자료구조 실습 보고서

[제 02 주] ArrayBag 을 사용하는 동전가방

제출일	2017/03/19
학 번	201000287
소 속	일어일문학과
이 름	유 다훈

3 프로그램 설명서

1 주요 알고리즘 및 자료구조

- 알고리즘
  - 입력
    - 처음 가방에 들어갈 최대 가방 사이즈를 입력한다.
    - 1 이 입력되면 코인 액수를 입력 받아 가방에 넣는다
    - 2 가 입력되면 코인의 액수를 입력 받아 가방에서 해당 코인을 삭제한다.
    - 3 이 입력되면 총 코인의 개수와 가장 큰 코인의 값, 현재 있는 코인의 총 금액을 출력한다.
    - 4 가 입력되면 코인의 액수를 입력 받아 해당 코인을 검색하여 개수를 출력한다.
    - 출력
    - 각 기능에 해당하는 숫자가 입력되면 각각 총 코인의 개수, 가장 큰 코인의 값, 가방에 있는 코인의 합을 출력한다.
- 자료구조
  - 코인을 담는 가방의 역할을 하는 Coin 타입의 1 차원 배열
  - 프로그램 실행, 가방의 크기를 결정하는 총 코인 갯수, 메뉴, 메뉴 종료, 프로그램 종료시에 각각 쓰이는 enum 형태의 자료구조

2 함수 설명서

Class	AppController			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	run()	없음	없음	동전 가방 프로그램을 실행시키는 메소드
	showMessage(MessageID aMessageID)	에러메세지 혹은 알림	없음	에러메세지 혹은 알림에 따라 메세지를 출력하는 메소드

Class	AppView			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppView()	없음	없음	생성자 메소드
	int inputInt()	없음	입력받은 코인값	사용자로부터 코인값을 입력받아 리턴
	void outputResult(int aTotalCoinSize, int aMaxCoinValue, int aSumOfCoinValue)	총 코인갯수, 제일 큰 코인값, 총 코인값의 합	없음	총 코인의 갯수, 가장 큰 코인의 값 총 코인값의 합을 출력
	void outputMessage(String aMessageString)	출력하고 싶은 문자열	없음	메세지 출력 메소드.
	void outputSearch(int aSearchValue, int aSearchedSize)	찾으려는 코인값 찾으려는 코인값의 갯수	없음	찾으려는 코인의 값과 해당 값을 가진 코인이 몇 개 있는지 출력.

Class	Coin			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	Coin(int givenValue)	코인값	없음	주어지는 값으로 코인의 값을 정하는 생성자 메소드
	int value()	없음	코인값 리턴	코인 값을 반환
	void setValue(int newValue)	저장하고싶은	없음	코인값을 새롭게 저장

		새로운 코인값		
	boolean equals(Coin aCoin)	코인	입력된 코인의 값과 현재 코인이 값이 같으면 참, 아니면 거짓.	입력받은 코인값과 가방안에 있는 코인들 중 같은 값이 있는지 확인하려할 때 사용.

Class	ArrayBag			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	ArrayBag()	없음	없음	가방의 크기를 디폴트값으로 설정하는 기본 생성자 메소드
	ArrayBag(int givenMaxSize)	가방의 최대 크기	없음	가방에 코인이 들어갈 수 있는 량을 주어지는 파라미터값 만큼으로 설정하는 생성자 메소드
	int size()	없음	가방 내의 코인이 들어가있는 사이즈 리턴	가방 안에 들어가 있는 코인의 갯수를 리턴
	boolean isEmpty()	없음	가방이 비어있으면 참, 아니면 거짓	가방이 비어있는지 비어있지 않은지 검사
	boolean isFull()	없음	가방이 꽉 차있으면 참, 아니면 거짓	가방이 꽉 차있는지 아닌지를 검사
	boolean doesContain(Coin anElement)	코인	가방안에 해당 코인이 있으면 참, 아니면 거짓	가방 안에 특정 코인이 들어있는지 없는지를 검사
	int frequencyOf(Coin anElement)	코인	특정 값을 가진 코인의 갯수	가방 안에 특정 값을 가진 코인의 개수를 리턴
	int maxElementValue()	없음	가방 안에 들어있는 코인들 중 최대값	가방 안에 모든 코인들을 조사해서 제일 큰 코인 값을 리턴
	int sumElementValues()	없음	가방 안에 들어있는 모든 코인 값의 합	가방 안의 모든 코인의 값을 더해서 리턴
	boolean add(Coin anElement)	코인	가방이 꽉 차있으면 거짓, 가방에	가방 안에 공간이 없으면 추가하지말고, 공간이 있으면 주어진 코인을 더함

			공간이 있으면 참	
	boolean remove(Coin anElement)	코인	주어진 코인을 가방 안에서 못찾으면 거짓, 찾으면 참	가방 안에 주어진 코인을 찾으면, 해당 코인을 삭제하고 나머지 값들을 한 칸씩 앞으로 당긴다. 또한 코인이 들어있는 총 갯수를 알리는 size 를 줄인다.
	void clear()	없음	없음	가방 안의 모든 코인을 비운다.

3 종합 설명서

- 프로그램 시작 후 가방에 들어갈 총 코인의 개수를 설정한다.
- 프로그램이 종료될 때까지 수행하고자하는 메뉴를 입력한다.
- 1 번을 입력하면 가방에 넣고자 하는 코인 값을 입력한다.
- 2 번을 입력하면 가방에서 지우고자하는 코인 값을 입력한다.
- 3 번을 입력하면 현재 가방 안에 있는 코인의 총 개수, 가장 값이 큰 코인, 총 코인 값의 합을 출력한다.
- 4 번을 입력하면 찾고자 하는 코인의 값을 입력하고, 해당 값을 가진 코인의 개수를 출력한다.
- 9 번을 입력하면 가방 안에 있는 코인의 총 개수, 가장 값이 큰 코인, 총 코인 값의 합을 출력 한 후 프로그램을 종료한다.

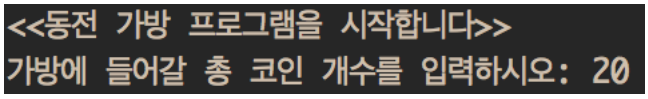
4 프로그램 장단점 분석

- 장점
  - 배열을 이용하여 Bag 이라는 자료구조 형태를 구현해볼 수 있다.
  - Bag 에서 내가 원하는 값을 찾아서 삭제, 조회를 해볼 수 있다.
- 단점
  - 가방 내부에 아이템을 검색할 때 x 번째에 어떤 아이템이 들어 있는지 알 수 없다
  - 가방 내부에 특정 아이템이 몇 번째 순서에 들어있는지 알 수 없다.
  - 아이템을 배열 중간에서부터 넣을 수는 없다.
  - 메뉴에 표시된 번호 이외의 번호를 입력했을 때 오류 메세지를 출력할 수 없다.

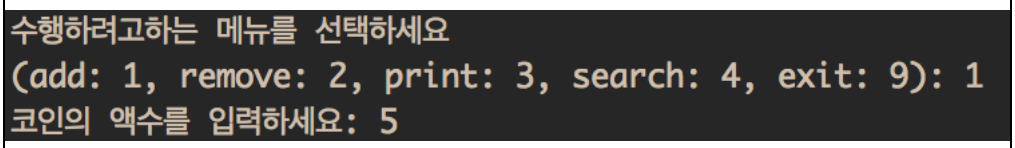
4 실행 결과 분석

1 입력과 출력

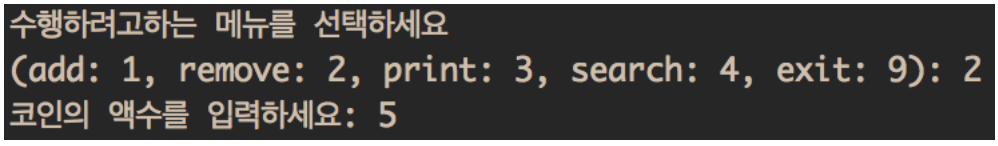
입력 - 가방에 들어갈 총 코인 개수 입력



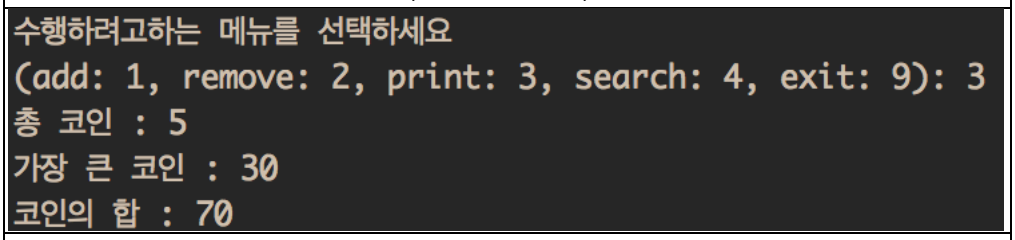
입력 - 가방에 코인을 넣고자 할 때



입력 - 가방의 특정 코인 값을 삭제하고자 할 때



입력 - 가방 내부 총 코인, 가장 큰 코인, 코인의 합을 알고자 할 때



입력 - 가방에서 특정 값을 가진 코인이 몇 개 있는지 알고자 할 때

```
수행하려고하는 메뉴를 선택하세요
(add: 1, remove: 2, print: 3, search: 4, exit: 9): 4
코인의 액수를 입력하세요: 5
5코인은 2개 존재합니다.
```

입력 - 프로그램을 종료하고자 할 때

```
수행하려고하는 메뉴를 선택하세요
(add: 1, remove: 2, print: 3, search: 4, exit: 9): 9
9가 입력되어 종료합니다.
총 코인 : 5
가장 큰 코인 : 30
코인의 합 : 70
<<동전 가방 프로그램을 종료합니다>>
```

2 결과 분석

- 가방에 들어갈 총 코인 개수를 입력했을 때 : 가방으로 쓰일 배열의 최대 크기 설정
- 1 번을 입력 했을 때 : 가방 안에 특정 값의 코인을 넣음
- 2 번을 입력 했을 때 : 가방 안에 특정 값의 코인을 삭제
- 3 번을 입력 했을 때 : 가방 안에 들어 있는 코인의 총 개수, 가장 큰 코인의 값, 모든 코인 값의 합을 출력
- 4 번을 입력 했을 때 : 가방 안에 들어 있는 특정 값을 가진 코인을 입력하고, 그 값을 가진 코인이 몇 개가 들어 있는지 출력.
- 9 번을 입력 했을 때 : 가방 안에 들어 있는 코인의 총 개수, 가장 큰 코인의 값, 모든 코인의 값의 합을 출력하고 프로그램을 종료한다.
  ⇒ 프로그램이 정상적으로 작동되는 것을 확인할 수 있다.

4 소스코드

Class	AppController

```
public class AppController {
	private AppView _appView;
	private ArrayBag _coinCollector; //코인 컨트롤러

	public AppController() {
		this._appView = new AppView();
	}

	public void run() {
		// TODO Auto-generated method stub

		int totalCoin = 0; //들어갈 수 있는총 코인의 갯수
		int input = 0; //넣으려는 코인
		int order = 0; //메뉴 번호

		this.showMessage(MessageID.Notice_StartProgram); //프로그램 시작
		this.showMessage(MessageID.Notice_InputTotalCoin);//총 코인의 개수 입력 출력

		totalCoin = this._appView.inputInt(); //총 코인의 갯수
		this._coinCollector = new ArrayBag(totalCoin); //총 코인의 갯수만큼 가방공간 생성

		while (order != 9) { //메뉴 번호가 9 가 아니라면 계속 실행
			this.showMessage(MessageID.Notice_Menu); //메뉴 출력
			order = this._appView.inputInt(); //수행하려고 하는 메뉴 입력받기
			if(order == 1) { //1 번메뉴
				this.showMessage(MessageID.Notice_InputCoin);  //넣으려는  코인의
액수
				input = this._appView.inputInt(); //액수 입력
				Coin anCoin = new Coin(input); //코인값을 저장하는 코인객체 생성
				this._coinCollector.add(anCoin); //코인값을 가방에 저장
			}
			else if (order == 2) { //2 번메뉴
				this.showMessage(MessageID.Notice_InputCoin); //코인액수 입력
				input = this._appView.inputInt();
				Coin givenCoin = new Coin(input); //입력받은 코인액수를 이용하여
코인 객체 생성
				this._coinCollector.remove(givenCoin); //입력받은 코인액수를 삭제
			} else if (order == 3) { //3 번메뉴
				this._appView.outputResult(this._coinCollector.size(),
this._coinCollector.maxElementValue(), this._coinCollector.sumElemnetValues());
				//가방에 들어있는 코인의 갯수 , 가장 큰 코인값, 모든 코인의 합
			} else if (order == 4) { //4 번 메뉴
				this.showMessage(MessageID.Notice_InputCoin); //코인값 입력
				input = this._appView.inputInt();
				Coin givenCoin = new Coin(input);
				this._appView.outputSearch(input,
this._coinCollector.frequencyOf(givenCoin));
				//입력받은 코인값이 몇개가 존재하는지 검색
			} else if (order == 9) { //9 번메뉴
				this.showMessage(MessageID.Notice_EndMenu); //메뉴를 종료
				this._appView.outputResult(this._coinCollector.size(),
this._coinCollector.maxElementValue(), this._coinCollector.sumElemnetValues());
				//결과출력. 총 코인갯수, 가장 큰 코인, 코인들의 합.
				this.showMessage(MessageID.Notice_EndProgram);  //프로그램 종료
```

```
메세지 출력
                                        System.exit(0); //프로그램 종료
                                }

                        }

                }
                private void showMessage(MessageID aMessageID) { //메세지 출력메소드
                        // TODO Auto-generated method stub
                        switch(aMessageID) {
                        case Notice_StartProgram : //프로그램 시작시
                                this._appView.outputMessage("<<동전 가방 프로그램을 시작합니다>>" + "₩n");
                                break;
                        case Notice_InputTotalCoin: //가방에 들어갈 총 코인 갯수
                                this._appView.outputMessage("가방에 들어갈 총 코인 개수를 입력하시오: ");
                                break;
                        case Notice_Menu : // 메뉴알림 및 선택
                                this._appView.outputMessage("수행하려고하는 메뉴를 선택하세요" + "₩n");
                                this._appView.outputMessage("(add: 1, remove: 2, print: 3, search: 4, exit: 9):
");
                                break;
                        case Notice_InputCoin: // 코인의 액수
                                this._appView.outputMessage("코인의 액수를 입력하세요: ");
                                break;
                        case Notice_EndMenu: // 종료합니다
                                this._appView.outputMessage("9 가 입력되어 종료합니다." + "₩n");
                                break;
                        case Notice_EndProgram: // 프로그램 종료
                                this._appView.outputMessage("<<동전 가방 프로그램을 종료합니다>>");
                                break;
                        default:
                                break;

                        }
                 }

}
```

Class	AppView

```
import java.util.Scanner;

public class AppView {

        private Scanner _scanner;

        public AppView() {
                this._scanner = new Scanner(System.in);
        }

        public int inputInt() {

                return _scanner.nextInt(); //코인을 입력받음.
        }
```

```
	public void outputResult(int aTotalCoinSize, int aMaxCoinValue, int aSumOfCoinValue) {
		System.out.println("총 코인 : " +aTotalCoinSize); // 총 코인 출력
		System.out.println("가장 큰 코인 : " +aMaxCoinValue); //가장 큰 코인
		System.out.println("코인의 합 : " + aSumOfCoinValue); //총 코인의 합

	}

	public void outputMessage(String aMessageString) {
		System.out.print(aMessageString); //메세지 출력

	}
	public void outputSearch(int aSearchValue, int aSearchedSize) {
		System.out.println(aSearchValue+"코인은 " + aSearchedSize + "개 존재합니다."); //검색하고
싶은 코인이 몇개 있는지

	}

}
```

Class	ArrayBag

```
public class ArrayBag {

	private static final int DEFAULT_MAX_SIZE = 100;
	private int _maxSize;
	private int _size;
	private Coin _elements[];


	public ArrayBag() { //가방의 기본 생성자.
		this._maxSize = DEFAULT_MAX_SIZE;
		this._elements = new Coin[DEFAULT_MAX_SIZE];
		this._size = 0;

	}

	public ArrayBag(int givenMaxSize) { //가방의 생성자
		this._maxSize = givenMaxSize; //주어진 값으로 최대크기를 정함
		this._elements = new Coin[givenMaxSize]; //주어진 값만큼 코인이 들어갈 가방크기 생성
		this._size = 0;

	}
	public int size() {
		return this._size; //코인이 들어가는 사이즈 리턴
	}
	public boolean isEmpty() {
		return (this._size == 0); // 가방이 비었는지 안비었는지
	}
	public boolean isFull() {
		return (this._size == this._maxSize); //가방이 꽉 찼는지 안찼는지
	}
	public boolean doesContain(Coin anElement) { //가방에 특정 코인이 들어있는지 안들어있는지
		boolean found = false;
```

```
		for (int i = 0; i < this._size && ! found; i++) { //가방의 코인갯수크기만큼이나 특정 코인 찾을때까지
			if(this._elements[i].equals(anElement)) { // i 번째의 코인이 특정코인과 같으면
				found = true; //빙고
			}
		}
		return found;
	}
	public int frequencyOf(Coin anElement) { //특정 코인이 몇개나 들어있는지
		int frequencyCount = 0;
		for (int i = 0; i < this._size; i++) {
			if(this._elements[i].equals(anElement)) { //i 번째 코인이 특정 코인과 같으면
				frequencyCount++; //갯수 증가
			}

		}
		return frequencyCount;
	}
	public int maxElementValue() { //코인들 중의 최대값 반영
		int maxValue = 0;
		for (int i = 0; i < this.size() ; i++) { //가방안에 들어있는 코인의 갯수만큼 확인
			if(maxValue < _elements[i].value()) //최대값보다 i 번째 코인의 값이 크다면
			maxValue = _elements[i].value(); //i 번째 코인의 값을 최대값으로 변경
		}
		return maxValue;
	}
	public int sumElemnetValues() { //코인들의 합
		int sumValue = 0;
		for (int i = 0; i < this.size(); i++) { // 가방안에 들어있는 코인들의 갯수만큼
			sumValue += _elements[i].value(); //합에 코인의 값을 게속 더함.

		}
		return sumValue;
	}

	public boolean add(Coin anElement) { // 가방에 코인넣기
		if(this.isFull()) { //만약 꽉 찼으면
			return false; //넣지 말기
		} else { //꽉안찼으면
			this._elements[this._size] = anElement; //size 번째 공간에 코인을 넣기
			this._size++; //사이즈 증가
			return true;
		}
	}

	public boolean remove(Coin anElement) { //가방의 코인 삭제하기
		int foundIndex = 0;
		boolean found = false;

		for (int i = 0; i < this._size && !found; i++) { //가방에 들어있는 코인의 갯수나 해당 코인을 찾았을때까지
			if(this._elements[i].equals(anElement)) { //i 번째 코인이 특정코인과 같다면
				foundIndex = i; //몇번째 코인인지 저장하고
				found = true; //빙고
			}
		}
		//삭제된 원소 이후의 모든 원소를 앞쪽으로 한 칸씩 이동시킨다.
```

```
                if (!found) { //찾지않았으면
                        return false; //통과
                } else { //찾았으면
                        for (int i = foundIndex; i < this._size-1; i++) { //찾은 코인의 순번부터 코인갯수의 -
1 만큼 진행
                                this._elements[i] = this._elements[i+1]; //찾은 코인의 순번에 그 다음
순번의 코인을 집어넣음
                                /*한칸씩 땡김*/
                        }
                        this._elements[this._size-1] = null; //한칸씩땡기면 맨 뒷칸은 비어있게되므로
null 처리
                        this._size--; //코인이 들어있는 총 갯수 줄임
                        return true;
                }
        }
        public void clear() { //가방 비우기

                for(int i = 0; i < this._size; i++) {
                        this._elements[i] = null; //가방의 모든 내용물을 비우는 작업
                }
                this._size = 0; //들어있는 갯수를 0 개로
        }




}
```

Class	Coin

```
public class Coin {
        /*코인 클래스*/
        private int _value;

        public Coin() {
                //기본생성자
        }
        public Coin(int givenValue) {
                this._value  = givenValue; //코인값을 저장
        }

        public int value() {
                return this._value; //코인값 리턴
        }


        public void setValue(int newValue) {
                this._value = newValue; //코인값을 새로저장
        }

        public boolean equals(Coin aCoin) {
                return (this._value == aCoin.value());
        }
```

```
}
```