(디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

-8일차 학습 및 질문 노트-

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – Kyeonghwan Lee(이경환)
airtrade7@naver.com

▣ 생성자 1-1

```
class ConsTest {
    /* 데이터 저장 영역 시작 */
    int age;
    String name;
    /* 데이터 저장 영역 끝 */

    /* 기능 설정 영역 시작 */
    // 생성자는 아래와 같이 여러 형식으로 구성할 수 있다!
    // 이와 같이 이름이 같고 입력이 다른 케이스로 매서드를 사용하는 방식에 대해 함수 오버로딩이라고 한다.
    // 입력의 개수를 가지고 판단하지 않으며 사용되는 입력에 데이터타입을 보고 판단한다는 점에 주의해야 한다.
    ConsTest() {
        System.out.println("안녕 나는 ConsTest() 이라고해!");
    }
    ConsTest(int a) {
        System.out.println("안녕 나는 ConsTest(int a) 이라고해!");

        age = a;
    }
    ConsTest(float f) {
        System.out.println("안녕 나는 ConsTest(float f) 라고해!");
    }
    ConsTest(int a, String n) {
        System.out.println("안녕 나는 ConsTest(int a, String n) 이라고해!");

        name = n;
        age = a;
    }

    public int getAge() {
        return age;
    }
    public String getName() {
        return name;
    }
    /* 기능 설정 영역 끝 */
}
```

생성자

▶ 값을 반환하지 않는다.(리턴 타입이 없다.)
▶ 생성자의 이름은 클래스의 이름과 동일하다.
▶ new를 할 때 호출된다.

함수오버로딩

▶ 이름이 같고 입력이 다른 케이스로 매서드를 사용하는 방식

▶ 클래스는 데이터 타입, 객체는 데이터가 메모리에 올라가는 것, 인스턴스 = 객체
▶데이터의 저장 영역과 기능 설정 영역이 꼭 관련 있어야 되는 건 아니다.
Ex) 프로그래머의 치킨 튀기는 능력 이랄까...

▣ 생성자 1-2

```
public class WhyConstructorTest {
    public static void main(String[] args) {
        ConsTest ct1 = new ConsTest();
        ConsTest ct2 = new ConsTest(10);
        ConsTest ct3 = new ConsTest(20, "hi");
        ConsTest ct4 = new ConsTest(40);
        ConsTest ct5 = new ConsTest(3.3f);

        // * 우리가 사용하는 모든 데이터는
        // 메모리(PC상에서 DRAM)에 올라가야지만 사용할 수 있고 눈으로 볼 수 있다.
        // 결국 객체라는 단어 자체는 메모리에 데이터를 올렸습니다의 추상화된 표현이라 볼 수 있겠다.

        // 원래 여기서 사용했던 Setter는 어디로 갔나요 ?
        // Setter가 없는데도 결과가 나오네요 ?
        // 결국 생성자는 객체를 처음 생성할 때 초기값을 설정해주는 역할을 수행한다!
        // (결론적으로 초기 생성에 한정하여 Setter의 역할을 대신해줄 수 있다)

        /*
        ct2.setName();
        ct2.setAge();
        ct2.setMajor();
              vs
        new ConsTest(이름, 나이, 전공);
         */

        // Tip: 아직 우리는 초보자 단계에 있다.
        //      그러므로 Setter를 안써서 구현을 못하는것보다는
        //      Setter를 사용해서 일단 구현을 할 수 있도록 하는 것이 중요하다!

        System.out.printf("ct1 name = %s, age = %d\n", ct1.getName(), ct1.getAge());
        System.out.printf("ct2 name = %s, age = %d\n", ct2.getName(), ct2.getAge());
        System.out.printf("ct1 name = %s, age = %d\n", ct1.getName(), ct1.getAge());

        System.out.printf("ct3 name = %s, age = %d\n", ct3.getName(), ct3.getAge());
        System.out.printf("ct4 name = %s, age = %d\n", ct4.getName(), ct4.getAge());
    }
}
```

클래스는 같지만 ct1.ct2...는 독립적인 개체이다. 서로 영향을 주지 않고 데이터 타입으로 판별한다.

Int의 초기값은 0, String의 초기값은 null 결론은 아무것도 들어있지 않다.

편의성을 위해 후자 사용

▣ Prob35

```java
import java.util.Scanner;

class TestDice3 {
    int comDice;
    int userDice;

    Scanner scan;

    TestDice3() {
        comDice = getRandomDice();
        userDice = getRandomDice();

        scan = new Scanner(System.in);
    }

    int getRandomDice() {
        return (int)(Math.random() * 6 + 1);
    }

    // 리턴이 없어서 void(Setter도 리턴이 없어서 void)
    void checkWinner() {
        switch (whoWin()) {
            case 1:
                System.out.printf("폐관수련입니다. %d(컴퓨터) vs %d(사용자)\n", comDice, userDice);
                break;
            case 2:
                System.out.printf("사용자가 이겼습니다. %d(컴퓨터) vs %d(사용자)\n", comDice, userDice);
                break;
            case 3:
                System.out.printf("비겼으니 형은 면하였습니다. %d(컴퓨터) vs %d(사용자)\n", comDice, userDice);
                break;
        }
    }

    int whoWin() {
        if (comDice > userDice) {
            return 1;
        } else if (comDice < userDice) {
            return 2;
        } else {
            return 3;
        }
    }

    Boolean redoDiceGame() {
        System.out.print("게임을 계속 하시겠습니까 ? 0(아니오), 1(예) ");

        int num = scan.nextInt();

        Boolean isTrue = false;

        switch (num) {
            case 0:
                isTrue = false;
                break;
            case 1:
                // 게임을 다시 재개하므로 주사위값을 새롭게 설정할 필요가 있다.
                comDice = getRandomDice();
                userDice = getRandomDice();
                isTrue = true;
                break;
        }

        return isTrue;
    }
}

public class Prob35Enhance {
    public static void main(String[] args) {
        TestDice3 td = new TestDice3();

        // do ~ while()의 경우엔
        // 무조건 처음은 실행하고 이후엔 조건에 따라 반복을할지 말지를 결정한다.
        do {
            // 일단 한 번 해봐 ~
            td.checkWinner();
        } while(td.redoDiceGame()); // 한 판 더 할까 ?
        // 주사위를 다시 굴려서 셋팅
        // 다시 진행할지 여부는 어떻게 설정할 것인가 ?
    }
}

// Q: if, switch, for, do ~ while 등등은 class 인가요 ?
// A: 이들은 모두 키워드(keyword)라는 것에 해당하는 녀석들입니다.
//    클래스에 해당하는 것은 아니며 특정 동작을 서포트하는 컴파일러 전용 키워드라고 보면 됩니다.
```

Q:void가 쓰이는 이유에 대해서 다시 한번 설명 부탁드리겠습니다.

Q:해당 return은 어떤 역할을 하는 건가요? 삭제 시 미싱 리턴 에러 발생