**JAVA 프로그래밍 Lab #1**

담당 교수님 : 천인국 교수님

이름 : 현 지 원

학과 : 컴퓨터공학과

학번 : 20174069

1. JDK 9.0을 설치하고 javac와 java 프로그램을 사용하여 다음과 같은 프로그램을 입력하고 컴파일, 실행해본다.

Source Code #1

**public** **class** Pi {

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

**double** radius = 5.0;

**double** area;

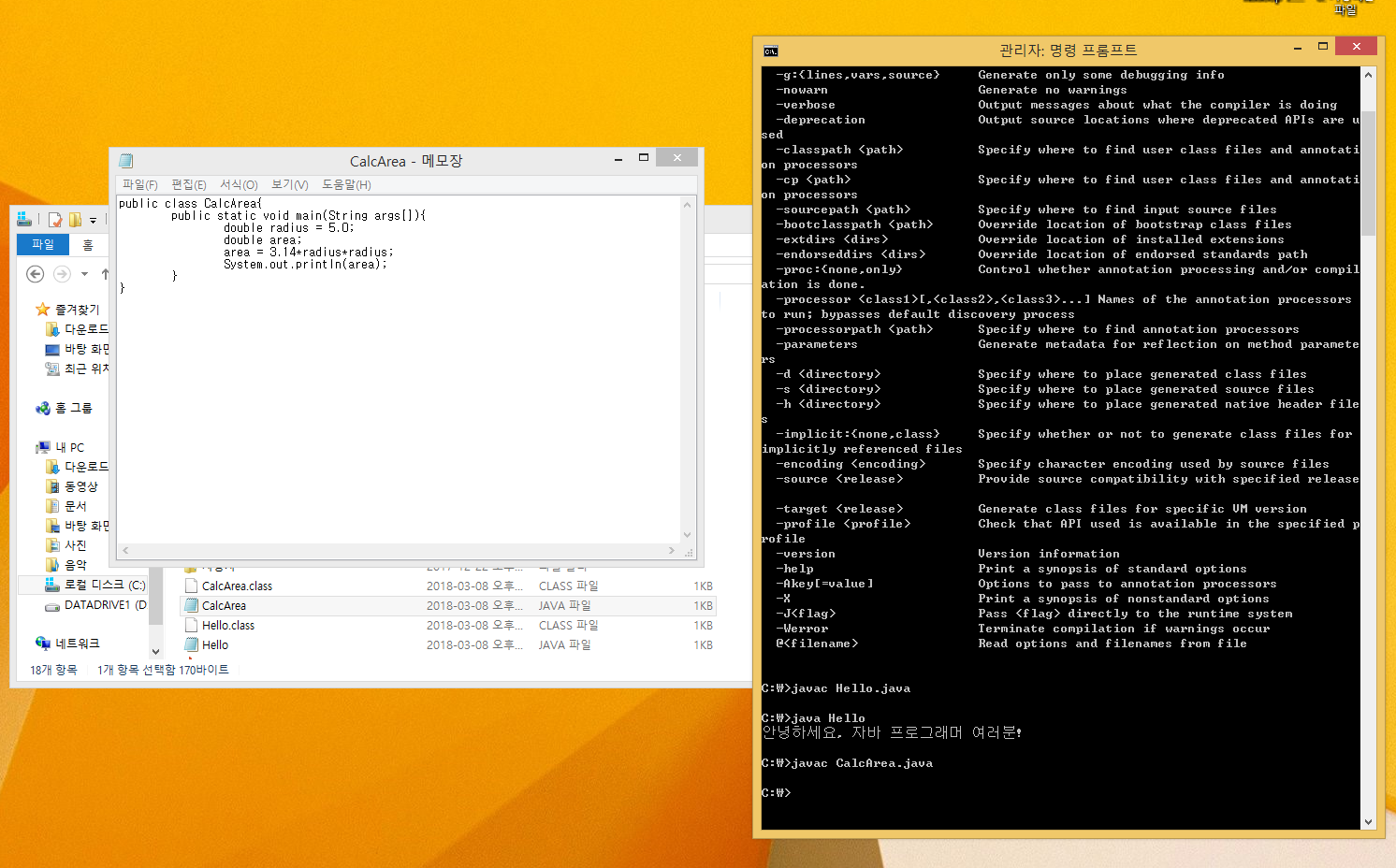
area = 3.14\*radius\*radius;

System.***out***.println(area);

}

}

실행결과



**해설**

- JDK는 명령어 프롬프트에서 직접 사용이 가능한 몇 가지의 명령어 도구들을 가지고 있고 자바 프로그램이 실행되는 원리를 알기 위해 해보아야 한다. JDK에 포함된 명령어 도구를 사용하기 위해서는 환경변수 PATH의 값을 변경해 주어야 한다.

메모장에 소스 코드를 입력하고 확장자를 .java로 저장하여 준다. cmd창에서 자바 컴파일러로 컴파일 한다. 자바 컴파일러의 이름은 javac이다. 컴파일러가 생성하는 파일을 클래스 파일이라고 한다. cmd창에 dir 명령어를 입력하여 클래스 파일이 생성되었는지 확인한다.

자바 프로그램을 성공적으로 컴파일 하였다면, 가상기계인 java를 이용하여 클래스 파일을 실행시켜준다. 자바 프로그램을 실행시키려면 클래스 이름만 입력하여야 한다.

2. Eclipse를 설치하고 Eclipse를 사용하여 1번 프로그램을 입력하고 컴파일, 실행해본다.

Source Code #2

**public** **class** Pi {

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

**double** radius = 5.0;

**double** area;

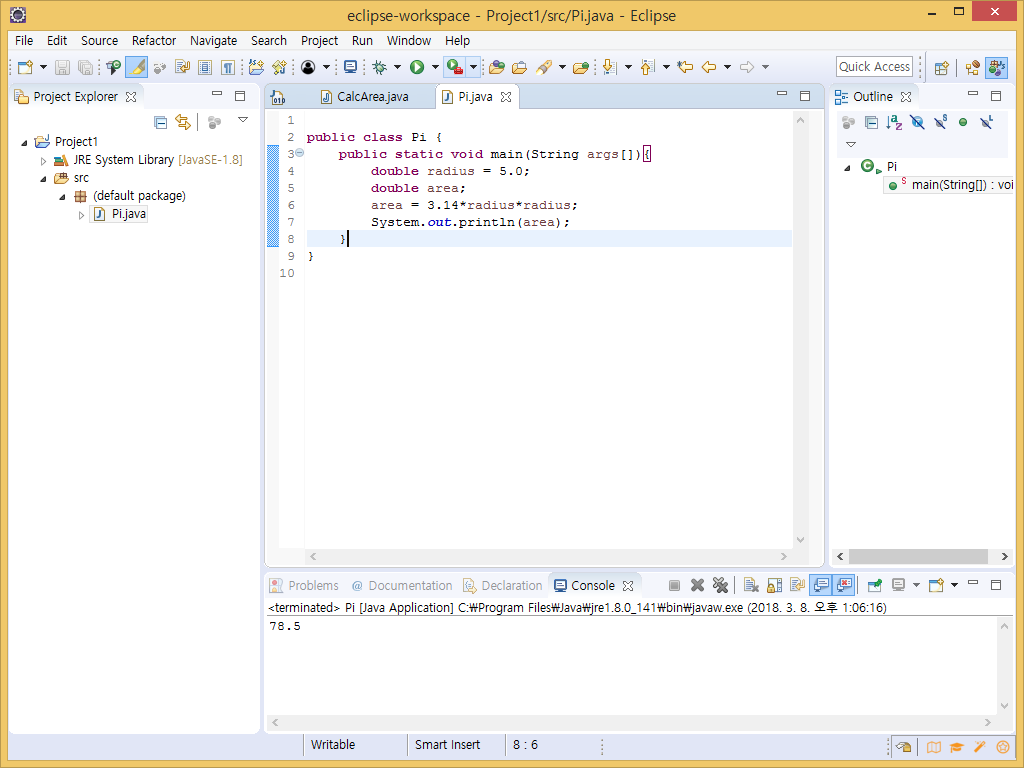
area = 3.14\*radius\*radius;

System.***out***.println(area);

}

}

실행결과



**해설**

- 2번 문제는 Eclipse를 설치하고 Eclipse를 사용하여 1번 프로그램을 입력하고 컴파일, 실행해보는 것이었다. Eclipse는 자바 프로그램을 쉽게 개발하기 위한 통합 개발 환경의 하나이다. Eclipse 페이지에서 다운을 받아서 실행한다. 프로젝트와 클래스를 생성해 준다. 클래스는 프로젝트 안에 작성하는 것으로 클래스들이 모여서 자바 프로그램이 된다.

2번째 줄의 **public** **class** Pi 문장은 Pi 클래스를 정의하는 문장이다. 자바에서 소스 파일 이름은 항상 public이 붙은 클래스의 이름과 동일하여야 한다. { }안에는 메소드를 정의하는 문장들이 작성되어있다. Public은 이 클래스는 누구든지 사용 가능 하다는 의미이다.

**public** **static** **void** main(String args[]) 이 문장은 메소드 main이 시작되는 문장이다. 메소드는 어떤 특정한 기능을 수행하는 코드들의 집합이다. 클래스 안에 선언된 함수가 메소드이다.

System.***out***.println(area) 이 문장은 area 값을 콘솔 창에 출력하라는 문장이다. System.***out***.println()을 이용하여 변수의 값이나 문자열을 화면에 출력할 수 있다.

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

이 수열을 반복 루프를 사용하여서 계산하는 프로그램을 작성하고 테스트하라.

Source Code #2

**public** **class** Pi {

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

**double** a = 3.0;

**double** b = 0.0;

**double** n = 2.0;

**double** result = 0;

**int** count = 1;

**for**(**int** i=1; i<=500; i++)

{

**if**(i%2==0)

{

count = -1;

a = a + count\*(4.0/((n)\*(n+1)\*(n+2)));

}

**else**

{

count = 1;

b = b + count\*(4.0/((n)\*(n+1)\*(n+2)));

}

n+=2 ;

result = a+b;

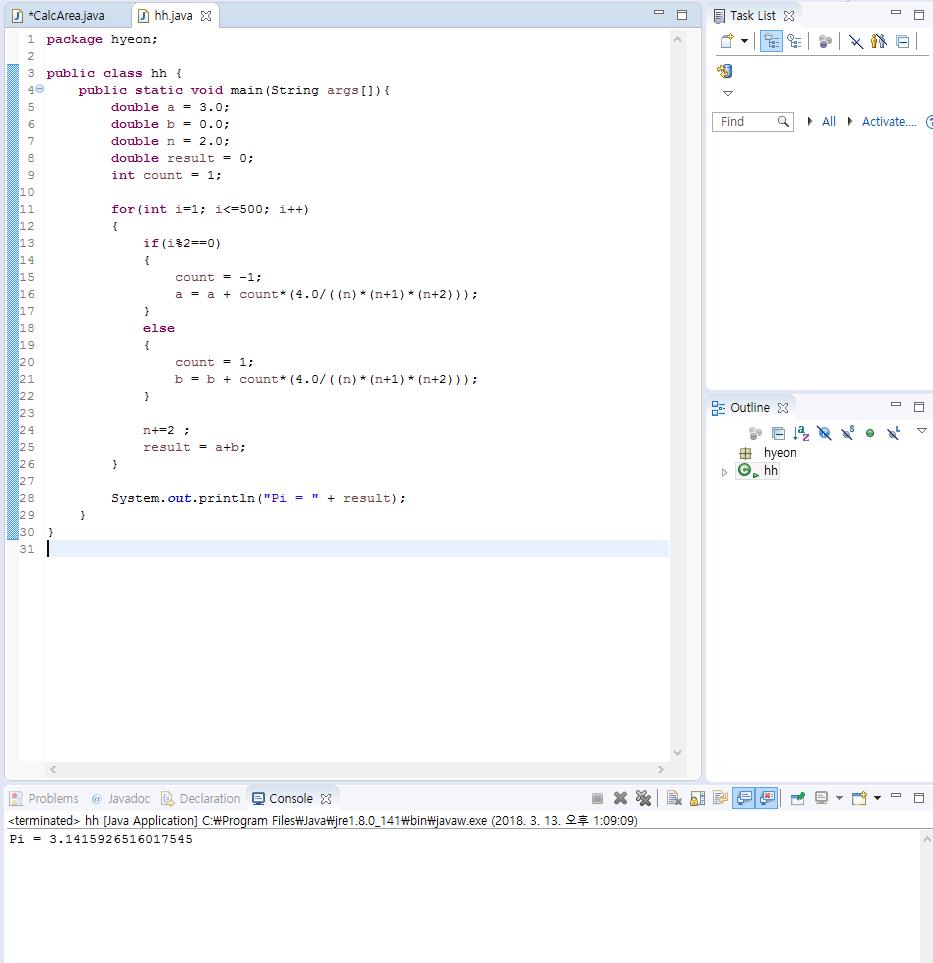
}

System.***out***.println("Pi = " + result);

}

}

실행결과



**해설**

- 3번 문제는 파이를 계산하는 수열을 반복 루프를 사용하여서 프로그램을 작성하고 테스트 하는 문제였다.

먼저 메소드 main을 작성하기에 앞서 **public** **class** Pi 문장을 써서 Pi 클래스를 정의하여 주었다. 그 다음 { }안에는 메소드를 정의하는 문장들을 작성하였다.

**public** **static** **void** main(String args[]) 이 문장은 메소드 main이 시작되는 문장이다. 변수들을 실수형 변수 double로 설정해 주었다. a는 제일 처음에 3을 더해주고 그 뒤에 짝수 번째 값들을 누적 시켜 더하는 값을 저장해 주기 위한 변수이다. b는 홀수 번째 값들을 누적시켜 더하는 값을 저장해주기 위해서 선언해 주었다. 만약에 i가 짝수로 나누어 떨어지면 count는 1에서 -1로 바뀌고 짝수 번째 식을 실행한다. I가 짝수가 아닐 경우 count는 1이고 홀수 번째 식을 계산해 준다. 조건문을 빠져 나온 다음 n의 값을 2씩 증가시켜준다. n은 분모의 값 이다. 분모가 2부터 2,3,4/ 4,5,6/ 6,7,8/ 8,9,10 ….이런 식으로 제일 첫번째 더해지는 수가 2의 배수 이기 때문에 n의 값을 2로 설정하고 n\*n+1\*n+2로 식을 작성해 주었다. 최종적으로 파이를 계산하기 위해 a와 b를 더해주어야 한다. 그 값을 result 변수에 저장해 주었다.

System.***out***.println("Pi = " + result); 이 문장을 사용하여 결과 값을 출력해 주었다. “Pi = “이라는 텍스트와 변수 result를 합하여 콘솔 창에 출력하라는 문장이다.

처음 코딩을 하고 오류가 났었다. 그 이유가 (4.0/((n)\*(n+1)\*(n+2))) 식을

(4.0/(n)\*(n+1)\*(n+2)) 이렇게 썼었는데 괄호를 한번 더 써주지 않았기 때문에 곱하기 연산자들끼리 계산할 때 오류가 난 것 이었다.