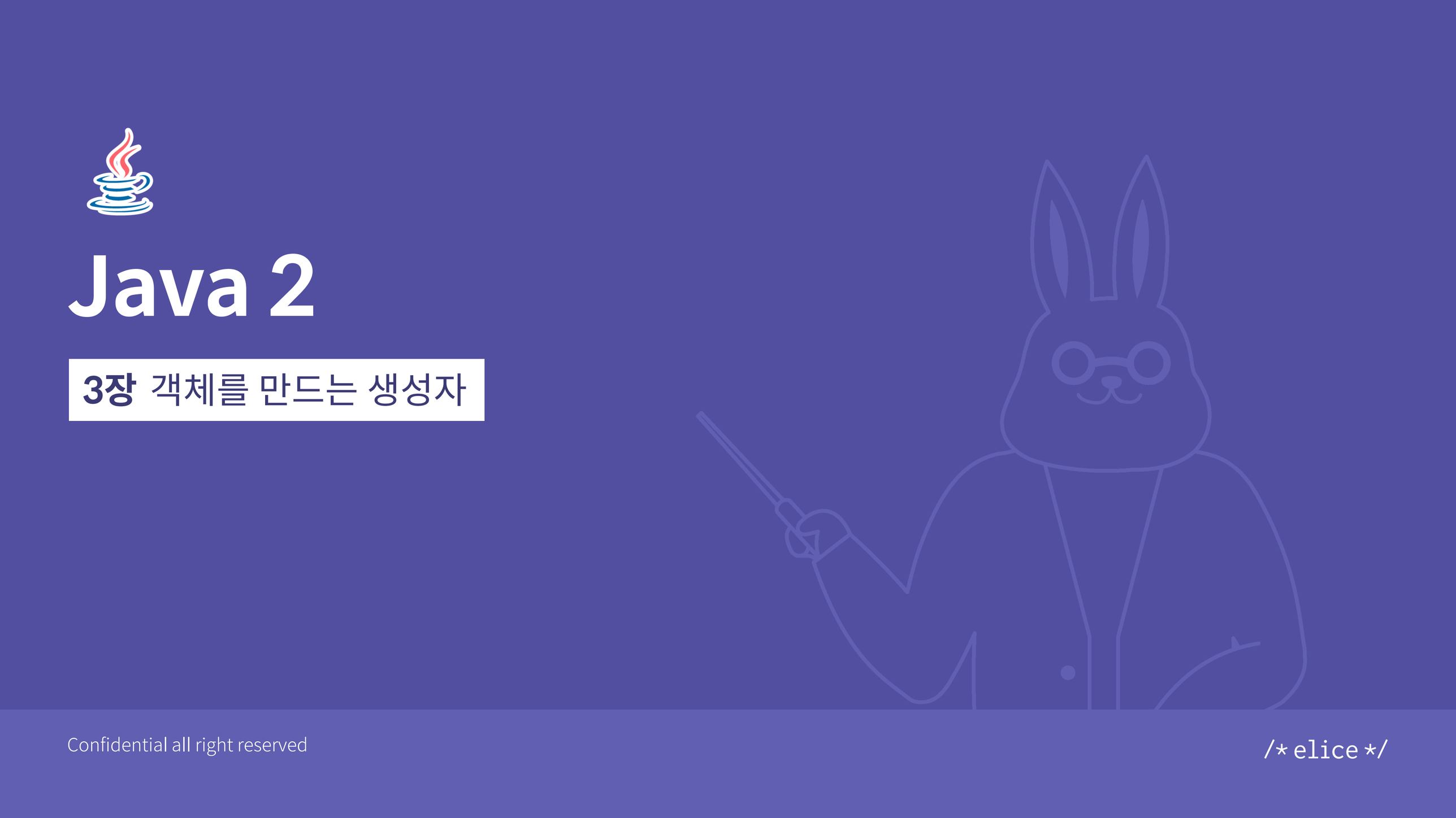


Java 2

3장 객체를 만드는 생성자

/* elice */

Contents

01

생성자

01 생성자

생성자

인스턴스를 생성할 때 **new**와 함께 사용

Example

```
Student s1 = new Student(); 생성자

s1.name = “Elice”;
s1.score = 100;
s1.study();
```

/* elice */

01 생성자

생성자

생성자 이름은 클래스 이름과 동일

Example

```
Student s1 = new Student();  생성자

s1.name = “Elice”;
s1.score = 100;
s1.study();
```

/* elice */

생성자

생성자는 멤버 변수, 메소드와 마찬가지로 클래스에서 정의

Example

```
class Student {
    int number;     //학번
    int score;      //시험 점수
    String name;    //이름
    public Student() {}          <-- 생성자
    void study(){
        System.out.println(“Studying”);  //공부하기
    }
}
```

01 생성자

기본 생성자

생성자를 선언하지 않아도 자동으로 만들어진다.

Example

```
class Student {
    int number;     //학번
    int score;      //시험 점수
    String name;    //이름
    // public Student() {}
    void study(){
        System.out.println(“Studying”);  //공부하기
    }
}
```

기본 생성자

생성자 구현

생성자에 인자가 필요하면 **명시적으로** 구현할 수 있음(예, 학번)

인스턴스가 **생성되는 시점에** 수행할 작업이
있는 경우 생성자를 활용

01 생성자

생성자 구현

생성자에 인자를 넘겨 멤버 변수를 **초기화**하는 등 특정 작업을 수행

Example

```
class Student {
    String name;

    public Student(String pname) {
        name = pname;
        System.out.println("Student 인스턴스 생성");
    }
}
```

this

인스턴스 “자기 자신”을 나타내는 예약어 : **this**

01 생성자

this

멤버 변수 name을 **매개변수 name**으로 초기화하면?

Example

```
class Student {
    String name;

    public Student(String name) {
        name = name;    // 둘 다 매개변수 name 으로 인식
  }
}
```

/* elice */

01 생성자

this

this를 이용한 멤버 변수 초기화

Example

```
class Student {
    String name;

    public Student(String name) {
        this.name = name;
        //this.name은 멤버변수, name은 매개변수를 각각 가리킨다.
    }
}
```

/* elice */

01 생성자

this

매개변수와 멤버 변수의 이름이 다르면
this를 쓰지 않아도 상관없다.

Example

```
class Student {
    String name;

    public Student(String pname) {
        name = pname;
    }
}
```

/* elice */

this

복잡한 소스코드에서
변수가 많아지면
관리하기 힘든 경우가 발생한다.

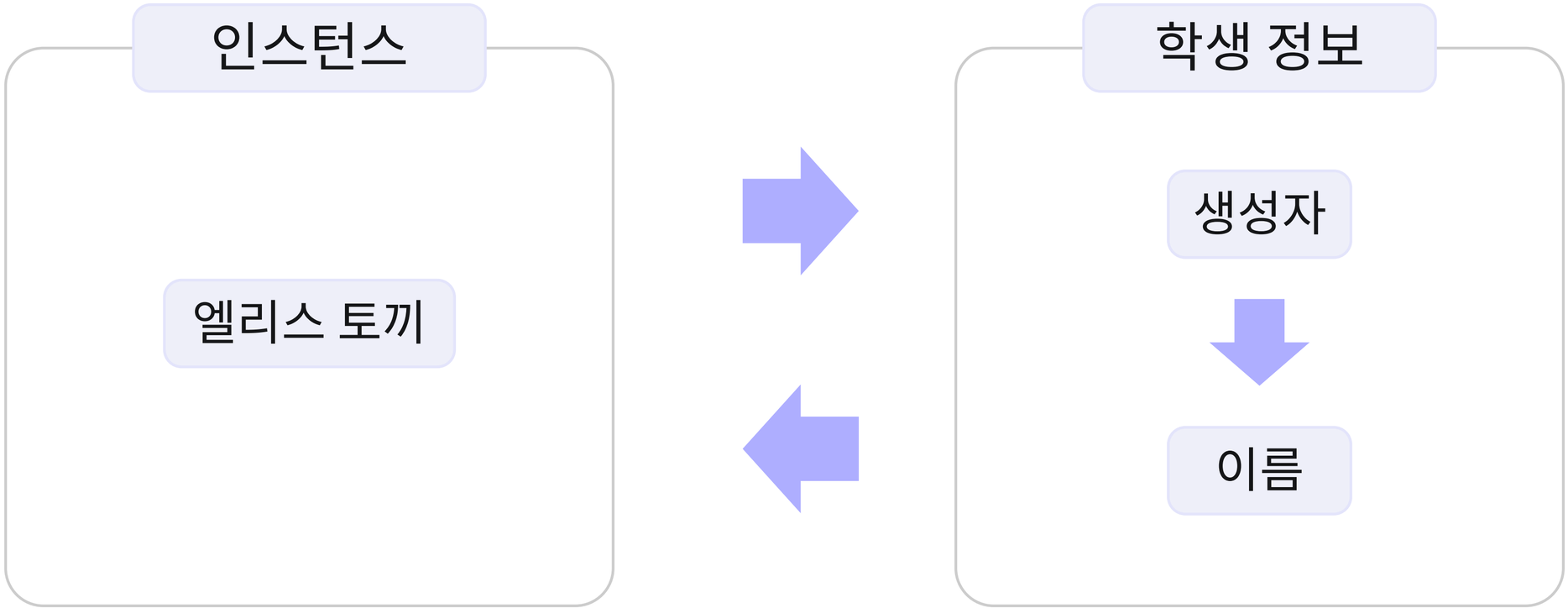
this

멤버 변수와 매개 변수에 같은 이름을 부여하고

this로 구분하면 유용하다.

[실습1] 생성자 구현하기

생성자를 구현해 멤버 변수를 초기화 해봅시다!



02

생성자 오버로드

02 생성자 오버로드

생성자 구현 문제

아래 코드에서 오류가 발생

Example

```
class Student {
    String name;

    public Student(String name) {
        this.name = name;
    }
}

...

Student s = new Student();  //오류 발생!
```

/* elice */

생성자 오버로드

생성자를 여러 개 선언할 수도 있다!

생성자 오버로드

생성자를 여러 개 선언할 수도 있다!

-> 생성자 오버로드

생성자 오버로드

기본 생성자를 직접 추가

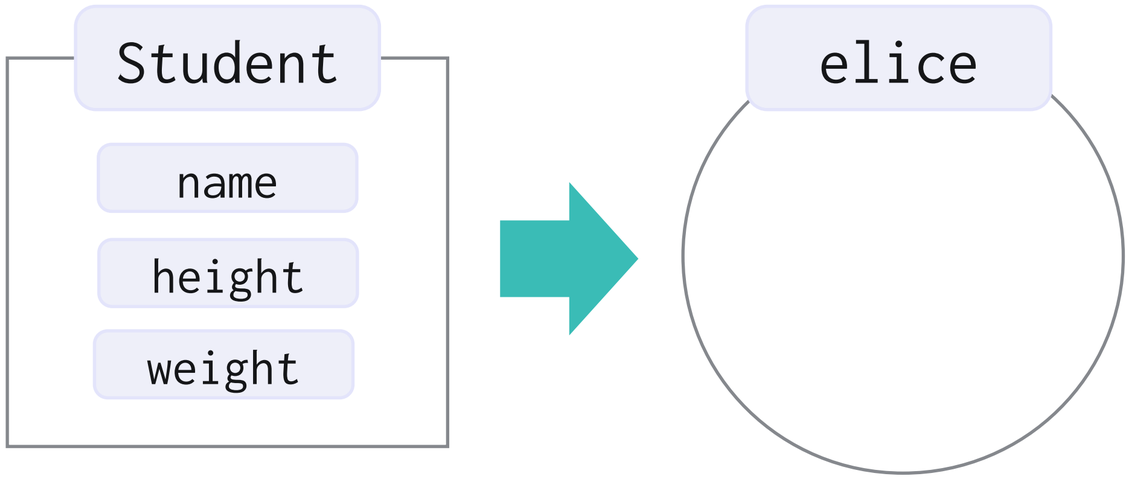
Example

```
class Student {
    String name;

    public Student() { }

    public Student(String name) {
        this.name = name;
    }
}
Student s = new Student();  //오류 해결
```

02 생성자 오버로드

멤버변수 초기화 1

```
Student elice = new Student();
```




```
/* elice */
```

02 생성자 오버로드

멤버변수 초기화 1

Example

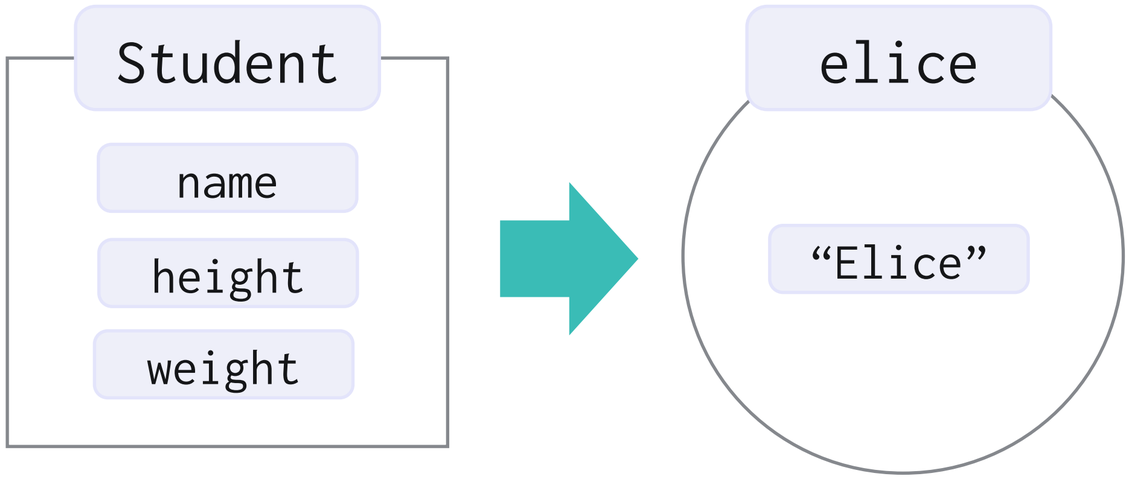
```
class Student {
    String name;    // 이름
    int height;     // 키
    int weight;     // 몸무게

    public Student() {}
    public Student(String name) {
        this.name = name;
    }
```

```
public Student(String name, int height, int weight) {
        this.name = name;
        this.height = height;
        this.weight = weight;
    }
}
Student s = new Student();
```

02 생성자 오버로드

멤버변수 초기화 2

```
Student elice = new Student(“Elice”);
```




```
/* elice */
```

02 생성자 오버로드

멤버변수 초기화 2

Example

```
class Student {
    String name;    // 이름
    int height;     // 키
    int weight;     // 몸무게

    public Student () {}
    public (String namStudent e) {
        this.name = name;
    }
```

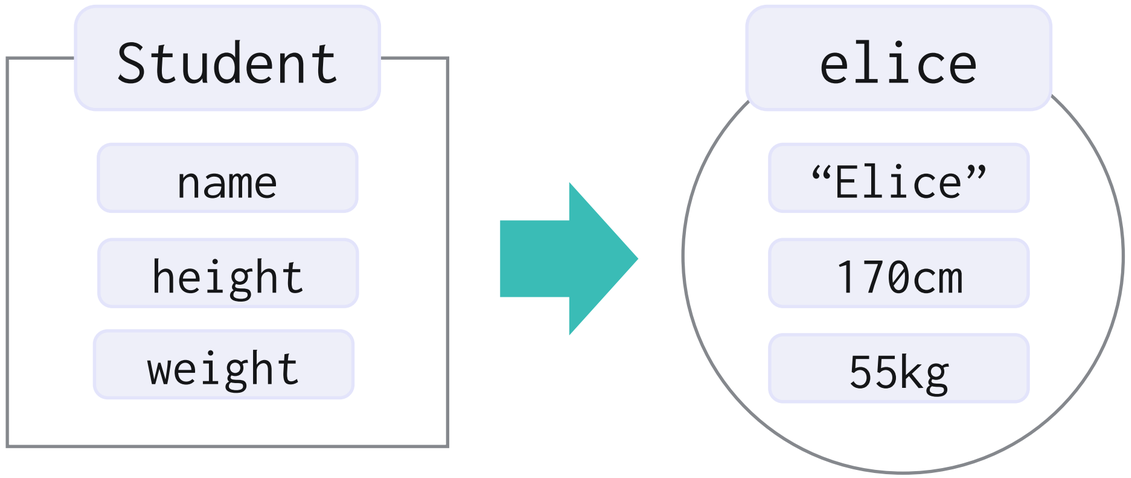
```
public Student(String name, int height, int weight) {
        this.name = name;
        this.height = height;
        this.weight = weight;
    }
}
Student elice = new Student(“Elice”);
```

/* elice */

02 생성자 오버로드

멤버변수 초기화 3

```
Student elice = new Student(“Elice”,170,55);
```




```
/* elice */
```

02 생성자 오버로드

멤버변수 초기화 3

Example

```
class Student {
    String name;    // 이름
    int height;     // 키
    int weight;     // 몸무게

    public Student() {}
    public Student(String name) {
        this.name = name;
    }
```

```
    public Student(String name, int height, int weight) {
        this.name = name;
        this.height = height;
        this.weight = weight;
    }
}
Student elice = new Student("Elice", 170, 55);
```

/* elice */

생성자 오버로드

이와 같이 필요에 따라

생성자를 여러 개 만들 수 있다.

02 생성자 오버로드

앞선 실습 분석

기본 생성자가 불필요한 경우

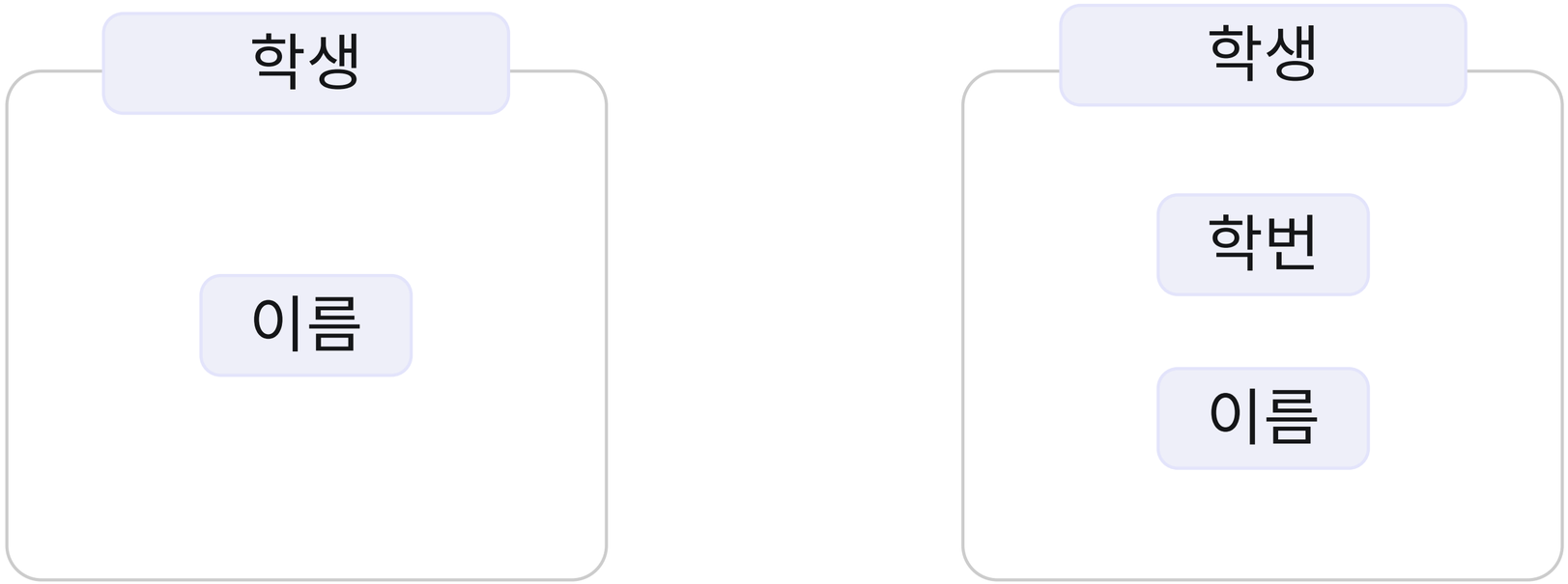
Example

```
class Student {
    String name;    //이름
    int number;     //학번

    public Student(int name) {
        this.name = name;
    }
    public Student(String name, int number) {
        this.name = name;
        this.number = number;
    }
}
```

/* elice */

[실습2] 생성자 오버로드

같은 이름의 생성자를 만들어봅시다!



03

static 변수

03 static 변수

static 키워드

static 키워드는 변수와 메소드 앞에 붙일 수 있다.

Example

```
public class Student {
    static int number;
    static void study() {System.out.println(“공부”);}
}
```

/* elice */

static 변수

모양과 숫자는 플레잉 카드마다 다르지만 너비와 높이는 동일

Example

```
public class Card {
    static int width = 10;      //카드의 너비
    static int height = 16;     //카드의 높이
    String shape;               //카드의 모양
    int number;                 //카드의 숫자(1~13)
}
```

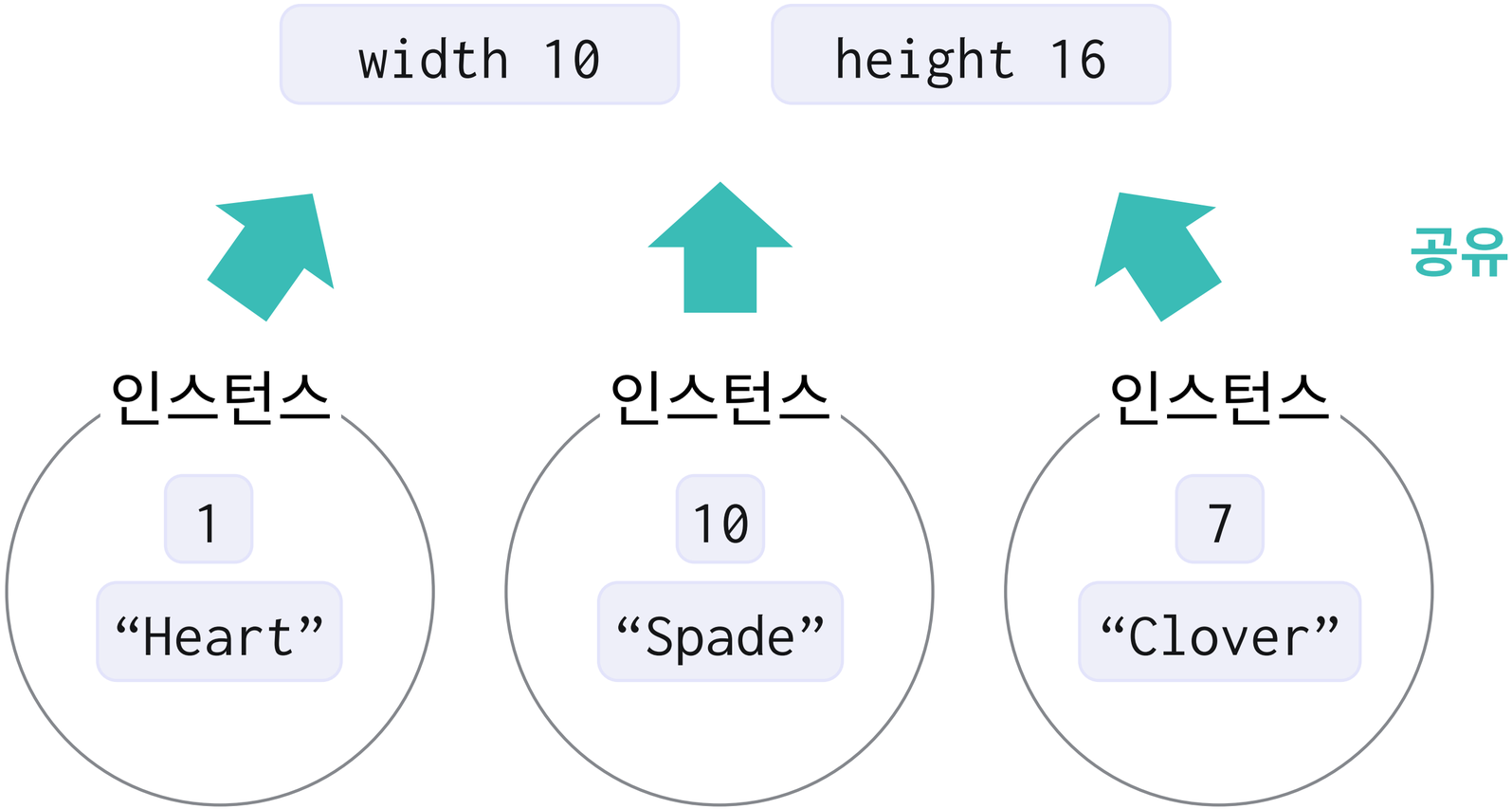
static 변수

프로그램이 실행될 때

인스턴스 생성과 상관없이 **먼저 생성**된다.

03 static 변수

static 변수



03 static 변수

static 변수

static 변수의 값이 **공유**됨을 확인

Example

```
Card card1 = new Card();
Card card2 = new Card();

System.out.println(card1.width);    //10
card1.width = 20;
System.out.println(card2.width);    //20
Card.width = 35;
System.out.println(card2.width);    //35
```

static 변수

프로그램이 시작될 때

인스턴스보다 **먼저 생성**되어 존재한다.

static 변수

static 변수는 인스턴스와 완전히 별개이며
인스턴스가 생성되지 않아도 사용할 수 있다.

static 변수

클래스에 속해 **한 번만** 생성되는 변수

이를 여러 인스턴스가 **공유**

클래스 변수라고도 한다.

03 static 변수

static 변수

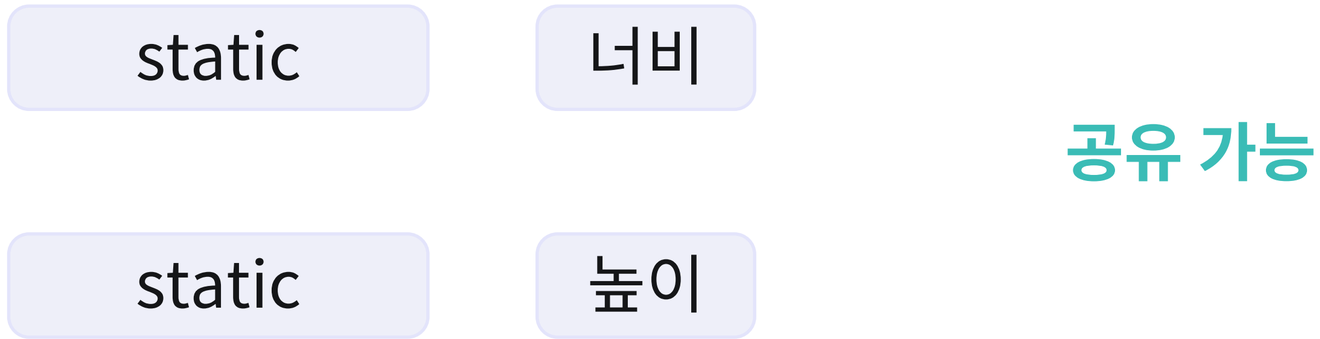
static 변수는 **클래스 이름**으로 참조해야 한다.

Example

```
Card card1 = new Card();
Card card2 = new Card();


card1.width = 20;                    //바람직하지 않음 (컴파일러 경고 발생)
System.out.println(card1.width); //바람직하지 않음 (컴파일러 경고 발생)
Card.width = 35
System.out.println(Card.width);
```

[실습3] static 변수

static 변수를 사용하여 변수의 내용을 공유해봅시다!



04

static 메소드

04 static 메소드

static 메소드

static 변수와 마찬가지로 **static 메소드** 또한 존재

Example

```
public class Main {
    public static void main(String[] args) {
            대표적 예시
    }
}
```

static 메소드

static 변수를 이용하여 값을 반환하는 예시

Example

```
public class Card {
    static int getArea() {
        return width * height;
    }
}
```

04 static 메소드

static 메소드

static 메소드에서 멤버 변수를 **사용할 수 없다.**

Example

```
public class Card {
    static int getCardValue() {
        return shape + number;     //Error!
    }
}
```

static 메소드

static 메소드에서 멤버 변수를 **사용할 수 없다.**

	일반 메소드	static 메소드
멤버 변수	O	X
static 변수	O	O

static 메소드

멤버 변수는 **인스턴스**를 생성 후에 사용 가능

static 메소드는 **인스턴스**를 생성하지 않아도 사용 가능

static 메소드의 사용

인스턴스를 생성할 필요 없이

클래스 이름으로 호출 가능

04 static 메소드

static 메소드의 사용

Math 패키지의 메소드는 모두 static 메소드

static double	**abs**(double a) Returns the absolute value of a double value.
static float	**abs**(float a) Returns the absolute value of a float value.
static int	**abs**(int a) Returns the absolute value of an int value.
static long	**abs**(long a) Returns the absolute value of a long value.
static double	**acos**(double a) Returns the arc cosine of a value; the returned angle is in the range 0.0 through *pi*.
static int	**addExact**(int x, int y) Returns the sum of its arguments, throwing an exception if the result overflows an int.
static long	**addExact**(long x, long y) Returns the sum of its arguments, throwing an exception if the result overflows a long.
static double	**asin**(double a) Returns the arc sine of a value; the returned angle is in the range -*pi*/2 through *pi*/2.
static double	**atan**(double a) Returns the arc tangent of a value; the returned angle is in the range -*pi*/2 through *pi*/2.

/* elice */

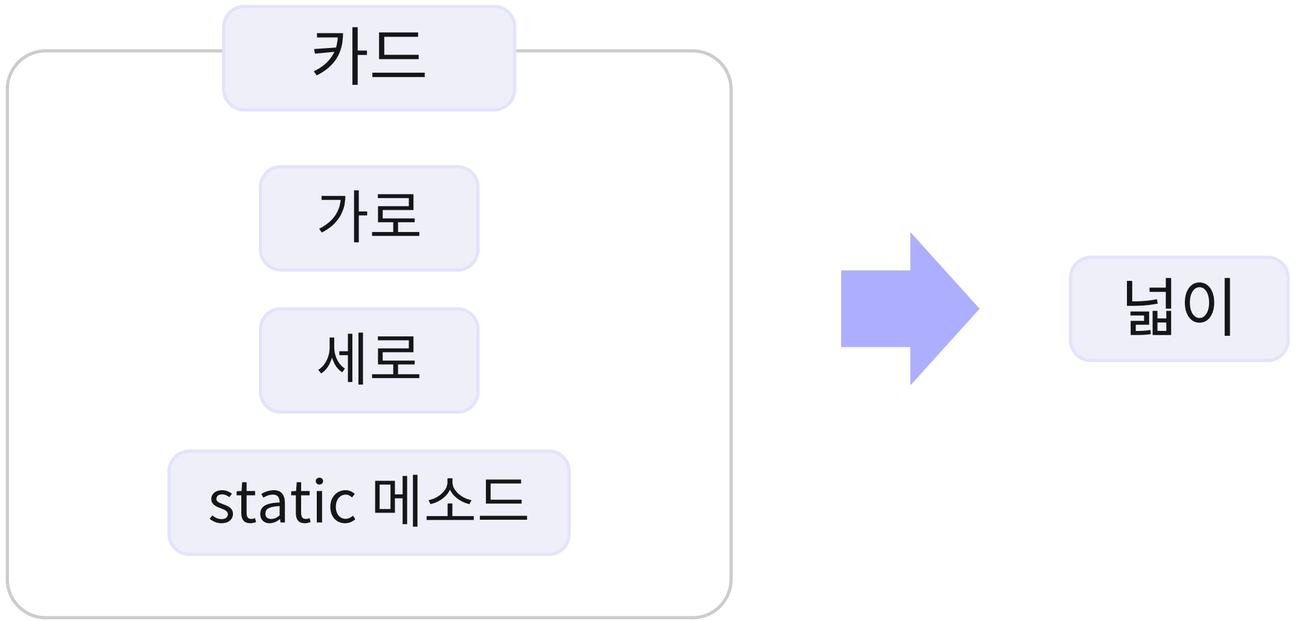
static 메소드의 사용

인스턴스를 만들 필요가 없을 때
static 메소드면 충분하다.

static 메소드만 모여 있는 클래스를 **Util 클래스**라고 부른다.

[실습4] static 메소드

static 메소드를 구현해 봅시다!



05

변수 유효 범위

변수 유효 범위

지금까지 세 가지 변수를 다룬다.

Example

```
public class Card {
    static String width = 10;   //static 변수
    int number;                 //멤버 변수
    int drawCardNumber(){
        int pnumber;            //지역 변수
        pnumber = getRandomInt(1, 13);
         return pnumber;
    }
}
```

지역 변수

메소드 내부에 선언

메소드가 반환될 때 메모리에서 제거된다.

멤버 변수

인스턴스가 생성될 때 생성

인스턴스가 더 이상 쓰이지 않을 때 제거된다.

(Garbage collection)

static 변수

프로그램을 실행할 때

인스턴스 생성과 무관하게 처음부터 생성

프로그램이 끝날 때 삭제

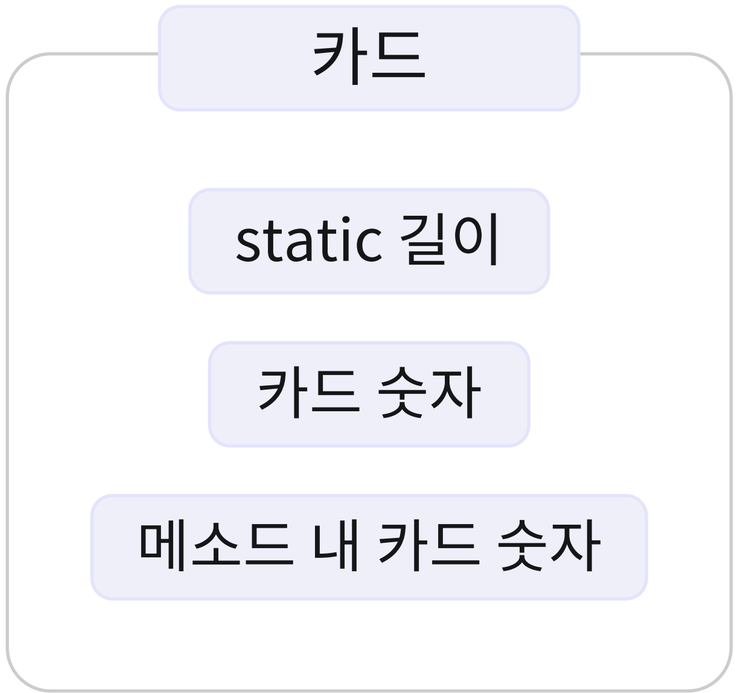
변수 유효 범위

각각의 특성을 이해하고

용도에 따라 사용하는 것이 중요

[실습5] 변수 유효 범위 체크하기

불러올 수 있는 변수를 확인해봅시다!



Credit

/* elice */

코스 매니저
강윤수

콘텐츠 제작자
강윤수

강사
유동환 선생님

디자인
박주연

Contact

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io