클래스와 객체

컴퓨터공학전공
박요한

목차

- 생성자
- 생성자 오버로딩
- this 레퍼런스
- this( )

클래스와 객체

- 클래스
  - ✓객체를 만들어내기 위한 설계도 혹은 **틀**
  - ✓객체의 **속성(state)**과 **행동(behavior)** 포함
- 객체
  - ✓클래스의 모양 그대로 찍어낸 실체
    - 프로그램 실행 중에 생성되는 실체
    - 메모리 공간을 갖는 구체적인 실체
    - 인스턴스(instance)라고도 부름
- 사례
  - ✓클래스: 자동차, 객체: 그랜저, 소나타, K5
  - ✓클래스: 소나타자동차, 객체: 회색 소나타, 흰색 소나타
  - ✓클래스: 사람, 객체: 나, 너, 윗집사람, 아랫집사람
  - ✓클래스: 붕어빵틀, 객체: 구워낸 붕어빵들

예제 4-2 : Rectangle 클래스 만들기 연습

너비와 높이를 입력 받아 사각형의 합을 출력하는 프로그램을 작성하라. 너비(width)와 높이(height) 필드, 그리고 면적 값을 제공하는 getArea() 메소드를 가진 Rectangle 클래스를 만들어 활용하라.

```
import java.util.Scanner;

public class Rectangle {
    int width;
    int height;

    public int getArea() {
        return width*height;
    }

    public static void main(String[] args) {
        Rectangle rect = new Rectangle(); // 객체 생성
        Scanner scanner = new Scanner(System.in);
        System.out.print(">> ");

        rect.width = scanner.nextInt();
        rect.height = scanner.nextInt();

        System.out.println("사각형의 면적은 " + rect.getArea());

        scanner.close();
    }
}
```

생성자 개념

- 생성자
  - ✓객체가 생성될 때 초기화를 위해 실행되는 메소드

> 객체 초기화란?
>
> 필드를 초기화하거나, 메소드를 호출해서 객체를 사용할 준비를 하는 것

기본 생성자

- 기본 생성자(default constructor)
  - ✓매개 변수 없고 아무 작업 없이 단순 리턴하는 생성자
  - ✓디폴트 생성자라고도 부름
- 클래스에 생성자가 하나도 선언되지 않은 경우, 컴파일러에 의해 자동으로 삽입

```
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(3);
    }
}
```

개발자가 작성한 코드
이 코드에는 생성자가 없지만
컴파일 오류가 생기지 않음

이유

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

    public Circle() {}

    public static void main(String [] args){
        Circle pizza = new Circle();
        pizza.set(3);
    }
}
```

컴파일러에 의해 자동 삽입된 기본 생성자

컴파일러가 자동으로 기본 생성자 삽입

생성자의 특징

- 생성자의 특징
  - ✓생성자는 메소드
  - ✓생성자 이름은 클래스 이름과 반드시 동일
  - ✓생성자 여러 개작성 가능 (오버로딩)
  - ✓생성자는 new를 통해 객체를 생성할 때, 객체당 한 번 호출
  - ✓생성자는 리턴 타입을 지정할 수 없음
  - ✓생성자의 목적은 객체 초기화
  - ✓생성자는 객체가 생성될 때 반드시 호출됨.
    - 그러므로 하나 이상 선언되어야 함
      - 개발자가 생성자를 작성하지 않았으면 컴파일러가 자동으로 기본 생성자 삽입

기본 생성자가 자동 생성되지 않는 경우

- 개발자가 클래스에 생성자가 하나라도 작성한 경우
  - ✓ 기본 생성자 자동 삽입되지 않음

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


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



   }
}
```

예제 4-3:두 개의 생성자를 가진 Circle 클래스

다음 코드는 2개의 생성자를 가진 Circle 클래스이다. 실행 결과는 무엇인가?

```
public class Circle {
    int radius;
    String name;

    public Circle() { // 매개 변수 없는 생성자
        radius = 1; name = ""; // radius의 초기값은 1
    }
    public Circle(int r) {
        radius = r;
    }
    public Circle(int r, String n) { // 매개 변수를 가진 생성자
        radius = r; name = n;
    }
    public double getArea() {
        return 3.14*radius*radius;
    }

    public static void main(String[] args) {
        Circle pizza = new Circle(10, "자바피자"); // Circle 객체 생성, 반지름 10

        double area = pizza.getArea();
        System.out.println(pizza.name + "의 면적은 " + area);

        Circle donut = new Circle(); // Circle 객체 생성, 반지름 1
        donut.name = "도넛피자";
        area = donut.getArea();
        System.out.println(donut.name + "의 면적은 " + area);
    }
}
```

//연습문제 4-2

```
import java.util.Scanner;

public class Grade {
```

멤버

생성자

average 매소드

```
    public static void main(String [] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("수학, 과학, 영어 순으로 3개의 점수 입력>>");
        int math = scanner.nextInt();
        int science = scanner.nextInt();
        int english = scanner.nextInt();
        Grade me = new Grade(math, science, english);
        System.out.println("평균은 " + me.average);
        scanner.close();
    }
}
```

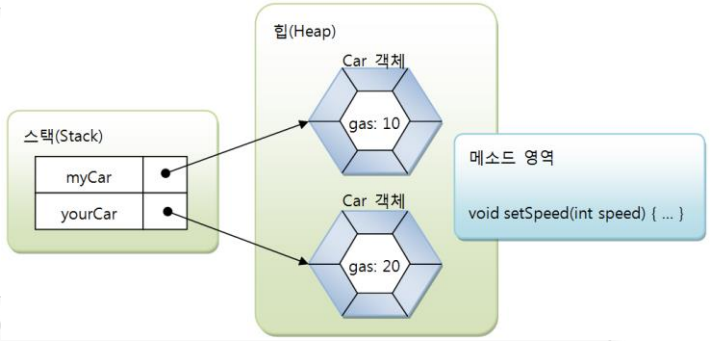
//연습문제 4-2

```
import java.util.Scanner;

public class Grade {
   private int math;
   private int science;
   private int english;

    public Grade(int math, int science, int english) {
       math = math;
       science = science;
       english = english;
   }

   public int average() {
       return (math+science+english)/3;
   }

   public static void main(String [] args) {
       Scanner scanner = new Scanner(System.in);
       System.out.print("수학, 과학, 영어 순으로 3개의 점수 입력>>");
       int math = scanner.nextInt();
       int science = scanner.nextInt();
       int english = scanner.nextInt();
       Grade me = new Grade(math, science, english);
       System.out.println("평균은 " + me.average);
       scanner.close();
   }
}
```

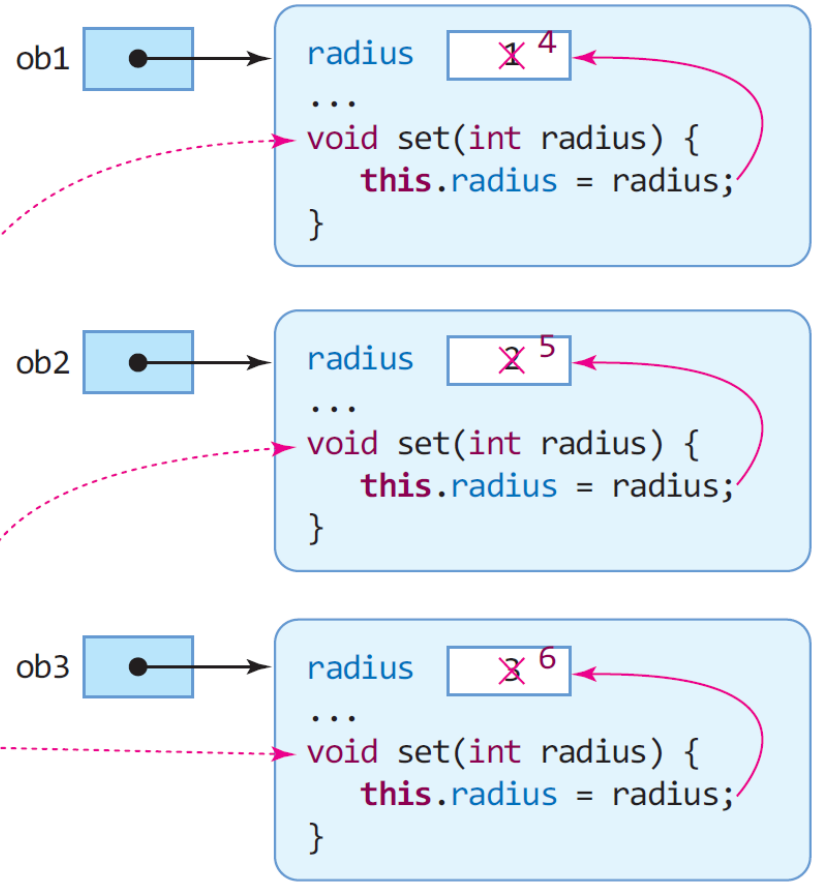
this의 필요성

- 객체의 멤버 변수와 메소드 변수의 이름이 같은 경우
- 다른 메소드 호출 시 객체 자신의 레퍼런스를 전달할 때
- 메소드가 객체 자신의 레퍼런스를 반환할 때

객체 속에서의 this




```
public class Circle {
   int radius;
   public Circle(int radius) {
      this.radius = radius;
   }
   public void set(int radius) {
      this.radius = radius;
   }

   public static void main(String[] args) {
      Circle ob1 = new Circle(1);
      Circle ob2 = new Circle(2);
      Circle ob3 = new Circle(3);

      ob1.set(4);
      ob2.set(5);
      ob3.set(6);
   }
}
```



this()로 다른 생성자 호출

- this()
  - ✓생성자 내에서만 사용 가능
  - ✓클래스 내의 다른 생성자 호출
  - ✓반드시 생성자 코드의 제일 처음에 수행

예제 4-5 this()로 다른 생성자 호출

예제 4-4에서 작성한 Book 클래스의 생성자를 this()를 이용하여 수정하라.

```
public class Book {
    String title;
    String author;
    void show() { System.out.println(title + " " + author); }

    public Book() {
        this("", "");
        System.out.println("생성자 호출됨");
    }

    public Book(String title) {
        this.title=title; this.author="작자미상";
        this(title, "작자미상");
    }

    public Book(String title, String author) {
        this.title = title; this.author = author;
    }
    public static void main(String [] args) {
        Book littlePrince = new Book("어린왕자", "생텍쥐페리");
        Book loveStory = new Book("춘향전");
        Book emptyBook = new Book();
        loveStory.show();
    }
}
```

title = " 춘향전"
author = "작자미상"

생성자 호출됨
춘향전 작자미상