**12161581 박진성**

**<1> 소스코드 요구사항 및 설계 주안점**

Harshad 수, Armstrong 수, Perfect 수 세가지를 2 ~ 1000 숫자 범위 안에서

선별후 각각 출력하는 코드가 요구되었다.

우선, package를 두가지로 분류하였다.

1. Input 2 ~ 1000을 받아서 3가지 수를 출력할 “Main” package
2. 3가지 수를 선별하는 함수가 들어있는 “solution” package

Armstrong 수 선별함수는,

While 문으로 각 자리수를 나누어 sum 변수에 더하는 루프를 생성하였으며

루프가 한번 돌 때마다, 주어진 수는 /10 처리되어 저장된다.

Perfect 수 선별함수는,

우선 주어진 수의 약수를 구해야 하므로,

for문을 통해 약수들을 선별한뒤, 약수가 맞으면 sum 변수에 누적합을 해주었다.

**<2> 소스코드 전체**

**<Main.java>**

**package** Main;

**import** solution.Armstrong;

**import** solution.Harshad;

**import** solution.Perfect;

**public** **class** Main {

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

System.***out***.println("학번:12161581 이름:박진성");

System.***out***.println("Harshad Number:");

**for**(**int** i = 2; i <= 1000; i++) {

**if**(Harshad.*solution*(i)) {

System.***out***.print(i + " ");

}

}

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

System.***out***.println("Armstrong Number:");

**for**(**int** i = 2; i <= 1000; i++) {

**if**(Armstrong.*solution*(i)) {

System.***out***.print(i + " ");

}

}

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

System.***out***.println("Perfect Number:");

**for**(**int** i = 2; i <= 1000; i++) {

**if**(Perfect.*solution*(i)) {

System.***out***.print(i + " ");

}

}

}

}

**<Harshad.java>**

**package** solution;

**public** **class** Harshad {

**public** **static** **boolean** solution(**int** arg) {

**int** n = arg;

**int** sum = 0;

**while**(n>0) {

**int** r = n % 10;

sum = sum + r;

n = n / 10;

}

**int** answer = arg % sum;

**if**(answer == 0)

**return** **true**;

**return** **false**;

}

}

**<Armstrong.java>**

**package** solution;

**import** java.lang.Math;

**public** **class** Armstrong {

**public** **static** **boolean** solution(**int** arg) {

**int** num = arg;

**int** sum = 0;

**while**(num > 0) {

sum += Math.*pow*((num % 10), 3);

num /= 10;

}

**if**(sum == arg)

**return** **true**;

**return** **false**;

}

}

**<Perfect.java>**

**package** solution;

**public** **class** Perfect {

**public** **static** **boolean** solution(**int** arg) {

**int** sum = 0;

**for**(**int** i = 1; i < arg; i++) {

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

{

sum+= i;

}

}

sum = sum;

**if**(sum == arg)

**return** **true**;

**return** **false**;

}

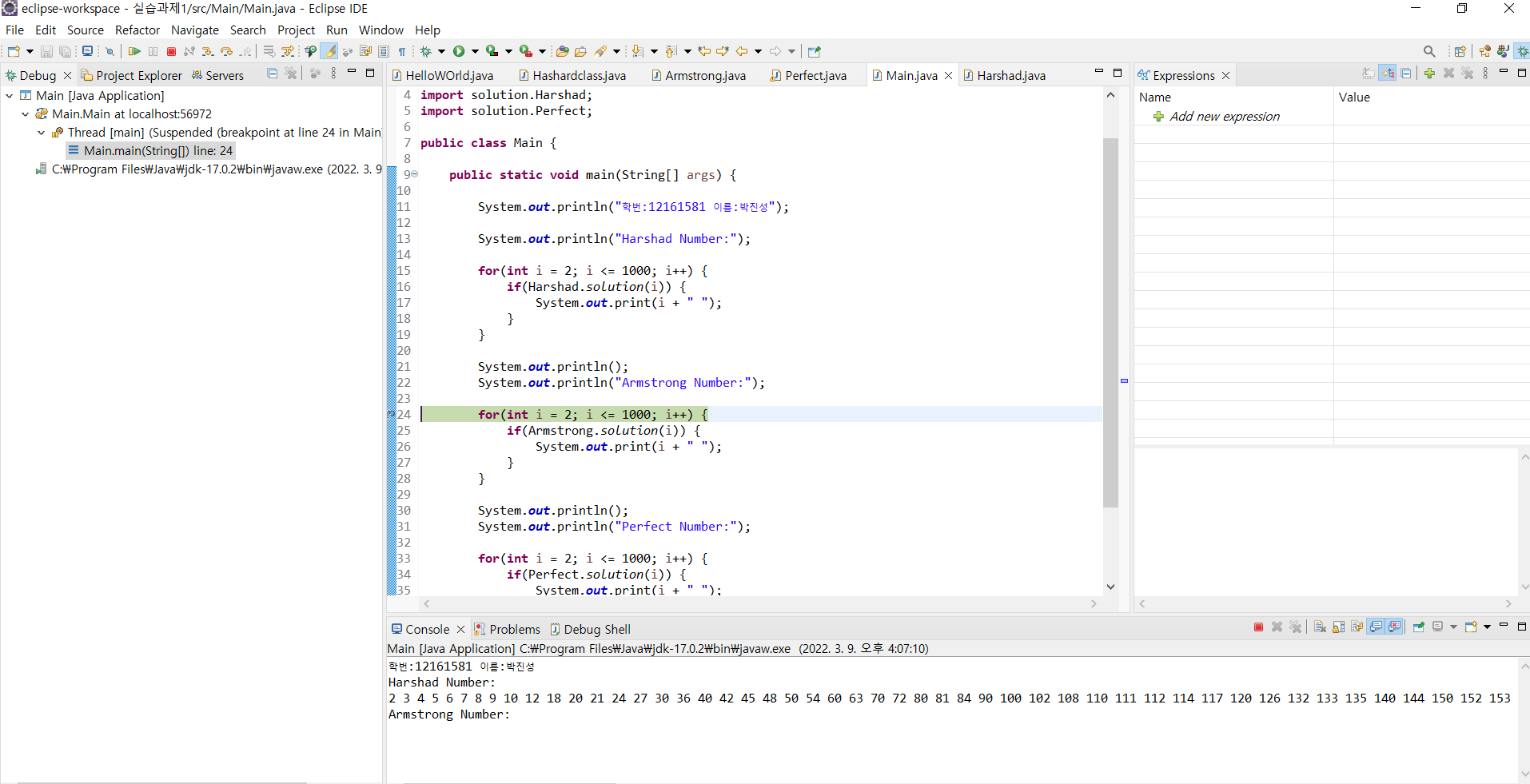
}

**<3> 실행 결과**

텍스트이(가) 표시된 사진

자동 생성된 설명

(실행 화면과 package explorer)



(디버깅 IDE)

우선 main 함수를 실행시켜, 각 solution 들에 접근하는 방식이므로

에러 발생시, 일단 break point 를 에러발생한 solution 에 설정해준다

Ctrl f11 로 디버깅을 실행시켜, f6 과 f5 를 통해

해당 solution 에서 어떤 input 일때 문제가 생기는가 에 대해 디버깅 해본다

**<4> 결론 및 배운점**

이클립스 라는 새로운 IDE 의 사용법 및, 실행, 디버깅 하는 법에 대해 체득할 수 있었다.

또한 java 라는 언어에서 출력함수 마다의 차이를 알 수 있었다.

System.out.print 함수는 띄어쓰기를 미포함한 출력

System.out.println 함수는 띄어쓰기를 포함한 출력이었다.

그리고 반복문 및 조건문은 C++ 과 매우 유사함도 느낄 수 있었다.

마지막으로 특이했던 점은, package 로 나누어서 코딩을 하고, 이를 연결하는것도

다른 언어에 비해 직관적으로 되어있는 듯 하여서, 팀프로젝트시 유용하겠다 라고 생각이 들었다.