자료구조 실습 보고서

[제 02 주] 동전가방

제출일 : 2015 – 03 – 11 수요일

201402395 이승희

1. 프로그램 설명서

1) 주요 알고리즘 / 자료구조 / 기타

: 코인이 들어갈 가방의 최대 사이즈(1부터 100이내의 정수)를 정하면 코인의 액수를 차례로 입력하여 입력한 코인들의 전체 수와 총액, 가장 큰 액수의 코인을 출력하고, 코인이 가방에 들어있는지, 원하는 특정 코인의 개수는 몇 개인지도 확인할 수 있다.

2) 함수 설명서

(1) Coin의 Public Member function

Public Coin()

Public Coin(int aValue) // 전달받은 금액으로 새로운 Coin을 생성한다.

Public int value() // 금액을 리턴한다.

Public void setValue (int Value) // 전달받은 aValue를 현재 코인의 금액으로 설정한다.

Public Boolean equals (Coin aCoin) // 입력받은 aCoin의 금액과 현재의 Value값이 같은지 확인하는 함수

(2) ArrayBag의 Public Member Function

Public ArrayBag()

Public ArrayBag (int givenMaxSize)

Public int size()

Public Boolean isEmpty() // 배열이 비어있는지 확인한다.

Public boolan isFull() // 배열이 가득 차 있는지 확인한다.

Public Boolean doesContain(Coin anElement) // 배열 안에 주어진 값이 존재 하는지 확인한다.

Public int frequencyOf(Coin anElement) // 배열 안에 주어진 값이 몇 개 있는지 확인한다.

Public int maxElementValue() // 배열 안에 가장 큰 원소를 돌려준다.

Public int sumElementValues() // 전체 원소의 합계를 돌려준다.

Public Boolean add(Coin anElement) // 배열에 값을 추가한다.

Public Boolean remove(Coin anElement) // 배열에서 값을 삭제한다.

Public void clear() // 배열을 초기화한다.

3) 종합 설명서

:『동전 가방 – 배열 가방』프로그램을 MVC 패턴을 적용하여 사용자로부터 입력받은 값에 따라 프로그램을 진행, 종료, 검토할 수 있다.

2. 프로그램 장단점 분석

: MVC패턴을 적용하여 프로그램을 한눈에 볼 수 있고 수정하기 쉽다는 장점이 있으나, 총 코인의 개수를 입력하였을 때 총 개수보다 입력하는 수의 개수가 더 클 때 저장은 되지 않지만 프로그램이 종료되는 등, 이 경우에 따른 상황이 주어지지 않아 아쉬웠다.

3. 실행 결과 분석

1) 입력과 출력 (화면 capture 시킬 것)

(1) 코인을 추가, 제거, 출력, 검색, 종료할 경우

```
〈〈동전 가방 프로그램을 시작합니다〉〉
가방에 들어갈 총 코인 개수를 입력하시오 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 35
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 20
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 2
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 3
총 코인 : 3
가장 큰 코인 : 35
코인의 합 : 65
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 4
코인의 액수를 입력하세요 : 10

10코인은 2개 존재합니다.
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 9
9가 입력되어 종료합니다
총 코인 : 4
가장 큰 코인 : 35
코인의 합 : 75
〈〈동전 가방 프로그램을 종료합니다〉〉
```

(2) 같은 코인이 여러 개 있을 때에 remove 할 경우

```
〈〈동전 가방 프로그램을 시작합니다〉〉
가방에 들어갈 총 코인 개수를 입력하시오 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 3
총 코인 : 3
가장 큰 코인 : 10
코인의 합 : 25
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 2
코인의 액수를 입력하세요 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) : 3
총 코인 : 2
가장 큰 코인 : 10
코인의 합 : 15
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4, exit : 9) :
```

(3) 코인의 최대 개수를 넘었을 경우

```
<<동전 가방 프로그램을 시작합니다>>
가방에 들어갈 총 코인 개수를 입력하시오 : 3
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) : 3
총 코인 : 3
가장 큰 코인 : 5
코인의 합 : 15
수행하려고하는 메뉴를 선택하세요
(add : 1, remove : 2, print : 3, search : 4, exit : 9) :
```

2) 결과 분석

: 프로그램을 원하는 바와 같이 짜긴 했지만, 입력한 코인의 개수가 지정한 총 개수보다 많을 때, 코인이 저장되지는 않지만 그에 걸맞는 Error Message가 출력되는 등의 대안이 없어서 아쉬웠다.

4. 소스코드

1) DS1_02_201402395_이승희

```java
public class DS1_02_201402395_이승희 {

  public static void main(String[] args) {
    AppController appController = new AppController();
    appController.run();
  }

}
```

2) AppController

```java
public class AppController {
	private AppView _appView;
	private ArrayBag _coinCollector;
```

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

public void run() {
        int totalCoin = 0;
        int input = 0;
        int order = 0;

        this.showMessage(MessageID.Notice_StartProgram);
        this.showMessage(MessageID.Notice_InputTotalCoin);
        totalCoin = this._appView.inputInt();
        this._coinCollector = new ArrayBag(totalCoin);

        while (order != 9) {
                this.showMessage(MessageID.Notice_Menu);
                order = this._appView.inputInt();
                if (order == 1) {
                        this.showMessage(MessageID.Notice_InputCoin);
                        input = _appView.inputInt();
                        Coin anCoin = new Coin(input);
                        this._coinCollector.add(anCoin);
                } else if (order == 2) {
                        this.showMessage(MessageID.Notice_InputCoin);
                        input = this._appView.inputInt();
                        Coin givenCoin = new Coin(input);
                        this._coinCollector.remove(givenCoin);
                } else if (order == 3) {
                        this._appView.outputResult(this._coinCollector.size(),
                                        this._coinCollector.maxElementValue(),
                                        this._coinCollector.sumElementValue());
                } else if (order == 4) {
                        this.showMessage(MessageID.Notice_InputCoin);
                        input = this._appView.inputInt();
                        Coin givenCoin = new Coin(input);
                        this._appView.outputSearch(input,
                                        this._coinCollector.frequencyOf(givenCoin));
                } else if (order == 9) {
                        this.showMessage(MessageID.Notice_EndMenu);
                        this._appView.outputResult(this._coinCollector.size(),
                                        this._coinCollector.maxElementValue(),
                                        this._coinCollector.sumElementValue());
                        this.showMessage(MessageID.Notice_EndProgram);
                } else
                        this.showMessage(MessageID.Error_WrongMenu);
```

```
                }
        }

        private void showMessage(MessageID aMessageID) {
                switch (aMessageID) {
                case Notice_StartProgram:
                        this._appView.outputMessage("〈〈동전 가방 프로그램을 시작합니다〉〉₩n");
                        break;
                case Notice_EndProgram:
                        this._appView.outputMessage("〈〈동전 가방 프로그램을 종료합니다〉〉₩n");
                        break;
                case Notice_InputTotalCoin:
                        this._appView.outputMessage("가방에 들어갈 총 코인 개수를 입력하시오 :
");
                        break;
                case Notice_Menu:
                        this._appView
                                        .outputMessage("수행하려고하는 메뉴를 선택하세요₩n"
                                                        + "〈add : 1, remove : 2, print : 3,
search : 4, exit : 9) : ");
                        break;
                case Notice_EndMenu:
                        this._appView.outputMessage("9가 입력되어 종료합니다₩n");
                        break;
                case Notice_InputCoin:
                        this._appView.outputMessage("코인의 액수를 입력하세요 : ");
                        break;
                case Error_WrongMenu:
                        this._appView.outputMessage("〈〈ERROR : 잘못된 메뉴입니다.〉〉₩n");
                        break;
                default:
                        break;
                }
        }

}
```

3) AppView

```
import java.util.*;

public class AppView {
        private Scanner _scanner;

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

```
	public int inputInt() {
		int _order = this._scanner.nextInt();
		return _order;
	}

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

	public void outputSearch(int aSearchValue, int aSearchedSize) {

		System.out.println(aSearchValue + "코인은 " + aSearchedSize + "개 존재합니다.");
	}

	public void outputMessage(String aMessageString) {
		System.out.print(aMessageString);
	}

}
```

4) MessageID

```
public enum MessageID {
	//Message IDs for Notices;
	Notice_StartProgram,
	Notice_EndProgram,
	Notice_InputTotalCoin,
	Notice_Menu,
	Notice_EndMenu,
	Notice_InputCoin,

	//message IDs for Errors:
	Error_WrongMenu;

}
```

5) Coin

```
public class Coin {
	private int _value;

	public Coin() {
		this._value = 0;
	}
```

```
	public Coin(int aValue) {
		this._value = aValue;
	}

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

	public void setValue(int aValue) {
		this._value = aValue;
	}

	public boolean equals(Coin aCoin) {
		if (this._value == aCoin._value)
			return true;
		else
			return false;
	}

}
```

6) ArrayBag

```
public class ArrayBag {
	private static final int DEFAULT_MAX_SIZE = 100;
	private int _maxSize;
	private int _size;
	private Coin _elements[];

	public void ArraryBag() {
		this._maxSize = DEFAULT_MAX_SIZE;
		this._elements = new Coin[this._maxSize];
		this._size = 0;
	}

	public ArrayBag(int givenMaxSize) {
		this._maxSize = givenMaxSize;
		this._elements = new Coin[this._maxSize];
		this._size = 0;
	}

	public int size() {

		return _size;
	}
```

```
public boolean isEmpty() {
        if (this._size == 0)
                return true;
        else
                return false;
}

public boolean isFull() {
        if (this._size == this._maxSize)
                return true;
        else
                return false;
}

public boolean doesContain(Coin anElement) {
        boolean found = false;
        for (int i = 0; i < this._size && found == true; i++) {
                if (this._elements[i].equals(anElement))
                        found = true;
        }
        return found;
}

public int frequencyOf(Coin anElement) {
        int frequencyCount = 0;
        for (int i = 0; i < this._size; i++) {
                if (this._elements[i].equals(anElement))
                        frequencyCount++;
        }
        return frequencyCount;

}

public int maxElementValue() {
        int maxValue = 0;
        for (int i = 0; i < this._size; i++) {
                if (maxValue < this._elements[i].value())
                        maxValue = this._elements[i].value();
        }
        return maxValue;
}

public int sumElementValue() {
        int sumValue = 0;
        for (int i = 0; i < this._size; i++) {
```

```
                    sumValue += this._elements[i].value();
                }
                return sumValue;

        }

        public boolean add(Coin anElement) {
                if (this.isFull() == true)
                        return false;
                else {
                        this._elements[this._size] = anElement;
                        this._size++;
                        return true;
                }

        }

        public boolean remove(Coin anElement) {
                if (this.isEmpty() == true)
                        return false;
                else {
                        for (int i = 0; i < this._size; i++) {
                                if (this._elements[i].equals(anElement)) {
                                        this._elements[i] = null;
                                        for (int j = i; j < this._size - 1; j++) {
                                                this._elements[j] = this._elements[j + 1];
                                        }
                                        this._size--;

                                }
                        }
                        return true;
                }
        }

        public void clear() {
                this._size = 0;
        }

}
```