Come to HUFS Meet the World!

객체지향프로그래밍

24.10.02 실습

- 담당 교수 : 전 병 환 교수님
- 조교 : 윤 종 엄
- 조교 메일 : juyoon@hufs.ac.kr

9월 25일 실습 정답 코드

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

- 실습 1: 반복문과 배열

```java
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int [] unit = {50000, 10000, 1000, 100, 10, 1};

        Scanner scanner = new Scanner(System.in);
        int amount = scanner.nextInt();
        int [] num = new int[6];

        for (int i=0; i<6;i++){
            num[i] = amount/unit[i];
            amount = amount%unit[i];
        }

        System.out.printf("50000원 짜리 %d개%n", num[0]);
        System.out.printf("10000원 짜리 %d개%n", num[1]);
        System.out.printf("1000원 짜리 %d개%n", num[2]);
        System.out.printf("100원 짜리 %d개%n", num[3]);
        System.out.printf("10원 짜리 %d개%n", num[4]);
        System.out.printf("1원 짜리 %d개", num[5]);
    }
}
```

9월 25일 실습 정답 코드

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

- 실습 2: 비정방형 배열

```java
import java.io.*;
import java.util.*;

class Main {
    public static void main (String[] args) {

        int intArray[][] = new int[5][];
        intArray[0] = new int[5];
        intArray[1] = new int[3];
        intArray[2] = new int[4];
        intArray[3] = new int[1];
        intArray[4] = new int[2];

        for (int i = 0; i < intArray.length; i++)
            for (int j = 0; j < intArray[i].length; j++)
                intArray[i][j] = (i+1)*10 + j;

        for (int i = 0; i < intArray.length; i++) {
            for (int j = 0; j < intArray[i].length; j++)
                System.out.print(intArray[i][j]+" ");
            System.out.println();
        }
    }
}
```

9월 25일 실습 정답 코드

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

- 실습 3: 예외처리(try-catch-finally)

```java
public static void main(String[] args) {
    // TODO Auto-generated method stub
    Scanner scanner = new Scanner(System.in);

    int num1 = 0; int num2 = 0; int sum = 0; int sub = 0; int mul = 0; int div = 0;

    while (true) {
        num1 = scanner.nextInt();
        num2 = scanner.nextInt();

        sum = sum(num1, num2);
        sub = sub(num1, num2);
        mul = mul(num1, num2);
        try {
            div = div(num1, num2);
            break;
        } catch (ArithmeticException e) {
            System.out.println("0으로 나눌 수 없습니다! 다시 입력하세요.");
        }
    }

    System.out.println(num1+" + "+num2+" = "+sum);
    System.out.println(num1+" - "+num2+" = "+sub);
    System.out.println(num1+" * "+num2+" = "+mul);
    System.out.println(num1+" / "+num2+" = "+div);
}
```

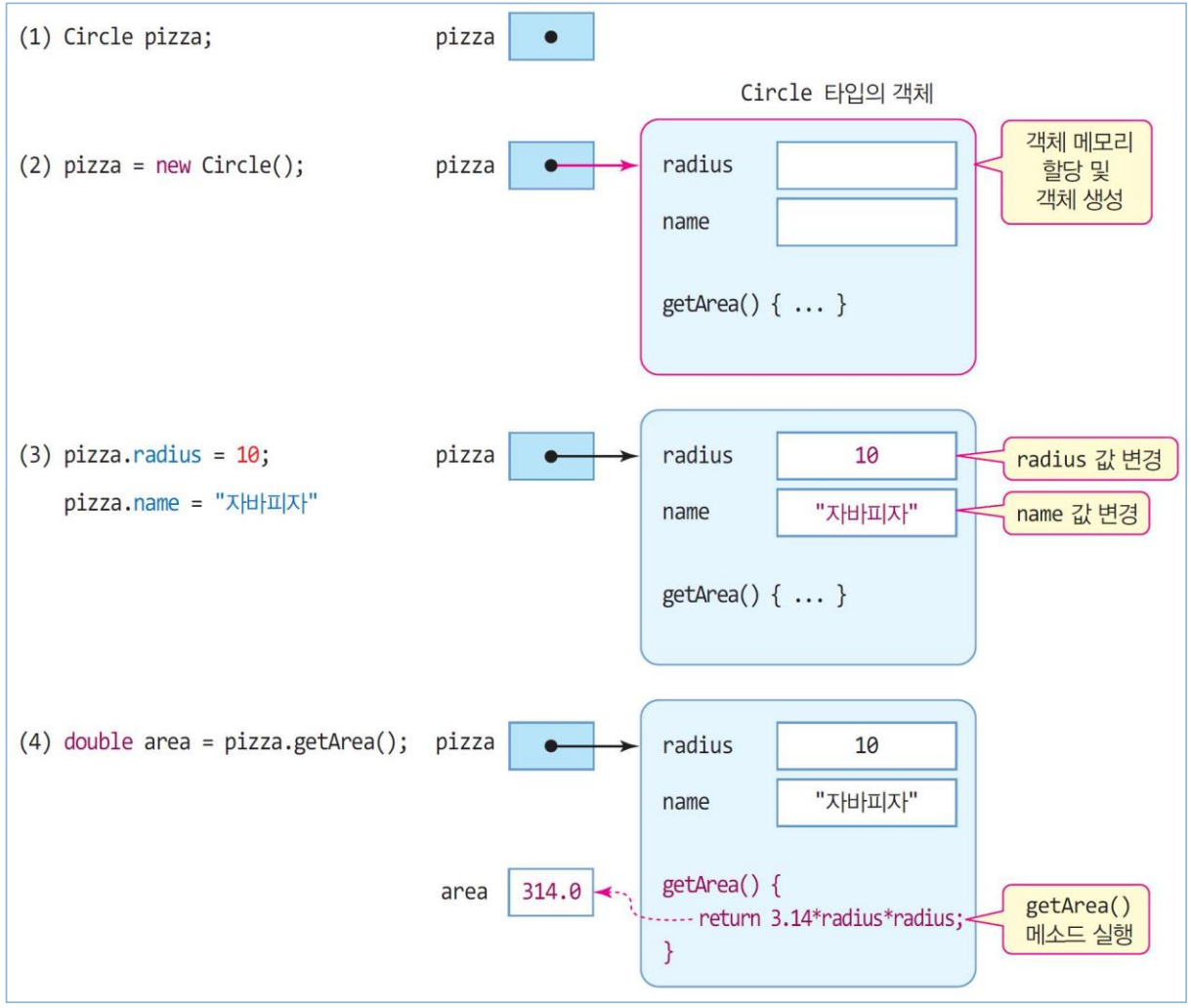
객체 생성 및 접근

객체 생성과 접근

1. 레퍼런스 변수 선언

2. 객체 생성
   - new 연산자 이용

3(4). 객체 멤버 접근
   - 점(.) 연산자 이용



생성자

- 객체가 생성될 때 초기화를 위해 실행되는 메소드

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

    public Circle() { // 매개 변수 없는 생성자
        radius = 1; name = ""; // radius의 초기값은 1
    }
    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);
    }
}
```

생성자 이름은 클래스 이름과 동일

public Circle () {}
→기본생성자

생성자는 리턴 타입 없음

```
자바피자의 면적은 314.0
도넛피자의 면적은 3.14
```

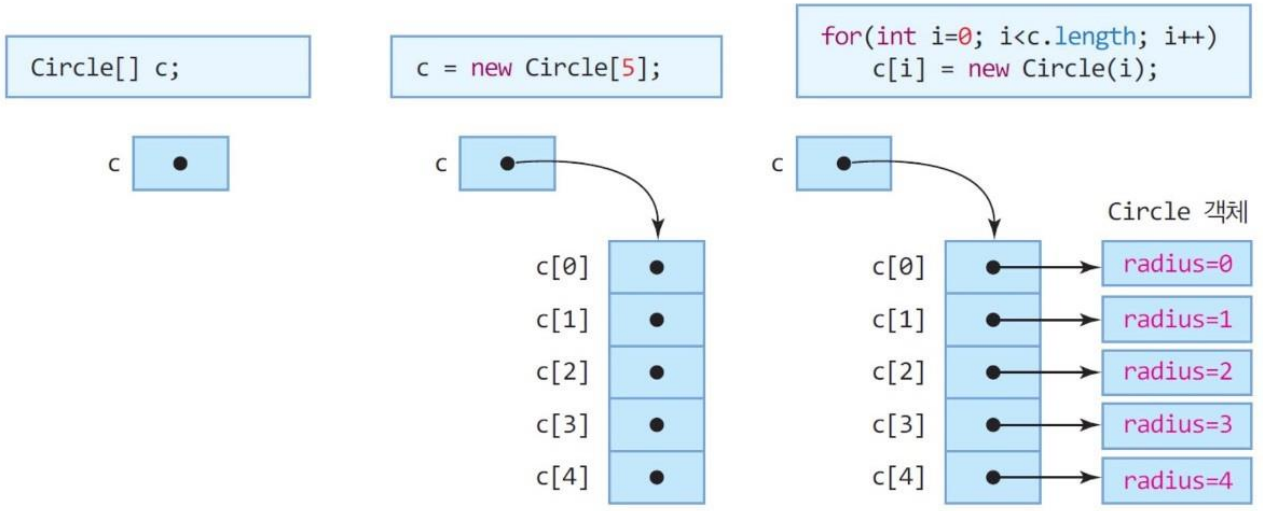
객체배열

- 객체 배열 생성 및 사용

```
Circle [] c;                                  // Circle 배열에 대한 레퍼런스 변수 c 선언
c = new Circle[5];                            // 레퍼런스 배열 생성

for(int i=0; i<c.length; i++) // c.length는 배열 c의 크기로서 5
    c[i] = new Circle(i);                     // 배열의 각 원소 객체 생성
```

```
for(int i=0; i<c.length; i++) // 배열에 있는 모든 Circle 객체의 면적 출력
    System.out.print((int)(c[i].getArea()) + " ");   // 배열의 원소 객체 사용
```

- 객체 배열 선언과 생성 과정



실습1 : 객체

- Point 클래스는 2차원 평면의 좌표를 나타내며, 두 점 사이의 거리를 계산할 수 있다. 이를 바탕으로 아래 요구사항에 맞게 코드를 작성하라.
  - 세 개의 Point 객체 생성
    - 두 개는 매개변수가 있는 생성자를 통해 좌표를 입력 받을 수 있음
    - 하나는 기본 생성자를 통해 (0,0) 좌표 생성
  - 두 점 사이의 거리를 계산하는 함수 작성
  - 가장 짧은 거리 계산(p1 vs p2, p2 vs p3, p1 vs p3)
  - 출력

this


```
public class Circle {
   int radius;

   public Circle() { radius = 1; }
   public Circle(int r) { radius = r; }
   double getArea() {
      return 3.14*radius*radius;
   }
   ...
}
```

=

```
public class Circle {
   int radius;

   public Circle() { this.radius = 1; }
   public Circle(int radius) {
      this.radius = radius;
   }
   double getArea() {
      return 3.14*this.radius*this.radius;
   }
   ...
}
```

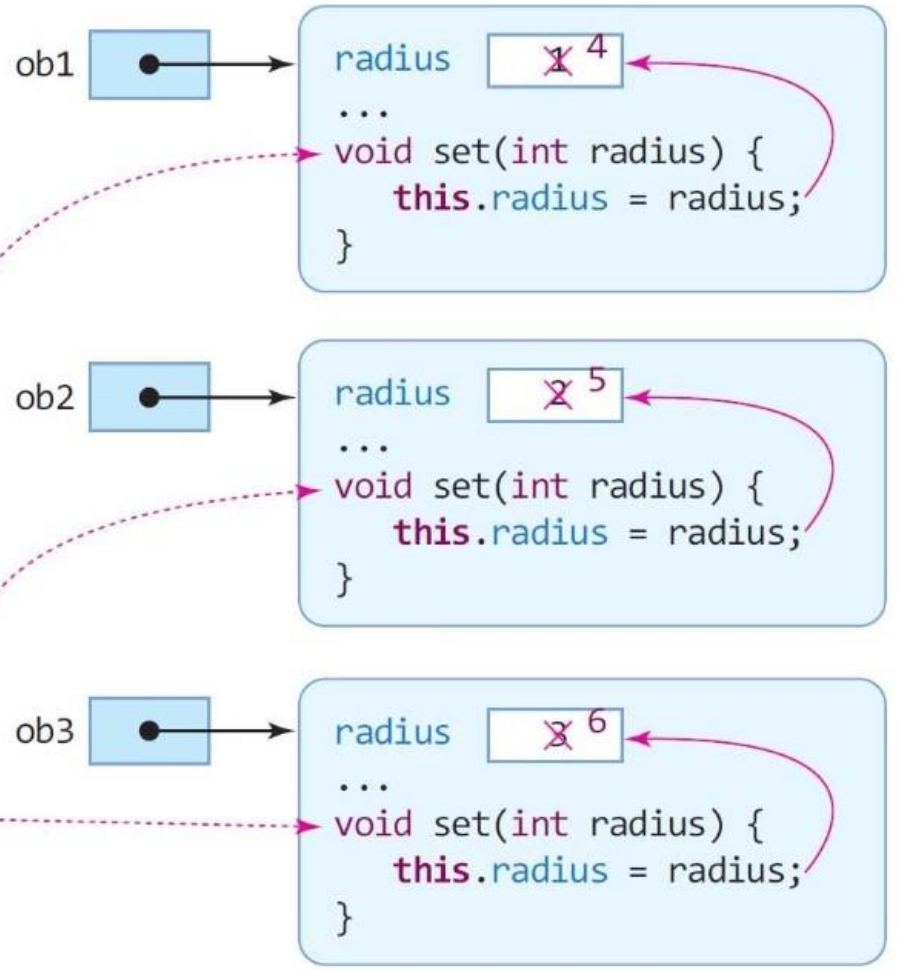
this.멤버형태

this를 사용하여 수정한 경우

- **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);
   }
}
```



실습2 : this

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

- 자동차 정보를 입력 받고, 특정 자동차의 최대 속도를 수정하는 프로그램을 작성하라.

  [조건]

  - 사용자로부터 두 대의 자동차 정보를 입력 받아 객체생성
  - 자동차는 이름(carName)과 최대 속도(maxSpeed)의 속성을 입력 받음
  - 생성된 객체 중 하나를 선택하여 최대속도 수정
  - 최대 속도 수정 방법 : 기존 속도에 추가 속도를 더하는 방식
  - 수정된 객체 정보 출력

메소드 오버로딩(Overloading)

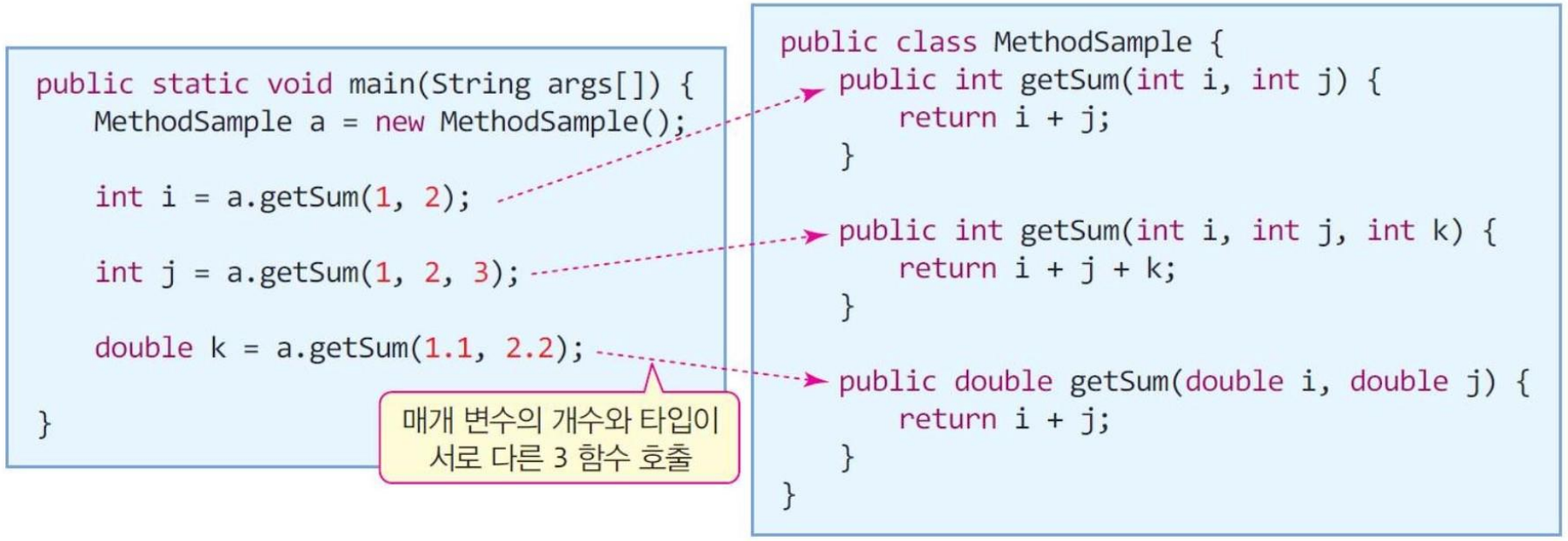
- 이름이 같은 메소드 작성, 다음 2개의 조건
  - 매개변수의 개수나 타입이 서로 다르고
  - 이름이 동일한 메소드들
- 리턴 타입은 오버로딩과 관련 없음
  - 오버로딩의 성공 여부를 따질 때 리턴 타입은 고려하지 않음

```
// 메소드 오버로딩이 성공한 사례

class MethodOverloading {
   public int getSum(int i, int j) {
      return i + j;
   }
   public int getSum(int i, int j, int k) {
      return i + j + k;
   }
}
```

```
// 메소드 오버로딩이 실패한 사례

class MethodOverloadingFail {
   public int getSum(int i, int j) {
      return i + j;
   }
   public double getSum(int i, int j) {
      return (double)(i + j);
   }
}
```

두 개의 getSum() 메소드는 매개변수의 개수, 타입이 모두 같기 때문에 메소드 오버로딩 실패

오버로딩된 메소드 호출


public static void main(String args[]) {
MethodSample a = new MethodSample();
int i = a.getSum(1, 2);
int j = a.getSum(1, 2, 3);
double k = a.getSum(1.1, 2.2);
}
매개 변수의 개수와 타입이 서로 다른 3 함수 호출
public class MethodSample {
public int getSum(int i, int j) {
return i + j;
}
public int getSum(int i, int j, int k) {
return i + j + k;
}
public double getSum(double i, double j) {
return i + j;
}
}

실습3 : 메소드 오버로딩

한국외국어대학교 HANKUK UNIVERSITY OF FOREIGN STUDIES

- 주어진 코드를 수정하지 않고 추가로 코드를 작성하여 완성하라.

[조건]

- 원의 넓이를 구하는 함수와 사각형의 넓이를 구하는 함수는 메소드 오버로딩 을 사용하라.
- 길이가 4인 객체 배열을 사용한다.
- 입력받는 도형은 Circle과 Rectangle 중 하나이다.
- 결과값은 소수 둘째자리까지 출력

```java
import java.io.*;
import java.util.*;

class Figure {
    String name;
    double radius;
    double length;
    double width;
    double area;

    public Figure(String name, double radius){

    }

    public Figure(String name, double length, double width){

    }

    public void getArea(double radius) {

    }
    public void getArea(double length, double width) {

    }

    public static void main(String[] args) {

    }
}
```

- 입력 예시

```
Circle 5
Rectangle 4 6
Circle 3
Rectangle 2 8
```

- 출력 예시

```
Circle의 넓이: 78.50
Rectangle의 넓이: 24.00
Circle의 넓이: 28.26
Rectangle의 넓이: 16.00
```