(디지털컨버전스)스마트 콘텐츠와 웹 융합응용SW개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 –Joongyeon Kim(김중연)
jjjr69@naver.com

2021년 5월 17일 질문노

2021년 5월 14일 질문노트

[김중연]

```java
/* 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 */
}

public class WhyConstructorTest {
    public static void main(String[] args) {
        ConsTest ct1 = new ConsTest();          //같은 클래스 변수이지만 각각 생성되는 인스턴스가 다르다 ct1
        ConsTest ct2 = new ConsTest( a: 10);    //같은 클래스 변수이지만 각각 생성되는 인스턴스가 다르다 ct2
        ConsTest ct3 = new ConsTest( a: 20, n: "hi");  //같은 클래스 변수이지만 각각 생성되는 인스턴스가 다르다 ct3
        ConsTest ct4 = new ConsTest( a: 40);  //같은 클래스 변수이지만 각각 생성되는 인스턴스가 다르다 ct4
        ConsTest ct5 = new ConsTest( f: 3.3f);  //같은 클래스 변수이지만 각각 생성되는 인스턴스가 다르다 ct5

        // * 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!

        //ct1에서는 아무것도 변수를 입력하지 않아서 0이 나옴
        //ct2, 3, 4는 각각 입력한 변수의 값이 호출된다.
        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());
```

```java
        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!

        //ct1에서는 아무것도 변수를 입력하지 않아서 0이 나옴
        //ct2, 3, 4는 각각 입력한 변수의 값이 호출된다.
        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("ct3 name = %s, age = %d\n", ct3.getName(), ct3.getAge());
        System.out.printf("ct4 name = %s, age = %d\n", ct4.getName(), ct4.getAge());
    }
}
```

WhyConstructorTest

```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" -javaagent:C:\Users\jjjr9\AppData\Loca
안녕 나는 ConsTest() 이라고해!
안녕 나는 ConsTest(int a, String n) 이라고해!
안녕 나는 ConsTest(int a, String n) 이라고해!
안녕 나는 ConsTest(int a) 이라고해!
안녕 나는 ConsTest(float f) 라고해!
ct1 name = null, age = 0
ct2 name = ㅜ, age = 10
ct3 name = hi, age = 20
ct4 name = null, age = 40
```

질문

Print는 총 4개이고 객체생성한 것은 총 5개입니다 그런데 메인에 입력한 프린트는 4개만나오는데 상위클래스에 입력한 프린트들은 객체수에 맞춰서 출력됐습니다. 왜 이런건지 알 수 있을까요? 객체를 생성할 때 같이 출력되는건가요?

```java
class PersonTest{
    String name;
    int age;

    //생성자의 특징
    //1. 정말 희안하게도 리턴 타입이 없다.
    //2. 클래스 이름과 메소드 이름이 같다.
    //3. new를 할 때 호출된다.
    PersonTest() { System.out.println("안녕 나는 생성자야"); }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}

public class ConstructorTest {
    public static void main(String[] args) {
        PersonTest pt = new PersonTest();

      pt.setAge(10);
      pt.setName("가오왕");

      System.out.printf("나는 %d살인 %s이다!\n", pt.getAge(), pt.getName());
  }
```

```java
class TestDice {
    int comDice;    //먼저 컴퓨터와 사람의 주사위의 변수를 지정한다
    int userDice;

    TestDice() {
        comDice = getRandomDice(); //comDice와 userDice에 랜덤한 주사위 값을 대입한다.
        userDice = getRandomDice();
    }

    int getRandomDice() { return (int)(Math.random() * 6 + 1); // 주사위 값을 반환하는 변수를 만든다 }

    Boolean userWin() {  //사용자와 컴퓨터의 주사위 값을 정한다.
        System.out.printf("%d(컴퓨터) vs %d(사용자)\n", comDice, userDice);

        if (comDice > userDice) {
            return false;
        } else if (comDice < userDice) {
            return true;
        } else {
            System.out.println("무승부입니다.");
            return false;
        }
    }
}

public class Prob34 {
    public static void main(String[] args) {
        // 우리가 이전에 Random과 제어문을 활용해서 주사위 게임을 만들었던적이 있다.
        // 이 주사위 게임을 class 방식으로 다시 만들어보자!
        // 컴퓨터도 주사위를 굴리고 사용자도 주사위를 굴려서 누가 더 큰 숫자를 얻었는지 확인해보자!
        TestDice td = new TestDice();

     if (td.userWin()) { //생성자는 초기생성에 한정해서 Setter의 역할을 대신할 수 있다
         System.out.println("사용자가 승리하였습니다.");
     } else {
         System.out.println("컴퓨터도 못이겼다. 폐관수련이 답이다.");
     }
}
```

```java
import java.util.Scanner;

class TestDice3 {
    int comDice;
    int userDice;

    Scanner scan;

    TestDice3() { //생성자를 만든다!
        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()) {  // 밑에 결과에 따라 자동으로 선택되는 스위치다!
            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() {  //게임을 계속 실행하기위한 코드
        System.out.print("게임을 계속 하시겠습니까 ? 0(아니오), 1(예) ");

        int num = scan.nextInt();

        Boolean isTrue = false; //isTrue의 값을 넣기위한 코드(true를 써도 상관없다.)

        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;  // 입력된 값이 false면 셈송됴, true면 새시삭
    }


ublic class Prob35 {
    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.
```