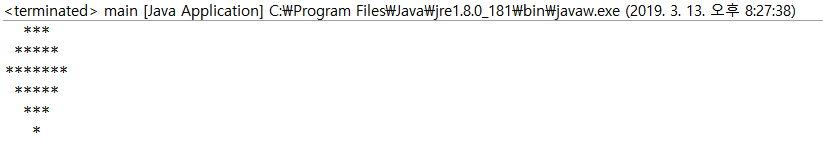
< JAVA PROGRAMMING LAB -1장: JAVA 출력문>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**1. 다음과 같은 패턴을 출력하는 프로그램을 작성하시오.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**public** **class** main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

System.***out***.println(" \* ");

System.***out***.println(" \*\*\* ");

System.***out***.println(" \*\*\*\*\* ");

System.***out***.println("\*\*\*\*\*\*\*");

System.***out***.println(" \*\*\*\*\* ");

System.***out***.println(" \*\*\* ");

System.***out***.println(" \* ");

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

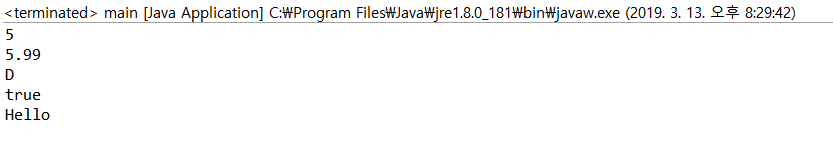
◆ DESCRIPTION

main클래스를 정의하였고 main메소드를 정의하였다. 자바는 객체지향 언어라서 클래스를 중요시 여긴다. c언어에서 처음 main함수에서 소스를 작성해야 되듯이 자바도 클래스를 선언하고 메소드를 선언한 후 소스를 작성하는 것이 규칙이다. 자바에서 출력문은 System.out.println();이다. 문제에서 별 문장을 7줄 작성하라 하였으니 출력문도 위처럼7문장을 작성했다.

< JAVA PROGRAMMING LAB -1장: JAVA 출력문>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**2. 다음과 가이 변수를 정의하고 변수의 값을 출력하는 프로그램을 작성하라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**public** **class** main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** myNum=5;

**float** myFloatNum=5.99f;

**char** myLetter='D';

**boolean** myBool=**true**;

String myText="Hello";

System.***out***.println(myNum+"\n"+myFloatNum+"\n"+myLetter+"\n"+myBool+"\n"+myText);

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

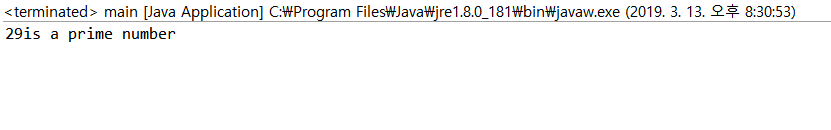
◆ DESCRIPTION

myNum변수를 정수형으로 선언하고 5로 초기화하였다. myfloatnum변수를 실수형으로 선언하고 5.99로 초기화하였다. 초기화할 때 ‘f’을 붙인 이유는 일반적으로 실수는 double로 인식하기 때문에 float로 바꾸어야 하는 것이다. 만약 바꾸지 않으면 4바이트 공간에 8바이트 실수가 변형되서 들어가는 상황이 발생한다. 그리고 myLetter변수를 문자형으로 선언하고 ‘D;로 초기화하였다. myBool이라는 변수를 Boolean형으로 선언 후 true로 초기화하였다. boolean은 java에서 새로 나온 자료형인데, 변수에 들어가는 데이터 타입이 false와 true밖에 없게 하는 것이다. 이후 myText라는 변수를 문자열형으로 선언 후 “Hello”로 초기화하였다. 이렇게 초기화한 변수들을 출력문을 통해 출력하였다.

< JAVA PROGRAMMING LAB -1장: JAVA 출력문>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**3. 29가 소수인지 아닌지를 검사하는 프로그램을 다음과 같이 작성해보자.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**public** **class** main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** num=29;

**boolean** flag=**false**;

**for**(**int** i=2;i<num; ++i) {

**if**(num%i==0) {

flag=**true**;

**break**;

}

}

**if**(!flag)

System.***out***.println(num+"is a prime number");

**else**

System.***out***.println(num+"is not a prime number");

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

◆ DESCRIPTION

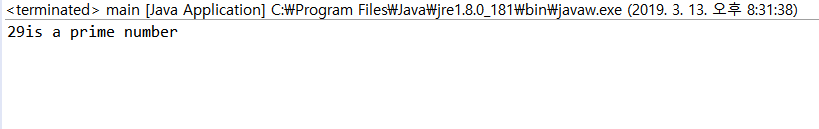
num변수를 정수형으로 선언 후 29로 값을 초기화하였다. flag변수를 boolean형으로 선언 후 false로 선언하였다. 이후 i가 2부터 num-1(28)일 때 까지 반복하는 반복문을 만들었다. 소스 블록에는 num(29)에 i를 나눈 나머지가 0이면 flag변수에 true를 저장하고 반복문을 빠져 나오는 소스밖에 없다. 29에 2부터 28까지 나누어서 나머지가 0인 것이 하나도 없으면 false를 유지하고 소수라고 볼 수 있다.

반복문을 빠져나온 후 만약 flag값이 false라면 소수라고 출력하고 그렇지 않다면 소수가 아니라고 출력한다.

< JAVA PROGRAMMING LAB -1장: JAVA 출력문>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**4. 3번 문제를 while 반복루프를 이용하여 다시 작성해보자.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**public** **class** main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** num=29,i=2;

**boolean** flag=**false**;

**while**(i<num) {

**if**(num%i==0) {

flag=**true**;

**break**;}

++i;

}

**if**(!flag)

System.***out***.println(num+"is a prime number");

**else**

System.***out***.println(num+"is not a prime number");

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

◆ DESCRIPTION

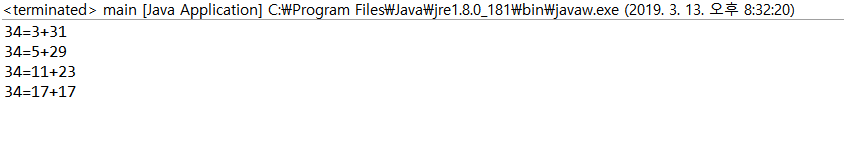
while문만 설명하겠다. for문은 (0.변수를 초기화하고 1.조건이 만족되면 소스 블록을 실행하고 2.변수 값을 올리고)에서 1,2번이 반복되는 구조이다. while문은 for문과 달리 변수를 초기화하는 구간이 없으니 반복문 이전에 i변수를 선언 및 초기화하였다.

while문은 보통 조건이 거짓이 될 때까지 실행이 되는 반복문으로 인식된다. 그래서 조건 구간에 i<num을 작성했다. 이후 똑같이 if문을 작성하였고 최후에 i의 값을 1올리면 3번 소스의 for문과 동일 기능을 하게 된다.

< JAVA PROGRAMMING LAB -1장: JAVA 출력문>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**5. 어떤 수(34)가 소수 2개의 합으로 표시될 수 있는지를 검사하는 프로그램을 작성해보자. 이것은 암호화에도 사용된다. 함수를 사용해도 좋고 사용하지 않아도 된다.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**public** **class** main {

**public** **static** **void** main(String[] args) {

**boolean** flag=**false**,flag2=**false**;

**int** min=0;

**for**(**int** i=3;i<34;i++) {

flag=**false**;

flag2=**false**;

**for**(**int** j=2;j<i;j++) {

**if**(i%j==0) {

flag=**true**;

**break**;}

}

**if**(!flag)

min=34-i;

**for**(**int** j=2;j<min;j++) {

**if**(min%j==0) {

flag2=**true**;

**break**;}

}

**if**(!flag && !flag2) {

**if**(i<=min)

System.***out***.println("34="+i+"+"+min);

}

}

}}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

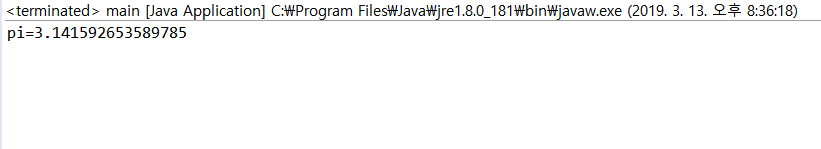
◆ DESCRIPTION

34=i+(34-i)일 때, i와 (34-i)모두 소수라는 의미이다. flag변수는 i가 소수인지 확인하는 변수로 false값이 계속 유지되면 소수이다. flag2는 (34-i)가 소수인지 확인하는 변수로 false값이 계속 유지되면 소수이다. 먼저 i가 3부터 33일 때 까지 반복하는 반복문을 만들었다. 반복문 안에는 j가 2부터 i-1까지 반복하는 반복문이 있다. 이 안쪽 반복문에서 i%j의 값이 0이 나온다면 i가 소수가 아닌 것이니, ture를 반환하고 반복문을 빠져 나온다. 만약 끝까지 나머지가 0이 나오지 않아 i가 소수라면 flag가 false라는 조건을 만족한다. 그리고 또 반복문 안에서 (34-i)가 소수인지 아닌지 반복을 통해 알아보는 것이다. 최종적으로 flag와 flag2가 모두 false이면 두 개의 합을 나타내는 문장을 출력한다. 위 내용을 i가 34가 될 때까지 반복한다. 감싸는 반복문 처음에 flag,flag2를 false로 다시 한 번 저장한 이유는 i와 (34-i)가 소수라고 일단 간주한 후에 소스가 진행되기 때문이다. 만약 이 소스가 없으면 flag or flag2값이 true일 때 값이 더 이상 바뀌지 않게 된다.

< JAVA PROGRAMMING LAB -1장: JAVA 출력문>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**6. 파이를 계산하는 다음과 같은 수식을 사용하면 약 300자리까지 계산할 수 있는 것으로 알려져 있다.**

**이 수열을 반복 루프를 사용하여서 게산하는 프로그램을 작성하고 테스트하라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**public** **class** main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**double** s=3;

**int** dis=1;

**for**(**double** n=2;n<=100000;n+=2) {

**if**((dis%2)!=0)

s+=(4/(n\*(n+1)\*(n+2)));

**else**

s-=(4/(n\*(n+1)\*(n+2)));

dis++;

}

System.***out***.println("pi="+s);

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

◆ DESCRIPTION

합을 결정하는 변수 s를 실수형으로 선언 후 3으로 초기화하였다. s에 합을 할지 뺄셈을 할지 결정하는 정수형 변수 dis를 1로 초기화하였다. 반복을 어디까지 할지 문제에 없어서 100000이 될 때까지로 정했다. 만약 dis에 2를 나눈 나머지가 0이 아니면, 즉 홀수이면 (4/(n\*(n+1)\*(n+2))을 더한다. 만약 그렇지 않아 짝수이면 뺀다. 위 사항을 반복한다. 참고로 반복문 소괄호 안에서 n을 실수형으로 선언한 이유는 (4/(n\*(n+1)\*(n+2))값을 실수로 하기 위함이다. java에서 정수에 실수를 연산하면 실수의 결과가 나오는 것을 이용하였다. 반복을 마치면 pi값 s를 출력할 수 있다.