# 개발 스터디 자바기초 #4

▼

자바 스터디 운영을 위해 작성하는 문서입니다.

2022.06.25작성완료

작성자 : daria (git) (velog)

▼ 짧은 질문시간!

- List 와 Set 의 차이는?

- forEach의 문법은? (for(T 변수 : 객체 리스트))
- 객체 for는 언제 사용할까요?

▼ 숙제 풀이

- **Scanner 클래스를 이용하여 2자리의 정수(10~99사이)를 입력받고, 십의 자리와 1의 자리가 같은 지 판별하여 출력하는 프로그램을 작성하라.**

```
import java.util.Scanner;

public class Hw1 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("2자리수 정수를 입력하세요>>");
    int num = sc.nextInt();

    if(num/10 == num%10) {
      System.out.println("Yes! 10의 자리와 1의 자리가 같습니다.");


    }else
    {
      System.out.println("No! 10의 자리와 1의 자리가 다릅니다.");

    }
  }

}
```

- **변의 길이를 나타내는 정수를 3개 입력받고 이 3개의 수로 삼각형을 만들 수 있는지 판별하라. 삼각형이 되려면 두변의 합이 다른 한 변의 합보다 커야 한다.**

  **정수 3개를 입력하시오>>4 3 5**
  **삼각형이 됩니다.**

```
import java.util.Scanner;

public class Hw2 {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    //연속된 값 입력은 디폴트값은 스페이스
    System.out.print("정수 3개를 입력하시오>>");
    int num1 = sc.nextInt();
    int num2 = sc.nextInt();
    int num3 = sc.nextInt();

    if(num1+num2>num3 || num2+num3 >num1 || num1+num3>num2 ) {
      System.out.println("삼각형이 됩니다.");
    }else {
      System.out.println("삼각형이 안됩니다.");
    }
  }
```

- **원의 정보를 받기 위해 키보드로부터 원의 중심을 나타내는 한 점과 반지름을 입력받는다. 두 개의 원을 입력받고 두 원이 서로 겹치는지 판단하여 출력하라.첫번째 원의 중심과 반지름 입력>>10 10 3**

  **두번째 원의 중심과 반지름 입력>>12 12 2**

  **두 원은 서로 겹친다.**

```
import java.util.Scanner;

public class  Hw3{

  public static void main(String[] args) {
    //외접과 내접의 개념을 생각하자.
    //반지름의 합보다 작으면 원이 겹침

    Scanner sc = new Scanner(System.in);

    System.out.print("첫번째 원의 중심과 반지름 입력>>");
    Double x1 = sc.nextDouble();
    Double y1 = sc.nextDouble();
    Double d1 = sc.nextDouble();
```

```
    System.out.print("두번째 원의 중심과 반지름 입력>>");
    Double x2 = sc.nextDouble();
    Double y2 = sc.nextDouble();
    Double d2 = sc.nextDouble();


    Double disc = Math.sqrt(((x1-x2)*(x1-x2))*((y1-y2)*(y1-y2)));
    Boolean p = disc <= (d1+d2) ? true : false;
    if(p == true) {
      System.out.println("두 원은 서로 겹친다.");
    }else {
      System.out.println("두 원은 서로 겹치지 않습니다.");
    }
  }
}
```

## ▼ 객체지향 프로그래밍

### ▼ 객체 지향 프로그래밍이란?

> 객체 지향 프로그래밍 *Object Oriented Programming*

문제를 여러 개의 객체 단위로 나눠 작업하는 방식
오늘날 가장 많이 사용하는 대표적인 프로그래밍 방식
예 > JAVA, C# etc…

### ▼ 특징

클래스를 이용해 연관 있는 처리 부분(메소드)와 데이터 부분(변수)를 하나로 묶어 객체(인스턴스)를 생성해 사용함.

### ▼ 예시

```
class Person{
  //변수
  String name;
  int age;

  //메소드
  public void introduce(){
    System.out.println("나는 " + age + "살 " + name + "입니다");
  }
  public void eat(){
    System.out.println("먹는다");
  }
  public void wear(){
    System.out.println("입는다");
```

```
  }
  public void sleep(){
    System.out.println("잔다");
  }
}
```

남정윤이 자기소개를 하는 코드를 이렇게 구현할 수 있다.

```
class OopTest1{
  public static void main(String args[]){
    Person p1 = new Person();
    p1.setName("남정윤");
    p1.setAge(21);
    p1.introduce();

  }
}
```

여러명이 자기소개를 하는 코드를 이렇게 구현할 수 있다.

```
class OopTest2{
  public static void main(String args[]){

  Map<String, Integer> people = new HashMap<>();
  people.put("남정윤", 21);
  people.put("장유경", 22);
  people.put("김정호", 35);
  people.put("박정미", 36);

  for(Map.Entry<String, Integer> map : people.entrySet()){
    Person p = new Person();
    p.setName(map.getKey());
    p.setAge(map.getValue());
    p.introduce();
    }
  }
}
```

▼ 클래스 (class)

클래스란 객체를 생성하기 위한 틀.
> 타코야끼를 생성하기 위해 타코야끼틀이 있다고 생각하자 !

타코야끼를 만들기 위해서는 타코야끼틀과 재료가 필요하다.

타코야끼는 안에 들어갈 반죽이 있어야하고, 문어가 있어야하고, 가쓰오부시가 있어야한다.

하지만 소스를 다른 맛으로 선택할 수도 있고, 문어가 아니라 다른 것을 넣고 먹고싶은 사람도 있을 것이다.

그렇기 때문에 만드는 최소한의 틀은 만들어 놓는다고 생각하자

```
class Tacoyakki{
  String maker;
  String dough;
  String ingredients;
  String source;
  boolean katsuobushi; true, false;
  int amount;

  public void cook(){
    System.out.println("타코야끼를 굽는다");
  }
}
```

이런 식으로 말이다 !

나중에 문어를 넣을지, 어떤 소스를 넣을지, 몇개를 구울 지는 나중에 생각하자!

> 💡 타코야끼틀(클래스)로 만들어진 타코야끼는 객체라고 부른다.

▼ 메소드 (method)

다른 프로그래밍 언어에는 **함수**라는 것이 별도로 존재한다. 하지만 자바는 클래스를 떠나 존재하는 것은 있을 수 없기 때문에 자바의 함수는 따로 존재하지 않고 클래스 내에 존재한다.

> 💡 자바는 이러한 클래스 내의 함수를 **메소드**라고 부른다.

믹서기를 생각해보자.

믹서기(객체)는 과일 재료(파라미터)를 갈아서 쥬스(리턴값)으로 만든다.

가는 것! 그것이 우리가 알고싶은 메소드이다.

▼ 구조

```
리턴자료형 메소드명(입력자료형1 매개변수1, 입력자료형2 매개변수2, ...) {
    ...
    return 리턴값;  // 리턴자료형이 void 인 경우에는 return 문이 필요없다.
}

//void cook(){
//System.out.println("타코야끼를 굽는다.");
//}

void cook(int amount){
System.out.println("타코야끼를 " + amount + "개 굽는다");
}
String cook(){
  return "타코야끼를 굽는다";
}

Tacoyakki tacoyakki = new Tacoyakki();
//tacoyakki.cook();
tacoyakki.cook(4); //
tacoyakki.cook(); //"타코야끼를 굽는다"
System.out.println(tacoyakki.cook(4));
String cook = tacoyakki.cook();

String cook(int amount){
  return "타코야끼를 " + amount + "개 굽는다";
}


int i = 5;


String cook = tacoyakki.cook();
```

리턴자료형은 메소드 수행 후 돌려줄 값의 자료형을 의미한다. 메소드의 리턴값은 return 이라는 명령을 사용한다.

메소드는 입출력 유무에 따라 다음과 같이 4가지로 분류할 수 있다.

- 파라미터과 리턴값이 모두 있는 메소드 (String 문자열과 null)
  System.out.println(cook(0)); //null;
  String cook(int amount){
  if(amount == 0){
  return null;
  }
  return "타코야끼를 " + amount + "개 굽는다";
  }

- 파라미터와  리턴값 모두 없는  메소드
  void cook(){}
- 파라미터은 없고 리턴값은 있는 메소드

  String cook(){
  　　return "타코야끼를 굽는다";
  }
- 파라미터은 있고 리턴값은 없는 메소드
  void cook(int amount){
  System.out.println("타코야끼를 " + amount + "개 굽는다");
  }

▼ 예시

```
class cclt{
main{

int a = 1;
int b = 2;
int c = 3;
int d = 4;

sum(sum(a,b),sum(c,d));

}
int sum(int a, int b) {
System.out.println(a+b);
    return a+b;
}
}
```

▼ 예시를 풀어보자!

▼ 정수 a와 b를 받아서 나눈 소수값을 반환하는 메소드

```
float divide(int a, int b){
return (float) a/b;
}
```

▼ 이름과 나이를 받아서 자기소개를 하는 메소드(반환값 x)

```
void introduce(String name, int age){
  System.out.println("제 이름은 " + name + "이고, 나이는 " +age+"살 입니다.");
}
```

▼ a와 b를 받아서 빼는 값을 print하는 메소드

```
void minus(int a, int b){
  System.out.println(a + " - " + b + " = " + (a-b));
}
```

▼ 값을 받지 않고 오늘 날짜를 반환하는 메소드 → new Date()를 사용

```
Date getDate(){
  return new Date();
}
```

▼ static

💡 static 정적인 멤버변수나 메소드를 생성할 때 사용함.
쉽게 말하면, 공용으로 사용하느냐 아니냐

```
class Num{
    static int num = 0; //클래스 필드
    int num2 = 0; //인스턴스 필드
}

public class Static_ex {

    public static void main(String[] args) {
      Num number1 = new Num(); //첫번째 number
      Num number2 = new Num(); //두번째 number

      number1.num++; //클래스 필드 num을 1증가시킴
      number1.num2++; //인스턴스 필드 num을 1증가시킴
      System.out.println(number2.num); //두번째 number의 클래스 필드 출력
      System.out.println(number2.num2); //두번째 number의 인스턴스 필드 출력
    }
}
```

정적 메소드는 객체를 선언하지 않아도 사용할 수 있음.

```
class Name{
    static void print() { //클래스 메소드
  System.out.println("내 이름은 홍길동입니다.");
    }

    void print2() { //인스턴스 메소드
  System.out.println("내 이름은 이순신입니다.");
```

```
    }
}

public class Static_ex {

    public static void main(String[] args) {
        Name.print(); //인스턴스를 생성하지 않아도 호출이 가능

        Name name = new Name(); //인스턴스 생성
        name.print2(); //인스턴스를 생성하여야만 호출이 가능
    }
}
```

정적 메소드는 클래스가 메모리에 올라가는 즉시 생성됨.

그래서 객체를 생성하지 않아도 됨.

(하지만, 클래스가 올라간 상황에서 계속해서 메모리를 차지함)

▼ 실습 예제

계산기 프로그램을 class를 만들어보자

계산기를 몇번 썼는지

해당 값을 도출하기 위해 몇번의 계산을 했는지

클리어

더하기

빼기

나누기

곱하기

%

=

출력하기

자신이 넣고 싶은 기능

으로 계산기 프로그램을 작성해보자