능