12주차 실습과제

20135151 이갑성

<소스>

**public** **class** polyNode {

**int** jisu;

**int** gesu;

polyNode link;

**public** polyNode()

{

**this**.gesu = 0;

**this**.jisu = 0;

**this**.link = **null**;

}

**public** polyNode(**int** gesu, **int** jisu)

{

**this**.gesu = gesu;

**this**.jisu = jisu;

**this**.link = **null**;

}

}

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

polyNode firstNode;

polyNode lastNode;

**public** **boolean** isZero() {

**if** (firstNode == **null**)

**return** **true**;

**else**

**return** **false**;

}

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

polyNode newNode = **new** polyNode(gesu, jisu);

**if** (firstNode == **null**)

{

firstNode = newNode;

}

**else**

{

polyNode p = firstNode;

polyNode q = **null**;

**boolean** ch = **false**;

**while** (p.link != **null**) // Insert역할을 하는 조건문 = > 지수를 비교해서 위치를 찾는다.

{

**if** (p.jisu > newNode.jisu) {

q = p;

p = p.link;

} **else** **if** (p.jisu < newNode.jisu) {

ch = **true**;

**break**;

}

}

**if** (ch == **true** || (ch == **false** && p == firstNode)) // if의 다른 조건문 하나는 polyNode값이 하나이고 다음노드가없는 link가 null일때

// 위에 상태일때는 while문에 해당이 안되서 따로 만들어줘야한다.

{ //노드가 하나이고, 그 노드의 지수와 새로운 노드의 지수를 비교한다.

**if**(p == firstNode && p.jisu < newNode.jisu) //기존 노드의 지수보다 새로운 노드의 지수가 큰 경우

{

polyNode temp = p;

firstNode = newNode;

firstNode.link = temp;

}

**else** **if**(p == firstNode && p.jisu > newNode.jisu) //기존 노드의 지수보다 새로운 노드의 지수가 작은 경우

{

p.link = newNode;

}

**else**

{

q.link = newNode;

newNode.link = p;

}

}

**else** **if**(p.jisu < newNode.jisu) // while문은 p.link까지가 아닐때까지 조건문이 반복된다. 마지막 노드는 비교를 하지 못한다.

{ // 여기서 p는 이미 마지막노드 이고 p와 newNode의 지수를 비교한다.

q.link = newNode;

newNode.link = p;

}

**else**{

p.link = newNode; // while문으로 p가 마지막 노드에 위치하고 마지막 노드 다음에 값 추가.

}

}

}

**public** **void** print() {

polyNode p = firstNode;

**while** (p.link != **null**) {

System.***out***.print(p.gesu + "[" + p.jisu + "]" + "+");

p = p.link;

}

System.***out***.print(p.gesu + "[" + p.jisu + "]");

}

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

{

polyNode p = firstNode;

polyNode q = **null**;

**if**(p.jisu == jisu) //맨 앞의 노드를 삭제시 다음 노드를 firstNode로 지정한다.

{

firstNode = p.link;

}

**else**

{

**while**(p.link != **null**)

{

**if**(p.jisu == jisu)

{

q.link = p.link;

**break**;

}

q = p;

p = p.link;

}

**if**(p.jisu == jisu) // 현재 p값이 마지막 노드를 향하고 있는 상태 delete하고 자하는 값이 마지막 노드일때

{

q.link = **null**;

}

}

}

**public** Polynomial polyAdd(Polynomial q)

{

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

polyNode p1 = firstNode;

polyNode p2 = q.firstNode;

**while**(p1.link != **null** && p2.link != **null**)

{

**if**(p1.jisu == p2.jisu)

{

result.addTerm(p1.gesu + p2.gesu, p1.jisu);

p1 = p1.link;

p2 = p2.link;

}

**else** **if**(p1.jisu > p2.jisu)

{

result.addTerm(p1.gesu, p1.jisu);

p1 = p1.link;

}

**else** **if**(p1.jisu < p2.jisu)

{

result.addTerm(p2.gesu, p2.jisu);

p2 = p2.link;

}

}

**if**(p1.jisu == p2.jisu)

{

result.addTerm(p1.gesu + p2.gesu, p1.jisu);

}

**else** **if**(p1.jisu > p2.jisu)

{

result.addTerm(p1.gesu, p1.jisu);

result.addTerm(p2.gesu, p2.jisu);

}

**else** **if**(p1.jisu < p2.jisu)

{

result.addTerm(p2.gesu, p2.jisu);

result.addTerm(p1.gesu, p1.jisu);

}

**return** result;

}

}

**public** **class** PolynomialTest {

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

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

p1.addTerm(10, 3);

p1.addTerm(2, 4);

p1.addTerm(3, 2);

p1.addTerm(4, 5);

p1.addTerm(5, 0);

System.***out***.println("======p1 다항식 출력======");

p1.print();

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

System.***out***.println("노드가 비어 있습니까? : " + p1.isZero());

System.***out***.println("=======지수 2 삭제========");

p1.delete(2);

System.***out***.println("======p1 다항식 출력=======");

p1.print();

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

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

p2.addTerm(3, 4);

p2.addTerm(3, 2);

p2.addTerm(2, 3);

System.***out***.println("======p2 다항식 출력======");

p2.print();

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

System.***out***.println("=====p1+p2 결과 출력 =====");

Polynomial result = p1.polyAdd(p2);

result.print();

}

}

<결과>

