자료구조 실습 보고서

[제 3 주] 연결체인 가방

제출일 : 2015 – 03 – 18

201402395 이승희

1. 프로그램 설명서

1) 주요 알고리즘 / 자료구조 / 기타

: 이전의 실습을 토대로 Linked Chain을 추가하여 동전 가방을 만들었다. 연결체인은 하나하나의 노드로 연결, 각 노드는 두 개의 값을 가지고 있다. 첫 번째 값은 실제 우리가 저장하려는 정보이고, 두 번째 항목은 자신과 동일한 모습을 갖는 다른 노드를 가진다.

2) 함수 설명서

(1) Node의 Public Member functions

Public Node()

Public Node(Coin anElement) // Coin만 전달하여 Node를 생성

Public Node(Coin anElement, Node aNode) //Coin과 다음 Node를 전달하여 Node를 생성

Public Coin element() // Node에 있는 Coin을 전달한다.

Public Node next() // Node의 다음 Node를 전달한다.

Public void setElement(Coin anElement) // Node안에 있는 element를 전달 받은 anElement으로 변경한다.

Public void setNext(Node aNode) // 다음 Node를 aNode로 변경한다.

(2) LinkedBag의 Member Functions

Public LinkedBag() // LinkedBag의 생성자

Public int size() // Bag에 들어있는 개수를 확인한다

Public boolean isEmpty() // Bag이 비어있는지 확인한다.

Public boolean doesContain(Coin anElement) // Bag안에 주어진 값이 존재 하는지 확인한다.

Public int frequencyOf(Coin anElement) // Bag안에 주어진 값이 몇 개 있는지

확인한다

Public int maxElementValue() // Bag 안에 가장 큰 값을 확인한다.

Public int sumElementValues() // Bag 안에 들어있는 총 금액을 확인한다

Public boolean add(Coin anElement) // Bag에 값을 추가한다.

Public Coin remove(Coin anElement) // Bag에서 값을 삭제한다.

Public Coin removeAny() // Bag에서 무작위로 삭제하고, 여기서는 가장 앞의 값을 삭제한다.

Public void clear() // Bag을 초기화한다.

3) 종합 설명서

: 기존 프로그램에 removeAny() 메소드를 추가하여 아무 원소나 제거하는 기능을 추가한 동전 가방 프로그램이다. 지난 주와 다르게 연결체인을 활용한 프로그램이다.

2. 프로그램 장단점 분석

: MVC로 나누어져 있어 보기 편리하고 접근이 용이하다. arrayBag과 달리, 노드를 이용함으로써 공간의 제약이 적고 공간의 할당이 필요 없다.

3. 실행 결과 분석

1) 입력과 출력

(1) 코인을 추가하고 출력한 후 종료시킬 때

```
〈〈동전 가방 프로그램을 시작합니다〉〉
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 15
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 40
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 20
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 30
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 3
총 코인 : 5
가장 큰 코인 : 40
코인의 합 : 110
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 9
9가 입력되어 종료합니다
총 코인 : 5
가장 큰 코인 : 40
코인의 합 : 110
〈〈동전 가방 프로그램을 종료합니다〉〉
```

(2) 추가한 코인에서 특정 코인을 삭제하거나 검색할 때

```
〈〈동전 가방 프로그램을 시작합니다〉〉
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 15
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 3
총 코인 : 2
가장 큰 코인 : 15
코인의 합 : 20
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 2
코인의 액수를 입력하세요 : 5
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 3
총 코인 : 1
가장 큰 코인 : 15
코인의 합 : 15
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 4
코인의 액수를 입력하세요 : 15
15코인은 1개 존재합니다.
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 9
9가 입력되어 종료합니다
총 코인 : 1
가장 큰 코인 : 15
코인의 합 : 15
〈〈동전 가방 프로그램을 종료합니다〉〉
```

(3) 임의의 원소를 제거할 때

```
〈〈동전 가방 프로그램을 시작합니다〉〉
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 15
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 30
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 1
코인의 액수를 입력하세요 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 5
삭제된 코인 : 10
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) : 3
총 코인 : 2
가장 큰 코인 : 30
코인의 합 : 45
수행하려고하는 메뉴를 선택하세요
〈add : 1, remove : 2, print : 3, search : 4,removeAny : 5, exit : 9) :
```

2) 결과 분석

: 대부분의 기능들은 이전의 프로그램과 같지만 removeAny의 기능을 추가한 결과, 이 기능을 수행했을 때 node의 head부분부터 차례로 제거됨을 확인할 수 있었다.

4. 소스코드

1) Main

```
public class DS1_02_201402395_이승희 {

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

}
```

2) AppController

```
public class AppController {
  private AppView _appView;
  private LinkedBag _coinCollector;

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

  public void run() {

    int input = 0;
    int order = 0;

    this.showMessage(MessageID.Notice_StartProgram);

    this._coinCollector = new LinkedBag();

    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.sumElementValues());
    } 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 == 5) {
      Coin removeCoin = new Coin();
      removeCoin = this._coinCollector.removeAny();
      this._appView.outputRemove(removeCoin.value());
    } else if (order == 9) {
      this.showMessage(MessageID.Notice_EndMenu);
      this._appView.outputResult(this._coinCollector.size(),
          this._coinCollector.maxElementValue(),
          this._coinCollector.sumElementValues());
      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,removeAny : 5, 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 outputRemove(int removedCoin) {
    System.out.println("삭제된 코인 : "+removedCoin);
  }

  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 outputSearch(int aSearchValue, int aSearchedSize) {

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

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

}
```

4) LinkedBag

```
public class LinkedBag {
  private int _size;
  private Node _head; // LinkedBag의 원소들을 담을 Linked Chain

  public LinkedBag() {
    this._size = 0;
    this._head = null;
  }

  public int size() {
    return this._size;
  }

  public boolean isEmpty() {
    return (this._size == 0);
  }

  public boolean isFull() {
    return false;
  }
```

```
public boolean doesContain(Coin anElement) {
  boolean found = false;

  Node searchNode = this._head;
  while (searchNode != null && !found) {
    if (searchNode.element().equals(anElement))
      found = true;
    else
      searchNode = searchNode.next();
  }
  return found;
}

public int frequencyOf(Coin anElement) {
  int frequencyCount = 0;
  Node currentNode = this._head;
  while (currentNode != null) {
    if (currentNode.element().equals(anElement))
      frequencyCount++;
    currentNode = currentNode.next();
  }
  return frequencyCount;
}

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


public int maxElementValue() {
  int maxValue = 0;
  Node searchNode = this._head;
  for (int i = 0; i < this._size; i++) {
    if (maxValue < searchNode.element().value())
      maxValue = searchNode.element().value();
    searchNode = searchNode.next();
  }
  return maxValue;
}

public int sumElementValues() {
  int sumValues = 0;
  Node searchNode = this._head;
  for (int i = 0; i < this._size; i++) {
    sumValues += searchNode.element().value();
    searchNode = searchNode.next();
  }
  return sumValues;

}

public Coin any() {
  if (this.isEmpty())
    return null;
  else
    return this._head.element();
}


public boolean add(Coin anElement) {
  if (this.isFull())
    return false;
  else {
    Node newNode = new Node();
    newNode.setElement(anElement);
    newNode.setNext(this._head);
    this._head = newNode;
    this._size++;
    return true;
  }

}

public boolean remove(Coin anElement) {
  if (this.isEmpty())
    return false;
  else {
    Node previousNode = null;
    Node currentNode = _head;
    boolean found = false;

    while (currentNode != null && !found) {
      if (currentNode.element().equals(anElement)) {
        found = true;
      } else {
        previousNode = currentNode;
        currentNode = currentNode.next();
      }
    }


    if (!found)
      return false;
    else {
      if (currentNode == this._head)
        this._head = this._head.next();
      else
        previousNode.setNext(currentNode.next());
      this._size--;
      return true;
    }
  }
}
```

```
  public Coin removeAny() {
    if (this.isEmpty())
      return null;
    else {
      Coin removedElement = this._head.element();
      this._head = this._head.next();
      this._size--;
      return removedElement;
    }
  }
}
```

5) Node

```
public class Node {
  private Coin _element;
  private Node _next;

  public Node() {
    this._element = null;
    this._next = null;
  }

  public Node(Coin anElement) {
    this._element = anElement;
    this._next = null;
  }

  public Node(Coin anElement, Node aNode) {
    this._element = anElement;
    this._next = aNode;
  }

  public Coin element() {
    return this._element;

  }

  public Node next() {
    return this._next;
  }

  public void setElement(Coin anElement) {
    this._element = anElement;
  }


  public void setNext(Node aNode) {
    this._next = aNode;
  }

}
```

6) 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;
  }

}
```

7) 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;

}
```