8주차 실습과제

20135151 이갑성

1번 문제 – Mult를 추가한 Polynomial.

<소스>

**public** **class** Polynomial {

**int** arr[];

**int** max;

**int** size;

**int** degree;

**public** Polynomial(**int** max) {

**this**.max = max;

arr = **new** **int**[max];

size = 0;

degree = 0;

}

**public** **void** isEmpty() {

**if** (size > 0)

System.***out***.println("it is not empty");

**else**

System.***out***.println("it is empty");

}

**public** **int** length() {

**return** size;

}

**public** **void** insert(**int** jisu, **int** gesu) {

arr[jisu] = gesu;

size++; // size = size + 1;

**if** (jisu > degree) {

degree = jisu;

}

}

**public** **int** retrieve(**int** jisu) {

**return** arr[jisu];

}

**public** **void** replace(**int** jisu, **int** regesu) {

arr[jisu] = regesu;

}

**public** **void** delete(**int** jisu) {

arr[jisu] = 0;

size = size - 1;

}

**public** **void** Print() {

**int** i;

**for** (i = degree; i >= 0; i--) {

**if** (arr[i] != 0) {

System.***out***.print(arr[i] + "[" + i + "]");

**if** (i > 1)

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

}

}

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

}

**public** **void** addEO(Polynomial p1) {

**int** result[] = **new** **int**[100];

System.***out***.println("덧셈 결과: ");

**if** (p1.degree > degree) {

**for** (**int** i = 0; i <= p1.degree; i++) {

result[i] = p1.arr[i] + arr[i];

}

**for** (**int** j = p1.degree; j >= 0; j--) {

**if** (result[j] != 0) {

System.***out***.print(result[j] + "[" + j + "]");

**if** (j > 0) {

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

}

}

}

} **else** **if** (degree > p1.degree) {

**for** (**int** i = 0; i <= degree; i++) {

result[i] = arr[i] + p1.arr[i];

}

**for** (**int** j = degree; j >= 0; j--) {

**if** (result[j] != 0) {

System.***out***.print(result[j] + "[" + j + "]");

**if** (j > 0) {

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

}

}

}

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

}

}

**public** **void** mult(Polynomial p1) {

**int** result[] = **new** **int**[100];

**int** temp[] = **new** **int**[100];

**int** jisu;

**int** gesu;

**int** muldegree = 0;

System.***out***.println("곱셈결과: ");

**if** (degree >= p1.degree) {

**for** (**int** i = degree; i >= 0; i--) {

**for** (**int** j = p1.degree; j >= 0; j--) {

jisu = i + j;

**if** (jisu > muldegree)

muldegree = jisu;

gesu = arr[i] \* p1.arr[j];

temp[jisu] = gesu;

}

**for** (**int** k = muldegree; k >= 0; k--) {

result[k] = temp[k] + result[k];

}

temp = **new** **int**[100]; // 배열의 초기화

}

**for** (**int** l = muldegree; l >= 0; l--) {

**if** (result[l] != 0) {

System.***out***.print(result[l] + "[" + l + "]");

**if** (l > 1)

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

}

}

} **else** **if** (p1.degree >= degree) {

**for** (**int** i = p1.degree; i >= 0; i--) {

**for** (**int** j = degree; j >= 0; j--) {

jisu = i + j;

**if** (muldegree < jisu)

muldegree = jisu;

gesu = p1.arr[i] \* arr[j];

temp[jisu] = gesu;

}

**for** (**int** k = muldegree; k >= 0; k--) {

result[k] = result[k] + temp[k];

}

temp = **new** **int**[100];

}

**for** (**int** l = muldegree; l >= 0; l--) {

**if** (result[l] != 0) {

System.***out***.print(result[l] + "[" + l + "]");

**if** (l > 1)

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

}

}

}

}

}

**import** java.util.Scanner;

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

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

Scanner sc = **new** Scanner(System.***in***);

**int** max;

System.***out***.println("배열의 크기 값을 입력하세요: ");

max = sc.nextInt();

Polynomial p1 = **new** Polynomial(max);

Polynomial p2 = **new** Polynomial(max);

Polynomial p3 = **new** Polynomial(max);

Polynomial p4 = **new** Polynomial(max);

p1.insert(0, 3);

p1.insert(1, 5);

p1.insert(5, 4);

p1.isEmpty();

System.***out***.println("length " + p1.length());

System.***out***.println(p1.retrieve(5));

p1.replace(2, 6);

p1.insert(2, 1);

System.***out***.println("length " + p1.length());

p1.delete(1);

System.***out***.println("length " + p1.length());

System.***out***.println(p1.retrieve(2));

p2.insert(10, 3);

p2.insert(8, 5);

p2.insert(5, 4);

p2.insert(2, 2);

p2.insert(0, 7);

p1.Print();

p2.Print();

p2.addEO(p1);

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

p3.insert(3, 2);

p3.insert(2, 3);

p3.insert(0, 6);

p3.Print();

p4.insert(3, 4);

p4.insert(1, 2);

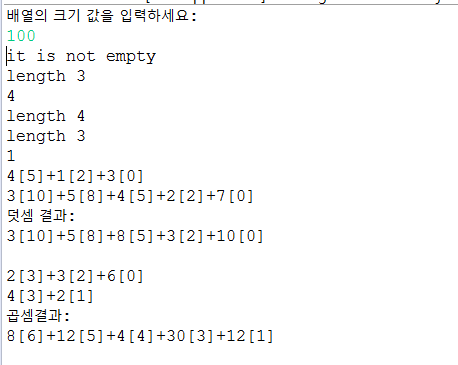
p4.Print();

p4.mult(p3);

}

}

<결과>



2번문제 – 객체 Polynomial 안에 계수와 지수 들어간 유형.

<소스>

**public** **class** Polynomial {

**private** **int** MAX;

**private** Term terms[];

**private** **int** noOfTerms; // = size; 객체개수(= 원소개수)

**int** degree; // 최고차항을 저장하는 변수

**public** Polynomial() {

**int** MAX = 10;

terms = **new** Term[MAX];

noOfTerms = 0;

degree = 0;

}

**public** **boolean** ispZero() {

**if** (noOfTerms > 0)

**return** **false**;

**else**

**return** **true**;

}

**public** **int** maxJisu() {

**for** (**int** i = 0; i < noOfTerms; i++) {

**if** (degree < terms[i].getJisu())

degree = terms[i].getJisu();

}

**return** degree;

}

**public** **void** addTerm(**int** gesu, **int** jisu) {

terms[noOfTerms] = **new** Term(gesu, jisu);

// terms 배열에 객체생성.

noOfTerms = noOfTerms + 1;

}

**public** **void** delTerm(**int** jisu) {

**for** (**int** i = 0; i < noOfTerms; i++) {

**if** (terms[i].getJisu() == jisu) {

terms[i].setJisu(0);

terms[i].setGesu(0);

**for** (**int** k = i + 1; k < noOfTerms; k++)

terms[k - 1] = terms[k];

}

}

noOfTerms = noOfTerms - 1;

}

**public** **void** Print() {

**for** (**int** i = 0; i < noOfTerms; i++) {

System.***out***.print(terms[i].getGesu() + "[" + terms[i].getJisu() + "]");

**if** (i < noOfTerms - 1)

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

}

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

}

**public** **int** findIndex(**int** jisu) {

**for** (**int** i = 0; i < noOfTerms; i++) {

**if** (terms[i].getJisu() == jisu)

**return** terms[i].getGesu();

}

**return** 0;

}

**public** **int** findNextIndex(**int** jisu) {

**for** (**int** i = 0; i < noOfTerms; i++) {

**if** (terms[i].getJisu() == jisu)

**return** terms[i + 1].getJisu();

}

**return** 0;

}

**public** Polynomial polyAdd(Polynomial p) {

**int** p1index = 0, p2index = 0, reindex = 0;

Polynomial result = **new** Polynomial();

**while** (p1index < noOfTerms && p2index < p.noOfTerms) {

**if** (p.terms[p2index].getJisu() > terms[p1index].getJisu()) {

result.terms[reindex] = **new** Term(p.terms[p2index].getGesu(), p.terms[p2index].getJisu());

p2index++;

reindex++;

} **else** **if** (p.terms[p2index].getJisu() < terms[p1index].getJisu()) {

result.terms[reindex] = **new** Term(terms[p1index].getGesu(), terms[p1index].getJisu());

p1index++;

reindex++;

} **else** **if** (p.terms[p2index].getJisu() == terms[p1index].getJisu()) {

result.terms[reindex] = **new** Term(terms[p1index].getGesu() + p.terms[p2index].getGesu(),

p.terms[p2index].getJisu());

p1index++;

p2index++;

reindex++;

}

}

// 비교를 다하고 남은 값을 출력해주기 위해서 아래의 소스를 이용.

// 예를 들면 위에 두개의 다항식을 비교하면 나중에는 3[0]만 남게되는데 그것을 출력하는 소스이다.

**if**(terms[p1index] != **null**)

{

**while**(terms[p1index] != **null**)

{

result.terms[reindex] = **new** Term(terms[p1index].getGesu(), terms[p1index].getJisu());

reindex++;

p1index++;

}

}

**else** **if**(p.terms[p2index] != **null**)

{

**while**(p.terms[p2index] != **null**)

{

result.terms[reindex] = **new** Term(p.terms[p2index].getGesu(), p.terms[p2index].getJisu());

reindex++;

p2index++;

}

}

result.noOfTerms = reindex; // Polynomial 의 속성을 갖는 result에 noOfTerms도 값을

// 지정해주어야만

// Print할때 어디까지 출력을 할지 알 수 있다.

// 그래서 result의 noOfTerms(result.noOfTerms)에

// reindex를 넣어준다.

**return** result;

}

**public** Polynomial Mult(Polynomial p) {

Polynomial result = **new** Polynomial();

Polynomial temp = **new** Polynomial();

**int** teGesu, teJisu;

**for** (**int** i = 0; i < noOfTerms; i++) {

**for** (**int** j = 0; j < p.noOfTerms; j++) {

teGesu = terms[i].getGesu() \* p.terms[j].getGesu();

teJisu = terms[i].getJisu() + p.terms[j].getJisu();

temp.terms[j] = **new** Term(teGesu, teJisu);

}

temp.noOfTerms = p.noOfTerms;

result = result.polyAdd(temp);

}

**return** result;

}

}

**public** **class** Term {

**private** **int** jisu;

**private** **int** gesu;

**public** Term(**int** gesu, **int** jisu) {

**this**.gesu = gesu;

**this**.jisu = jisu;

}

**public** **void** setGesu(**int** gesu) {

**this**.gesu = gesu;

}

**public** **void** setJisu(**int** jisu) {

**this**.jisu = jisu;

}

**public** **int** getJisu() {

**return** jisu;

}

**public** **int** getGesu() {

**return** gesu;

}

}

**public** **class** PolyTest {

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

Polynomial p1 = **new** Polynomial();

p1.addTerm(6, 4);

p1.addTerm(2, 3);

p1.addTerm(3, 0);

System.***out***.println("첫번째 다항식");

p1.Print();

Polynomial p2 = **new** Polynomial();

p2.addTerm(2, 4);

p2.addTerm(4, 2);

p2.addTerm(2, 1);

System.***out***.println("두번째 다항식");

p2.Print();

Polynomial r = p1.polyAdd(p2);

System.***out***.println("첫번째 다항식과 두번째 다항식의 합");

r.Print();

r = p1.Mult(p2);

System.***err***.println("첫번째 다항식과 두번째 다항식의 곱");

r.Print();

}

}

<출력>

