자료구조 실습 보고서

[제 02주]

제출일:19.04.03

학번/이름: 201603867/조성환

1. 프로그램 설명서

1) 주요 알고리즘/ 자료구조/ 기타

ArrayBag은 4가지 기능을 담고 있다. 4가지는 add, remove, search, frequency인데, AppController에 의해 menu를 입력받고 menu에 해당되는 기능을 수행하며, 마지막에는 exit를 입력받고 끝난다.

ArrayBag은 Bag의 기능을 List로 구현한 자료구조이다. Bag이란 중복을 허용하며 원소들을 단순히 모아둔 자료구조이다. 이를 List로 구현을 하는 것이 바로 ArrayBag이다.

구현한 자료구조에는 원소 추가, 제거, 검색, 개수 조회가있다.

2) 함수 설명서

AppController

void addCoin()

this.coinBag().isFull() 함수를 통해 가방이 꽉 찼는지 확인한다. 꽉 차 있다면 차있다고 출력한다.

차있지 않으면 coinValue 변수를 AppView.inputCoinValue()함수를 통해 값을 받아온다.

this.coinBag().add(new Coin(coinValue)) 함수를 통해 coinBag에 Coin을 추가한다. Coin은 coinValue값을 넣어 객체를 생성한다.

만약 성공하면 true를 반환, 추가했다는 말을 출력한다.

실패하면 false를 반환, 실패했다는 출력문을 나타낸다.

void removeCoin()

coinValue 변수를 AppView.inputCoinValue()함수를 통해 제거할 값을 받아온다.

this.CoinBag().remove(new Coin(coinValue)) 함수를 통해 Coin을 제거한다. 새로 생성한 Coin에 coinValue값을 넣어준다.

성공하면 true를 반환, 삭제했다는 출력문을 나타낸다.

실패하면 false를 반환, 주어진 값이 존재하지 않다고 나타낸다.

void searchForCoin()

coinValue 변수를 AppView.inputCoinValue()함수를 통해 검색할 값을 받아온다.

this.CoinBag().doesContain(new Coin(coinValue)) 함수를 통해 Coin을 검색한다. 새로 생성한 Coin에 coinValue값을 넣어준다.

성공하면 true를 반환, 주어진 값이 존재한다는 출력문을 나타낸다.

실패하면 false를 반환, 주어진 값이 존재하지 않다고 나타낸다.

void frequencyOfCoin()

coinValue 변수를 AppView.inputCoinValue()함수를 통해 검색할 값을 받아온다.

int frequency= this.CoinBag().frequencyOf(new Coin(coinValue)) 함수를 통해 Coin을 검색한다. 새로 생성한 Coin에 coinValue값을 넣어준다.

성공한 개수 만큼 값을 늘려준 후 출력한다.

void undefinedMenuNumber(int aMenuNumber)

입력받은 munuNumber가 찰못됐을 시, 해당 숫자와 함께 잘못된 번호임을 출력한다.

int sumOfCoinValues()

sum변수를 0으로 선언한다
for문을 coinBag.size()만큼 돌린다.
해당 위치에 있는 값을 sum에 계속 추가한다.
마지막까지 돌린 후 나온 sum을 리턴한다.

in maxCoinValue()
maxValue를 0으로 초기화한다.
for문을 coinBag.size()만큼 돌린다.
maxValue가 해당 위치의 값보다 작을 때 maxValue를 해당 값으로 바꿔준다.
마지막까지 돌린 후 나온 maxValue를 리턴한다.

void showStatistics()
3가지를 출력한다.
동전의 개수를 this.coinBag().size()
가장 큰 값을 this.maxCoinValue()
모든 동전 값의 합을 this.sumOfCoinValues()를 통해 출력한다.

.ArrayBag생성자(int givenCapacity)
Arraybag객체를 생성할 때 최대 수용할 수 있는 개수를 입력받은 후 개수에 맞는 element를 set한 후 size를0으로 만든다.

.isEpmty()
Bag내 size가 0인지를 검사한다 맞으면 true를 반환한다.

.isFull()
Bag내 size와 capacity를 비교한다. 비교 후 size capacity는 가방이 담을 수 있는 최대 개수이다.

.doesContain(E anElement)
found라는 boolean변수를 하나 false로 초기화 선언한 후 for문을 통해 size크기 만큼 계속 내용물을 참조한다. 만일 동일한 원소(.equals)가 존재한다면 found를 true로 초기화 한 후 break를 통해 for문을 나온다.
마지막으로 found값을 리턴한다.

.frequencyOf(E anElement)
frequencyCount라는 int형 변수를 하나 0으로 초기화 선언한 후 for문을 통해 size크기 만큼 계속내용물을 참조한다. 만일 동일한 원소(.equals)가 존재한다면 frequencyCount를 1 추가한다. 내용물을 모두 참조한 뒤 frequencyCount값을 리턴한다.

.add(E anElement)
if 문을 통해 Bag가 꽉 찼는지를 검사한다. 만일 꽉차있으면 false를 리턴한다.
else 문을 통해서 추가를 해준다. size에 해당하는 값의 배열 위치에 anElement를 넣고 size를 1 더해준다. 마지막으로 true를 리턴한다.

.remove(E anElement)

foundIndex라는 int형 변수를 -1로 초기화 선언, found라는 boolean형 변수를 false로 초 기화 선언한다.
for문을 통해 size크기 만큼 계속내용물을 참조한다. 만일 동일한 원소(.equals)가 존재한다면 해당 I를 foundIndex에 초기화해주고 found를 true로 초기화해준다. 만일 찾으면 for문 조 건에 !found가 있기 때문에 for문을 나오게된다.
이후 if문에서 !found가 true, 즉, found가 false일 경우 false값을 리턴해준다.
else문에서 for문을 통해 해당 위치(foundIndex)+1의 값을 foundIndex위치의 값에 넣어주 고 foundIndex가 size-1이 될 때까지 반복한다.
마지막으로 size-1위치의 값이 2개이기 때문에 해당 값을 null로 해주고 size를 -1해준다. 이후 true를 리턴해준다.

3) 종합 설명서

ArrayBag은 Bag자료구조를 List로 구현한 형태이며, 기능은 Bag와 마찬가지로 원소들을 넣 어두고 모아두는 역할을 한다. 구현한 기능으로는 add, remove,
mvc모델을 이용해 model, view, controller를 분리시켰다.
model에는 ArrayBag, Coin 클래스가있고
view에는 AppView 클래스가 있고
controller에는 AppController, main클래스가있다.

model.ArrayBag 클래스에서는 AppController.run에 의해 menunumber값을 입력받고 받 은 입력에 따라 add, remove, frequency, elementAt의 메소드를 실행한다. 출력은 AppView에서 진행된다.

2. 프로그램 장단점/ 특이점 분석

예외처리를 하지 않았으며, 강의슬라이드에는 removeany, any 등 여러 메소드를 담고있으 나 가장 간단한 4가지 기능만 구현을 해놓은 것이 단점이다.

3. 실행 결과 분석

1) 입력과 출력

```
<<< 동전 가방 프로그램을 시작합니다 >>>
동전 가방의 크기, 즉 가방에 들어갈 동전의 최대 개수를 입력하시오: 10
수행하려고 하는 메뉴 번호를 선택 하시오(add: 1, remove: 2, search: 3, frequency: 4, exit: 9) :  1
동전 값을 입력하시오: 3
- 주어진값을 갖는 동전을 가방에 성공적으로 넣었습니다.
수행하려고 하는 메뉴 번호를 선택 하시오(add: 1, remove: 2, search: 3, frequency: 4, exit: 9) :  2
동전 값을 입력하시오: 2
-주어진 값을 갖는 동전은 가방안에 존재하지 않습니다.
수행하려고 하는 메뉴 번호를 선택 하시오(add: 1, remove: 2, search: 3, frequency: 4, exit: 9) :  3
동전 값을 입력하시오: 4
- 주어진 값을 갖는 동전은 가방 안에 존재하지 않습니다.
수행하려고 하는 메뉴 번호를 선택 하시오(add: 1, remove: 2, search: 3, frequency: 4, exit: 9) :  5
- 선택된 메뉴 번호5 는 잘못된 번호입니다.
수행하려고 하는 메뉴 번호를 선택 하시오(add: 1, remove: 2, search: 3, frequency: 4, exit: 9) :  9
가방에 들어있는 동전의 개수: 1동전 중 가장 큰 값: 3모든 동전 값의 합: 3<<< 동전 가방 프로그램을 종료합니다 >>>

Process finished with exit code 0
```

2) 결과 분석

menu number 1234,9 그리고 잘못된 입력에 대해서는 제대로 처리를 하고 있으며,
잘못된 동전의 처리에 대해서도 잘 처리되어있다.

4. 소스 코드

.AppController.run

```
int coinBagSize= AppView.inputCapacityOfCoinBag();
this.setCoinBag(new ArrayBag<>(coinBagSize));

int menuNumber= AppView.inputMenuNumber();
while (menuNumber!= MENU_END_OF_RUN){
    switch (menuNumber){
        case MENU_ADD:
            this.addCoin();
            break;

        case MENU_REMOVE:
            this.removeCoin();
            break;

        case MENU_SEARCH:
            this.searchForCoin();
            break;

        case MENU_FREQUENCY:
            this.frequencyOfCoin();
            break;

        default:
            this.undefinedMenuNumber(menuNumber);
            break;
    }
    menuNumber= AppView.inputMenuNumber();
}
this.showStatistics();
AppView.outputLine("<<< 동전 가방 프로그램을 종료합니다 >>>");
```

.ArrayBag

```java
public boolean doesContain(E anElement){
    boolean found= false;
    for(int i= 0; i< this.size(); i++){
        if(this.elements()[i].equals(anElement)){
            found= true;
            break;
        }
    }
    return found;
}
//주어진 원소가 bag에 몇개 있는지 알려준다.
public int frequencyOf (E anElement){
    int frequencyCount= 0;
    for(int i= 0; i< this._size; i++){
        if(this.elements()[i].equals(anElement)){
            frequencyCount ++;
        }
    }
    return frequencyCount;
}
public boolean add(E anElement){
    if(this.isFull()){
        return false;
    }else {
        this.elements()[this.size()]= anElement;
        this.set_size(this.size()+1);
        return true;
    }
}
```

```java
private E[] elements() {
    return this._elements;
}
private void set_elements(E[] newElements) {
    this._elements = newElements;
}

public int size() { return this._size;}
private void set_size(int newSize) { this._size = newSize; }

private int capacity() { return this._capacity; }
private void set_capacity(int newCapacity) { this._capacity = newCapacity; }


//생성자
@SuppressWarnings("unchecked")
public ArrayBag(){
    this.set_capacity(ArrayBag.DEFAULT_CAPACITY);
    this.set_elements((E[])new Object[this._capacity]);
    this.set_size(0);
}
/unchecked/
public ArrayBag(int givenCapacity){
    this.set_capacity(givenCapacity);
    this.set_elements((E[])new Object[this._capacity]);
    this.set_size(0);

}
```