클래스(class)의 기본 문법

클래스 개념 및 기본 구조

클래스 개념 및 기본 구조

☞ 클래스의 탄생

1 반 학생들의 성적 처리

3 자바의 시작점

변수 → 배열 → 구조체 → 클래스

2

```
int score1 = 80;
int score2 = 70;
int score3 = 92;
…
int score40 = 68;

double avg = 78.2;
```

```
int[] scores =
 {80, 70, 92, …, 68};

double avg = 78.2;
```

```
struct Score{

  int[] scores =
   {80, 70, 92, …, 68};

  double avg = 78.2;

}
```

- 구조체도 메서드 포함 가능
- 상속 불가

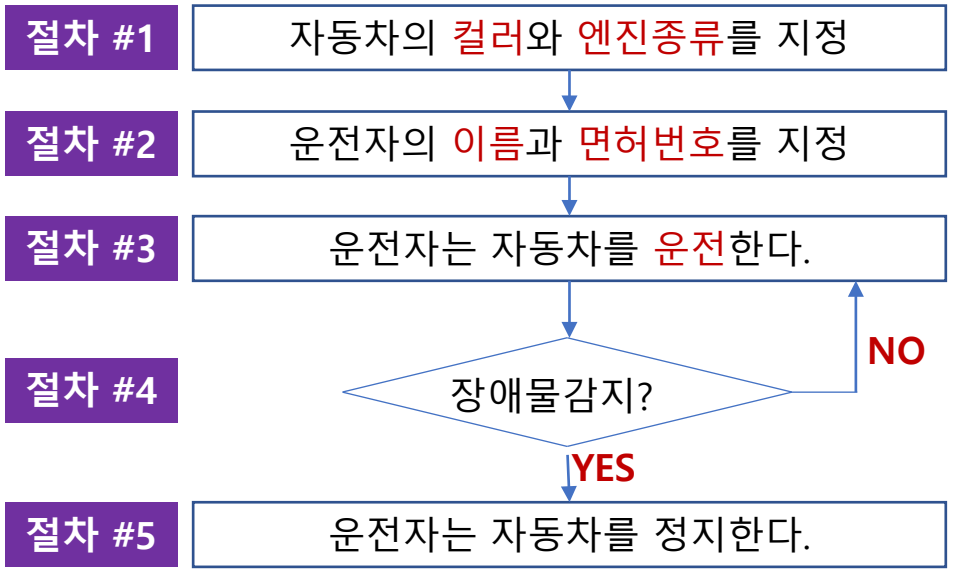
```
class Score{

  int[] scores = {80, 70, …, 68};

  double avg = 78.2;

  void printAvg(){
    System.out.println(avg);
  }
  void printScores(){
    for(int k:scores){
      System.out.println(k);
    }
  }
}
```

클래스 개념 및 기본 구조

☞ 절차지향형 vs. 객체지향형

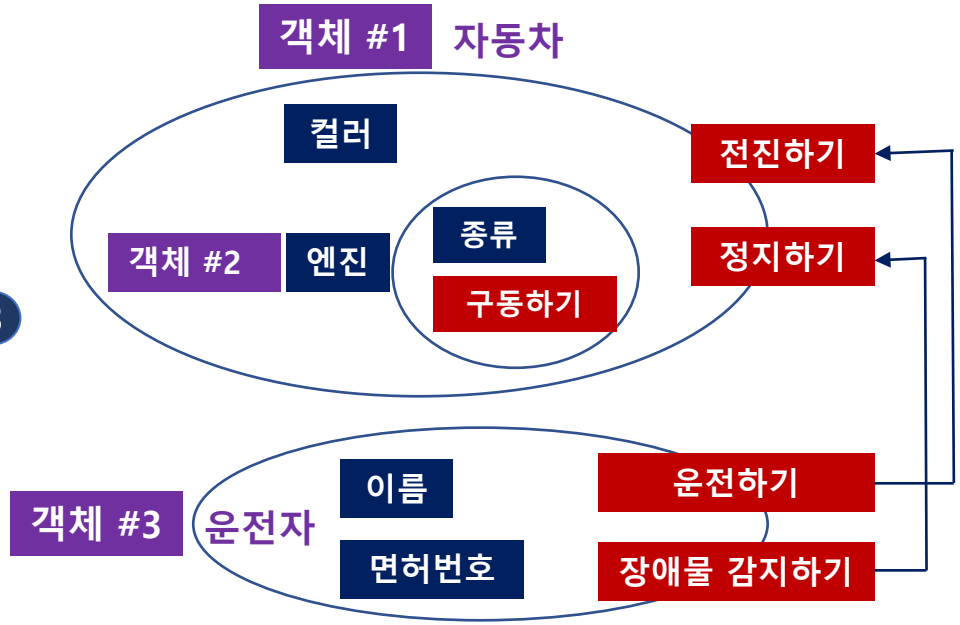
1 - 절차지향형 프로그래밍: 순서에 맞추어 단계적으로 실행하도록 명령어를 나열
- 객체지향형 프로그래밍: 객체를 구성하고 객체단위로 프로그래밍(필드/메서드)

2



절차지향형 (기능중심)

3



객체지향형 (객체중심)

클래스 개념 및 기본 구조

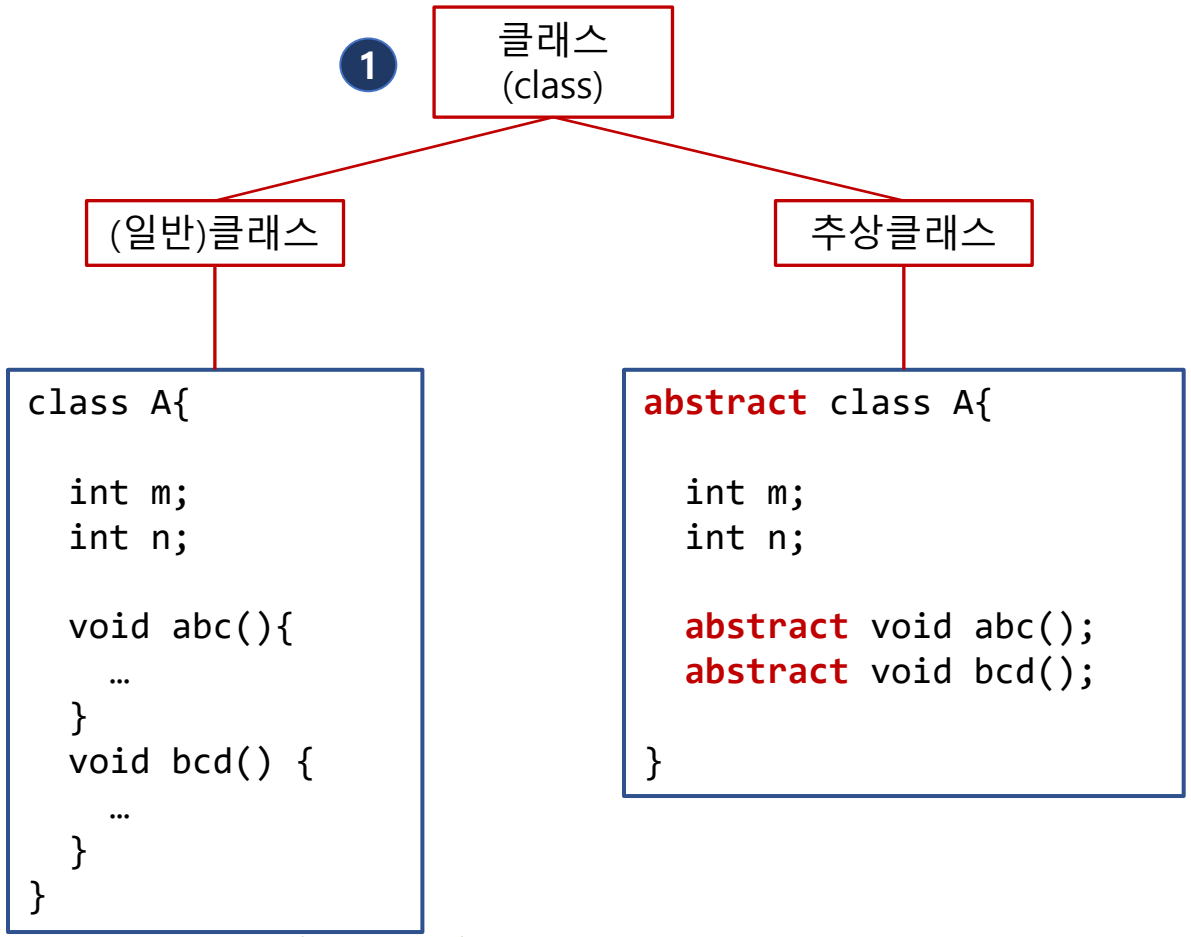
☞ 자바에서 제공하는 객체지향 요소

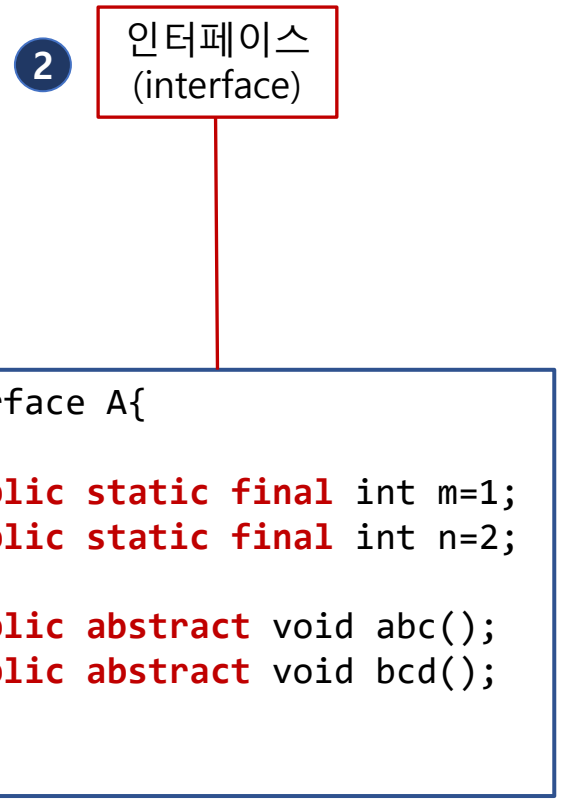
TIP

- 추상메서드(abstract method): 함수의 정의가 미완성된 메서드

TIP

- 추상클래스: 추상(abstract) 메서드를 하나 이상 가지고 있는 클래스
- 인터페이스: 모든 필드는 public static final, 모든 메서드는 public abstract



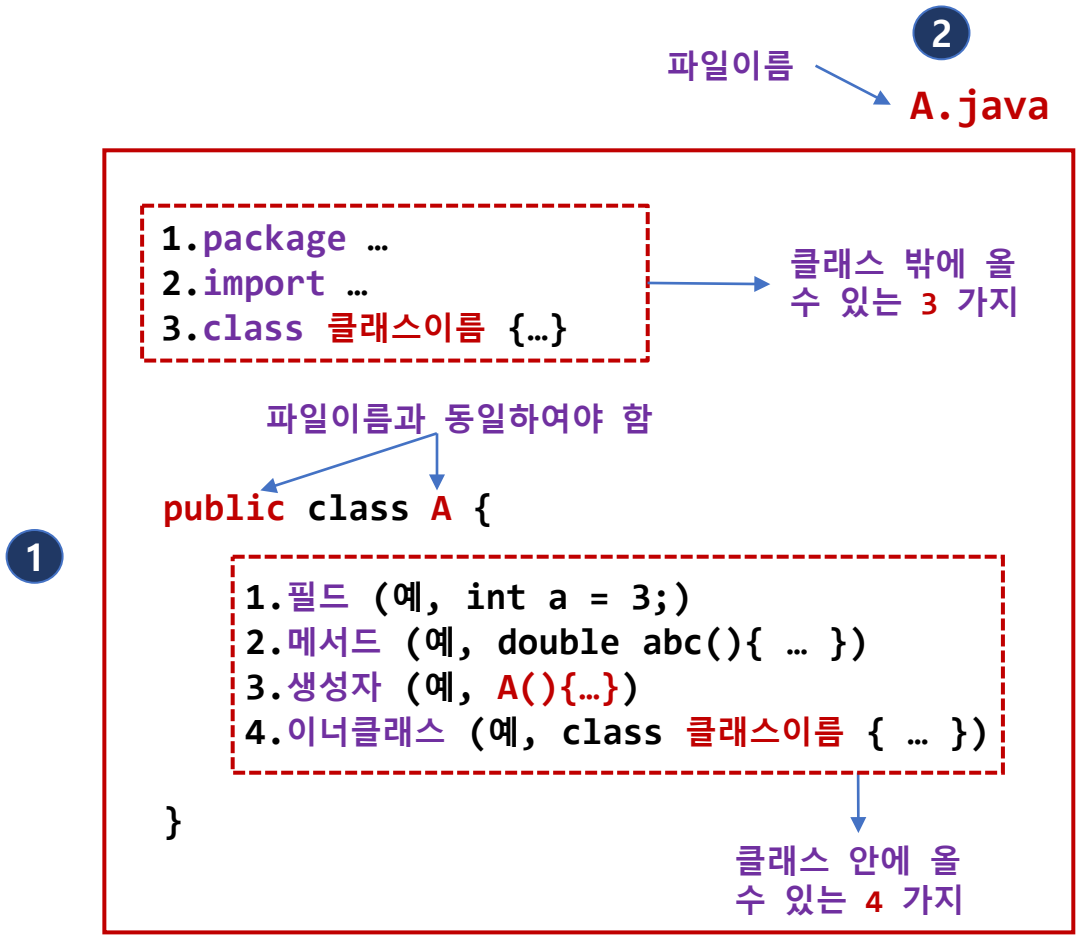


3

클래스의 활용

클래스 개념 및 기본구조

☞ 클래스의 구조



- **클래스 밖에 올 수 있는 3가지** 3

1. package
   - .java 파일의 폴더(패키지) 위치
   - default의 경우 폴더가 생성되지 않음
2. import
   - 다른 폴더(패키지) 위치의 클래스를 참조
3. 외부클래스 (external class)
   - 외부에 포함된 또 다른 클래스
   - public 키워드를 사용할 수 없음

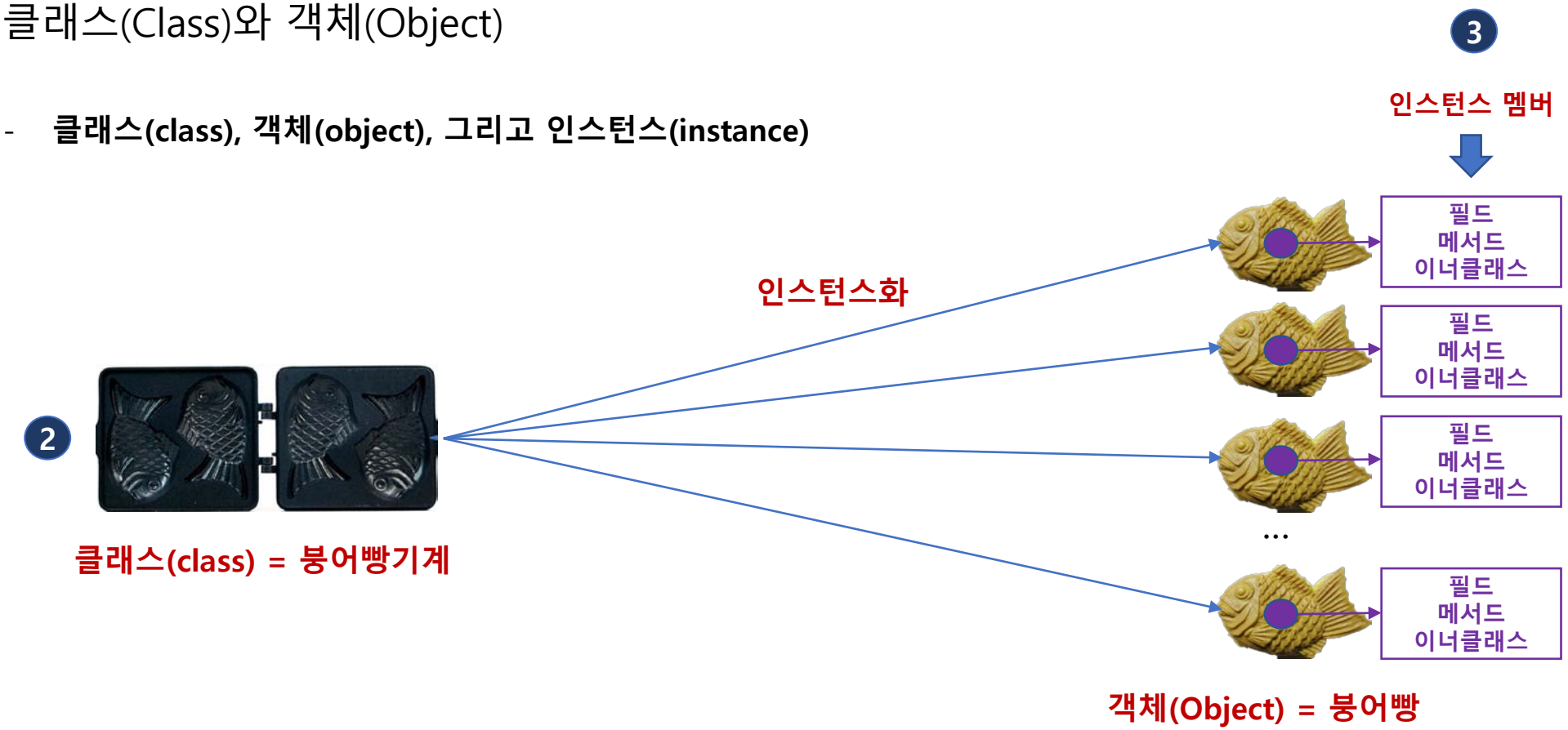
- **클래스 안에 올 수 있는 4가지** 4

1. 필드 **(멤버)**
   - 클래스 특징(속성)을 나타내는 변수 (int age =20)
2. 메서드 **(멤버)**
   - 클래스의 기능 (void working(){…})
   - **리턴타입+메서드이름+()+{ }**로 구성
3. 생성자
   - 객체 생성 기능 (**생성자이름+()+{ }**)
   - 생성자의 이름은 클래스 이름과 동일하여야 함
4. 내부클래스 (inner class) **(멤버)**
   - 클래스 내부 정의된 클래스

클래스 개념 및 기본 구조

☞ 클래스(Class)와 객체(Object)

1 - **클래스(class), 객체(object), 그리고 인스턴스(instance)**



4 객체(Object) = 클래스(Class)의 인스턴스(Instance)

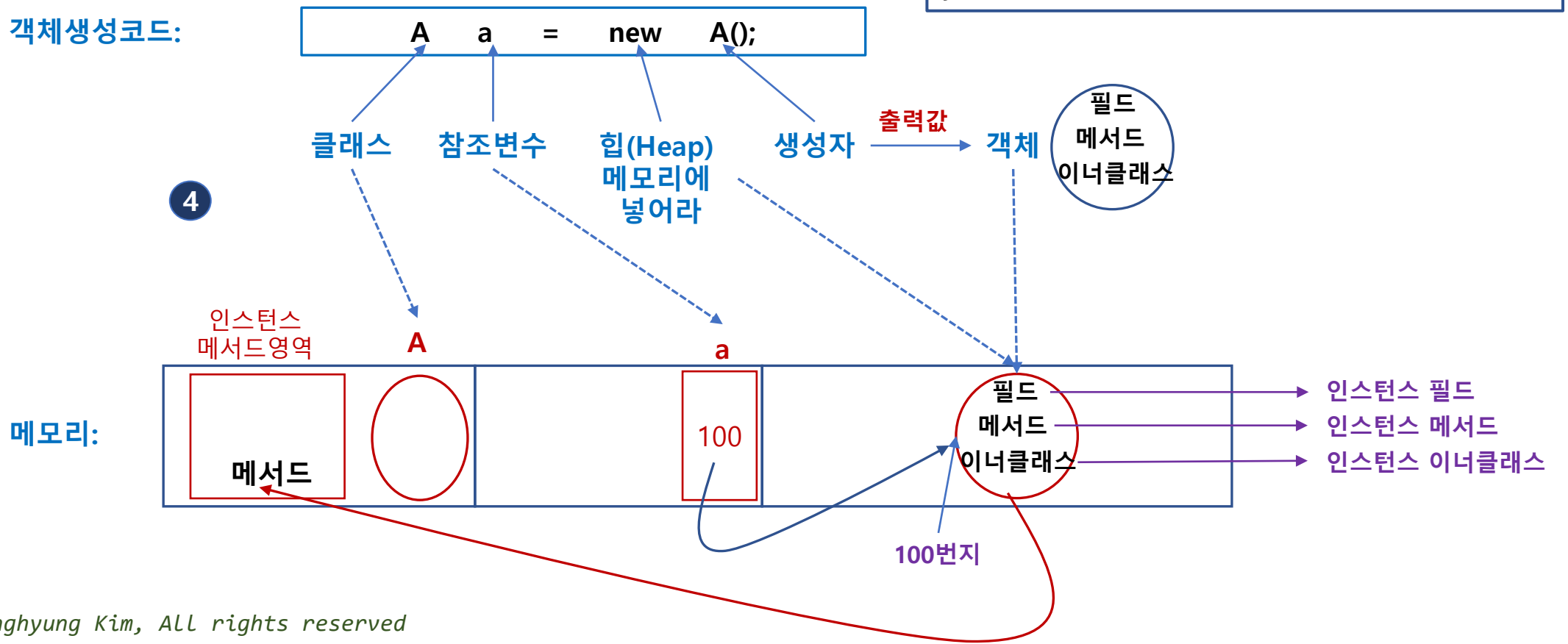
클래스 개념 및 기본 구조

2 **클래스**로부터 **객체를 생성하는 방법**과 객체를 **활용하는 방법**
붕어빵기계에서 **붕어빵을 찍는 방법**과 붕어빵을 **맛있게 먹는 방법**

1 ☞ 객체의 **생성 및 활용**

- **객체의 생성**

3
```
class A {
   필드
   메서드
   이너클래스
}
```



클래스 개념 및 기본 구조

☞ 객체의 **생성 및 활용**

1

- **객체의 생성** 메서드의 공유

2

TIP

모든 생성 객체는
- 동일한 메서드(기능)를 가짐 (즉, 객체를 구분하는 것은 **속성**)
- 따라서 모든 객체는 메서드를 공유

3 ex. **자동차 클래스로 빨간 자동차와 파란 자동차를 하나씩 만드는 경우**

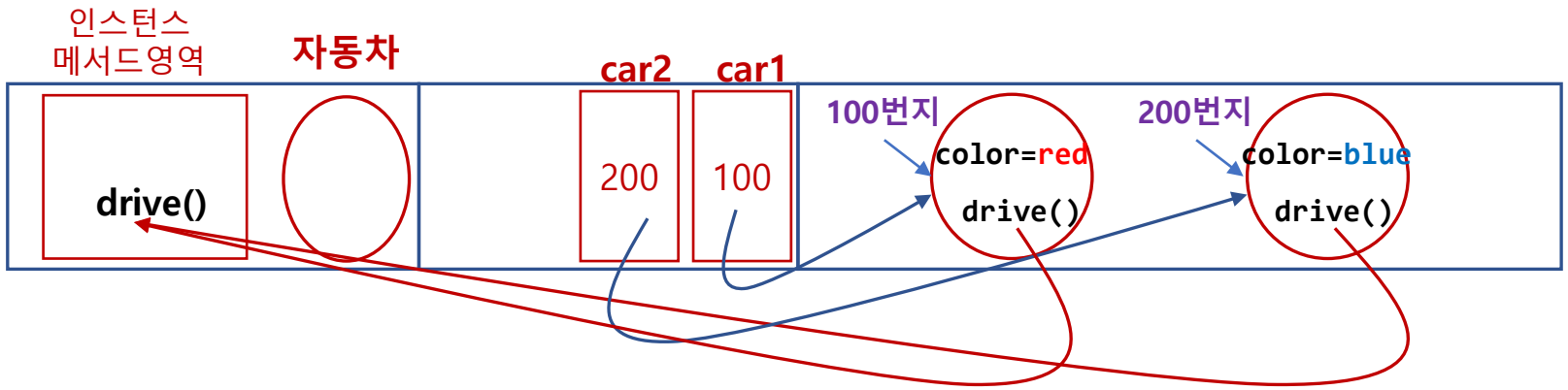
객체생성코드:

```
자동차 car1 = new 자동차(...);
자동차 car2 = new 자동차(...);
```

```
class 자동차 {
  String color;
  void drive(){ … }
  …
}
```

4

메모리:



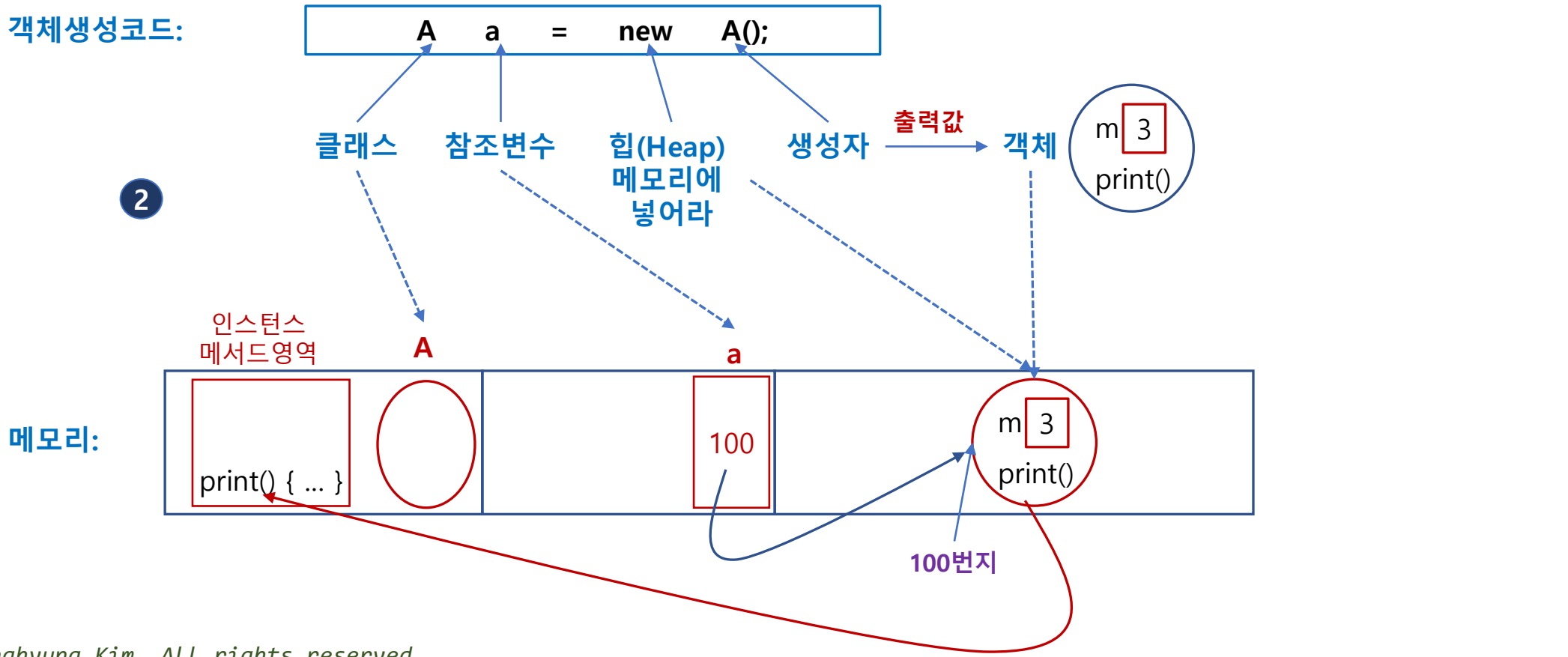
클래스 개념 및 기본 구조

☞ 객체의 **생성** 및 활용

- **객체의 생성**

```
class A{

  int m = 3;

  void print(){
    System.out.println("객체생성 및 활용");
  }

}
```

1



클래스 개념 및 기본 구조

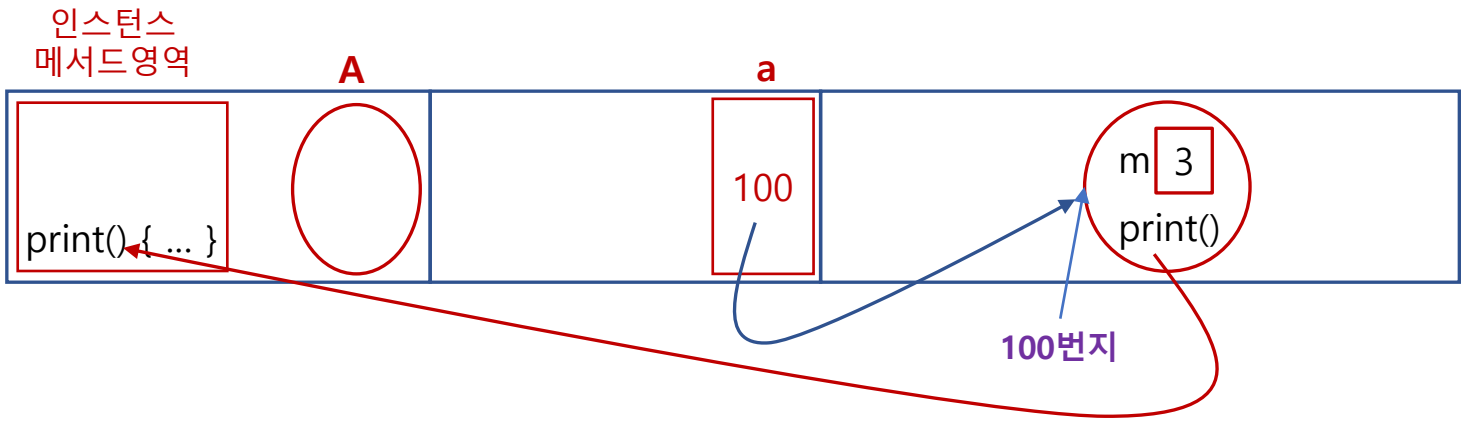
```
class A{

  int m = 3;

  void print(){
    System.out.println("객체생성 및 활용");
  }

}
```

☞ 객체의 생성 및 **활용**

1 - **객체의 활용 (외부 호출)**

참조변수명.필드명 **참조변수명.메서드명**

2 **메모리:**



3 **객체활용코드:**

```
//필드의 활용
System.out.println(a.m);
//메서드의 활용
a.print();
```

```
3

객체생성 및 활용
```

4

TIP

- Java는 Heap 메모리는 직접접근 불가
- 참조변수를 통해서만 접근 가능
- Heap 메모리 접근 방법
  참조변수명.필드명 / **참조변수명.메서드명**

해당번지 위치의
Heap메모리로 이동하라.

The End