(디지털컨버전스)스마트 콘텐츠와 웹 융합응용SW개발자 양성과정

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

2021년 5월 14일 질문노트

[김중연]

```java
import java.util.Scanner;

class Student {
    String name;
    int Class;
    int age;
    int IQ;

}

public class StudentTest {
    public static void main(String[] args) {
        //30번 문제

          Student student = new Student();
        Scanner scan=new Scanner(System.in);


        System.out.print("나이를 입력하세요");
        student.age= scan.nextInt();


        System.out.print("학년을 입력하세요");
        student.Class= scan.nextInt();


        System.out.print("IQ을 입력하세요");
        student.IQ= scan.nextInt();


        System.out.print("이름을 입력하세요");
        student.name = scan.nextLine();

        System.out.printf("이 학생의 나이는%d 세이며 %d학년이고 %d의 아이큐를 가졌다 이름은 %s라고한다\n."
                , student.age, student.Class, student.IQ, student.name);
    }
}
```

질문 1.
이름을 입력하는 부분이 스킵되고 바로 출력됩니다

(이름을 입력하세요 이 학생의 나이는1 세이며 1학년이고 12의 아이큐를 가졌다 이름은 라고한다)

왜 이런건지 알려주실 수 있나요?

Prob27.java AllocHeapArray.java Test2.java Prob28.java StudentTest.java

```java
public class Prob28 {
    public static void main(String[] args) {
        System.out.println("1, 2, 4, 8, ... 1024, ...");

        Scanner scan = new Scanner(System.in);

        System.out.print("몇 번째 항을 구할까요 ? ");
        int num = scan.nextInt();

        if (num <= 0) {
            System.out.println("잘못된 값을 입력하였습니다.");
        } else if (num < 2) {
            System.out.println("당신이 찾는 값은 1입니다.");
        } else {
            int numArr[] = new int[num];

            numArr[0] = 1;

            for (int i = 1; i < numArr.length; i++) {
                // 현재값은 이전값 x 2
                // -------------------------------
                // | 1 | 2 |   |   |   |   |   |
                // -------------------------------
                //  [0] [1] [2] [3] [4] [5] [6]

                numArr[i] = numArr[i - 1] * 2;
            }

            System.out.printf("%d 번째 항은 = %d\n", num, numArr[num - 1]);
        }
    }
```

2의 제곱승 인 수를 출력하는 코드

3의 제곱하는 거랑 거의 같다

| ength는 배열의 갯수를 의미한다 (' . ' 은 공간 내부에 접근하게다는 뜻을 내포한다.)

그리고 스캐너 변수인 num은 배열인 numArr[]에 대입된다

이 각주를 보면 알 수 있듯이 배열은 0부터 시작한다 따라서 -1을 눌러서 0을 빼버려야 한다!

출력할때도 마찬가지로 -1을 해주어야 에러가 안난다

```java
import java.util.Scanner;

public class Test2 {
    public static void main(String[] args) {
        //29번 문제
        System.out.println("3, 9, 27....");

        Scanner scan = new Scanner(System.in);

        System.out.print("몇 번째 항을 구할까요 ? ");
        int num = scan.nextInt();

        if (num <= 0) {
            System.out.println("잘못된 값을 입력하였습니다.");
        } else if (num < 2) {
            System.out.println("당신이 찾는 값은 1입니다.");
        } else {
            int numArr[] = new int[num];  //이 num은 스캐너 변수에 있는 num

            numArr[0] = 1;

            for (int i = 1; i < numArr.length; i++) {
                // Math.pow(A, B)는 A^B(A의 B승)을 계산한다.
                // Math.pow는 double을 결과로 내놓기 때문에 강제로 int 타입으로 변형하였음
                numArr[i] = (int)Math.pow(3, i);
            }

            System.out.printf("%d 번째 항은 = %d\n", num, numArr[num - 1]); //배열은 0부터시작
        }
    }
}
```

29번 문제 3의제곱 중에 원하는 숫자를 호출하기

0이하의 수를 입력하면 나오면 첫번째 if문이 출력되고 1을 입력하면 2번째if문이 출력된다.

마지막으로 3번째인 else문에서 3의 제곱을 구하는데 여기서 Math.pow(3, i)는 3의 i승을 의미한다

그리고 length는 배열의 숫자를 의미한다
(' . ' 은 공간 내부에 접근하게다는 뜻을 내포한다.)

그리고 스캐너 변수인 num은 배열인 numArr[]에 대입된다.

이 부분에서 계산이 이루어진다 (1부터 시작해서 만약 3을 입력했을 경우 for문을 2번 반복한다)

배열은 0부터 시작이니 제곱하는 부분에 -1을 해준다 (안 그러면 오류난다)

```java
import java.util.Scanner;

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

        Scanner scan = new Scanner(System.in);
        System.out.print("학급에 학생이 몇 명 있습니까 ? ");

        int studentNum = scan.nextInt();

        // 동적할당 되는 데이터를 관리하는 메모리 - Heap
        // new 로 만든 데이터는 전부 Heap에서 관리하게 된다.
        // 기존에 우리가 stack에 배열을 만들 경우엔 항상 제약 사항이 발생했었다.
        // 어떤 제약 사항 ? 개수가 고정됨(할당된 값으로)

        // Heap 방식의 할당에서는 공간을 필요한만큼 할당할 수 있어서 유연한 접근이 가능하다.
        // 근데 여기에도 제약사항이 있다 ? stack 보다 느리다.

        // 새로운 개념 new를 통해 공간을 만드는 방법
        // 1. new 를 적는다.
        // 2. 데이터타입을 적는다.
        // 3. 만약 데이터타입이 배열이라면 대괄호를 열고 몇 개를 만들지 적는다.
        //    만약 데이터타입이 클래스라면 소괄호를 열고 닫고 필요하다면 인자를 설정한다.
        //    (아직 클래스는 배우지 않았으므로 패스)

        // Heap 공간에 int형 배열을 studentNum 개수만큼 만들겠습니다.
        // 그리고 studentArr는 Heap에 생성된 공간을 제어하게 된다.
        // [0], [1], ... [3] 등의 인덱스가 Heap에 있는 공간을 바라보게 된다는 뜻인데
        // 복잡하게 생각하기 싫다면 기존 배열 제어하는 방식과 동일하되
        // 메모리 생성시에만 아래와 같은 방식을 사용한다고 보면 되겠다.
        int studentArr[] = new int[studentNum];

        // 비교 대상
        int arr[] = { 1, 2, 3, 4, 5 };

        for (int i = 0; i < studentNum; i++) {
            // studentNum 만큼 생성되니 해당 학생숫자에 맞게 80 ~ 100점의 점수를 가지도록 만듬
            studentArr[i] = (int)(Math.random() * 21) + 80;
            System.out.printf("studentArr[%d] = %d\n", i, studentArr[i]);
        }
    }
}
```

Heap

기존에 stack에 배열을 만들면 개수가 고정되는 (할당되는 값으로) 불편함이 있다

Heap은 공간을 필요한 만큼 할당 할 수 있어서 유연한 접근이 가능하다(다만 stack보다 느리다)

studentNum이 스캐너 변수이므로 입력하는 숫자만큼 배열을 할 수가 있다.

그리고 studentArr에 대입했으므로 studentArr[i]<-는 입력값이 된다

만약 21을 입력했다면 21개의 결과가 출력된다.

```java
public class Prob27 {
    public void test() {
        int num = 3, res = 2;
    }
    public static void main(String[] args) {
        // 피보나치수열
        // 사용자가 15를 입력하면 15번째 값을, 8을 입력하면 8번째 값을 구하도록 프로그래밍
        System.out.println("사용자로부터 n을 입력받아 n 번째 피보나치 수열의 항을 구합니다.");

        Scanner scan = new Scanner(System.in);
        System.out.print("n 값을 입력하시오: ");

        int num = scan.nextInt();
        int res = 0;

        // 예외 처리(음수와 0)
        if (num <= 0) {
            System.out.println("0번째 항 혹은 음수 항은 존재하지 않습니다.");
        } else if (num < 3) {    // 0이 아니며 음수가 아니고 3미만이라면 1번째와 2번째뿐
            // 1번째와 2번째값은 무조건적으로 1에 해당함
            System.out.println("당신이 찾는 값은 1입니다.");
        } else {
            // 위의 조건이 모두 만족되지 않는다는 것은 결론적으로 숫자 3보다 크다는 것을 의미함
            int first = 1, second = 1;
            // -2 를 했던 이유는 시작할 때 first 값과 second 값 2개를 알고 시작했기 때문
            for (int i = 2; i < num; i++) {
                //for (int i = 0; i < num - 2; i++) {
                res = first + second;
                first = second;
                second = res;
            }
            System.out.println("결과는 = " + res);
            System.out.println("first = " + first);
```

??? (없어도 잘 굴러간다.)

피보나치수열

첫번째 if문에서 0과 음수를 걸러내고
두번째 if문에서 첫번째 수와 두번째 수를 걸러내며
세번째인 else문에서 피보나치수열을 위한 계산을 한다.

피보나치수열을 만드는 핵심 코드!
첫번째와 두번째를 더하고
두번째와 세번째를 더해가는 것이 핵심이다

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

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

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

public class ClassTest {
    public static void main(String[] args) {
        // 클래스는 사용자가 직접 만들 수 있는 데이터타입(커스텀 가능)
        // 변수를 만드는 것과 동일하게 클래스를 사용해서 변수를 만들어보자
        Person person = new Person();

        person.setAge(12);
        person.setName("얍");


        System.out.println("이 사람은 몇살?"+person.age);
        System.out.println("이 사람의 이름은?"+person.name);

        System.out.printf("이 사람의 이름은%s이고 나이는%d세 입니다\n", person.name, person.age);
```

클래스 변수

Class는 사용자가 직접 만들 수 있는 데이터타입이다(커스텀)

Class class = new Class

이렇게 입력하여 클래스 변수를 생성하고 다른 클래스에서 불러올 수 있다.

Getter,Setter에 관한 설명은 9번에 있다

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

        int arr[] = { 2, 3, 4, 5, 6, 7 };

        System.out.println("arr의 길이 = " + arr.length);

        // 동적 할당은 모두 프로그램 실행 도중 생성하는 것을 의미한다(그래서 느림)
        // 동적 할당이 아닌 것들은 준비를 싹 해뒀다가 준비된 상태로 올림(그래서 빠름)
        int dynamicArr[] = new int[30];
        // 30개니까 0 ~ 30이 아니라 0 ~ 29라는 부분을 주의해야합니다.

        System.out.println("dynamicArr의 길이 = " + dynamicArr.length);
    }
}
```

배열

이렇게 저장된 배열은 Stack에 저장된다
(미리 할당하며 Heap보다 빠르다)
Stack: 정적 메모리 할당

Heap: 동적 메모리 할당

동적 할당
미리 할당하는 것이 아닌 일정 범위를
지정하여 프로그램 도중에 생성하는 것을
의미한다

Heap에서 적어놓은 거랑 일맥상통한다
(애초에 동적할당이 저장되는 메모리가
Heap이다)

0~29만큼 입력이 가능하다 (그 이상
넘어가면 에러난다)

Getter, Setter만드는 방법 Getter: 값을 얻고자 할 때 사용함 Setter: 값을 설정하고자 할 때 사용함 (void의 의미: 값을 반환하지 않는다)

```java
class Teacher{
    int age;
    String name;
    String major;

    // 우선 제일 간단한 Getter, Setter부터
    // 마우스커서를 아래 갖다 놓고 Alt+Insert를 누른다
    // Getter & Setter ->전부 선택하고 Ok하면 자동으로 Setter와 Getter가 만들어진다

    //메소드를 만드는 방법
    //1. 먼저 리턴(return) 타입을 작성한다
    //2. 메소드의 이름을 작성한다(용도에 맞게 작성한다)
    //    보통 Getter의 경우 값을 얻고자 할 때(즉 return 용도로 사용)
    //    Setter의 경우 값을 설정하고자 할 때 사용한다
    //    그 외에도 커스텀 메소드의 경우엔
    //    자동 완성 기능으로 만들 수 없기 때문에 메소그 작성법에 대해  알 필요가 있다.
    //3. 소괄호 내부에 인자로 입력 받을 매개변수를 설정한다
    //4. 중괄호 내부에 해당 메소드(기능)이 수행할 업무를 작성한다.

    /* 누군가가 age가 궁금해서 물어본다
    이에 대한 답을 해준다면 Getter에 해당한다.
    클래스 작성할 때는 이니셜마다 대문자를 붙였다.
    메소드는 시작은 소문자 그 이후부터의 이니셜은 대문자
    소괄호 내부는 인자가 배치되는데 텅 비어있는 것은 인자(입력)이 없는 상태
    결론: 값을 얻는 목적으로 사용하는 것이 Getter
     */
    int getAge(){
        return age;
    }

    //회원가입시 집 주소 적으라고 나오는데 여기서 입력하는 값들을 처리하는게 Setter에 해당한다
    // 특정한 값을 설정하는 목적으로 Setter가 사용된다.
    void setAge(int age){
        //this.age는 클래스 내부에 있는 age를 의미한다.
        //age는 입력으로 들어온 age에 해당한다.
        //결론: 값을 설정하는 목적으로 사용하는 것이 Setter
        this.age=age;
    }
    String getName(){
        return name;
    }
    void setName(String name){
        this.name=name;
    }
    String getMajor(){
        return major;
    }
    void setMajor(String major){
        this.major=major;
    }

}

public class ClassMethodTest {
    public static void main(String[] args) {
        Teacher t=new Teacher();

        t.setAge(40);
        t.setMajor("Physics");
        t.setName("Sam");

        System.out.printf("%s는 %s를 전공하였고 %d세다\n",
                          t.getName(), t.getMajor(), t.getAge());
```

```java
import java.util.Scanner;

class Dog{
    int age;
    String name;
    String breed;

    int getAge(){
        return age;
    }
    void setAge(int age){
        this.age=age;
    }
    String getName(){
        return name;     //여기서 값을 반환함
    }
    void setName(String name){
        this.name=name;      //여기서 값을 설정함
    }
    String getBreed(){
        return breed;
    }
    void setBreed(String breed){
        this.breed=breed;
    }
}
public class DogProfile {
    public static void main(String[] args) {
        Dog d=new Dog();
        Scanner scan =new Scanner(System.in);

        d.setName("멍멍이");    //이름 설정
        d.setAge(13);           //나이 설정
        d.setBreed("허스키");   //품종 설정

        System.out.printf("이름: %s 품종: %s, 나이: %d\n", d.name, d.breed, d.age);
    }
}
```

강아지 클래스 생성 32번 문제

Getter와 Setter를 생성한다 Setter엔 각 데이터타입에 맞는 변수를 소괄호 안에 입력해줘야 한다

Dog 클래스에 인스턴스를 만든 후 Setter에 값을 입력해준다

```java
class Cat{
    int age;
    String name;
    String breed;

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

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

    String getBreed(){
        return breed;
    }
    void setBreed(String breed){
        this.breed=breed;
    }
}

public class CatProfile {
    public static void main(String[] args) {
        Cat cat = new Cat();


       cat.setName("야통이");
       cat.setAge(11);
       cat.setBreed("코숏");

       System.out.printf("이름 : %s, 나이 : %d, 품종 : %s\n", cat.getName(), cat.getAge(), cat.getBreed() );
```

고양이 클래스 생성 33번 문제

class Cat 생성 후 값을 입력하고 Getter와 Setter를 생성한다

그후 main에 cat인스턴스를 생성한 후에 cat.set...(설정할 내용)을 입력한다