자료구조 실습 보고서

[제 05주] 별의 집합

제출일 : 2015 – 04 – 01

201402395 이승희

1. 프로그램 설명서

1) 주요 알고리즘 / 자료구조 / 기타

: 별의 정보를 입력 받을 때 별의 좌표는 int형, 이름은 String형으로 입력받으며 별의 이름으로 기존의 별을 삭제할 수 있다. 또한 removeAny() 메소드로 임의의 별을 삭제할 수 있다. 별의 정보가 이름과 좌표로 나뉘어지므로 이에 따라 이름, 좌표별로 검색이 가능하다. 9가 입력되면 입력은 종료되는 것으로 본다.

2) 함수 설명서

(1) Star의 Member Functions

Public Star(int aX, int aY) // X값과 Y값을 전달받아 새로운 Star를 생성하며 검색할 때 사용.

Public Star(String aStarName) // 별의 이름을 받아 새로운 Star를 생성하며 삭제 및 검색을 할 때 사용

Public int xCoordinate() // x좌표를 전달

Public int yCoordinate() // y좌표를 전달

Public String starName() // 별의 이름을 전달

Public void setXCoordinate(int aX) // X좌표를 새로운 값 aX로 수정

Public void setYCoordinate(int aY) // Y좌표를 새로운 값 aY로 수정

Public void setStarName(String aStarName) // 별의 이름을 새로운 이름 aStarName으로 변경

Public Boolean equals(Star aStar) // 전달받은 aStar값과 같은 것이 있는지 확인하며 True일 경우 같은 값이 존재하는 경우

(2) ArraySet의 Member Functions

Public int size() // 별의 개수를 확인

Public Boolean isEmpty() // 별의 집합이 비어있는지 확인

Public Boolean isFull() // 별의 집합이 가득 차 있는지 확인

Public Boolean doesContain(Star anElement) // 별의 집합 안에 anElement 값이 있는지 확인

Public Boolean add(Star anElement) // 별의 집합에 별의 데이터를 추가

Public Star remove(Star anElement) // 별의 집합에서 별의 데이터를 삭제

Public Star removeAny() // 임의의 데이터를 삭제하며 여기서는 가장 뒤의 값을 삭제

Public void clear() // 별의 집합을 초기화

(3) LinkedSet의 Member Functions

public Star removeAny() // 임의의 데이터를 삭제하며 여기서는 가장 앞의 값을 삭제

//나머지 함수는 기존 ArraySet 클래스의 함수와 동일

3) 종합 설명서

: 별자리의 이름과 좌표를 중복 없이 입력하며, 입력된 별들의 개수를 셀 수 있고 원하는 별과 임의의 별을 삭제할 수 있다. 또한 별의 이름, 좌표별로 검색 가능하다.

2. 프로그램 장단점 분석

: 하나의 프로그램에서 ArraySet과 Linked을 조금만 수정하면 같이 사용할 수 있다는 점이 장점인 것 같다. 단점이 있다면 출력할 부분이 많다는 것 뿐이다.

3. 실행 결과 분석

1) 입력과 출력

〈 별의 집합을 시작합니다 〉

1:입력 2:주어진 별 삭제 3:임의의 별 삭제
4:출력 5:이름으로 검색 6:좌표로 검색 9:종료
원하는 메뉴를 입력하세요 : 1
- [입력] -
- x좌표를 입력하시오 : 1
- y좌표를 입력하시오 : 1
- 별의 이름을 입력하시오 : a
1:입력 2:주어진 별 삭제 3:임의의 별 삭제
4:출력 5:이름으로 검색 6:좌표로 검색 9:종료
원하는 메뉴를 입력하세요 : 1
- [입력] -
- x좌표를 입력하시오 : 2
- y좌표를 입력하시오 : 3
- 별의 이름을 입력하시오 : b
1:입력 2:주어진 별 삭제 3:임의의 별 삭제
4:출력 5:이름으로 검색 6:좌표로 검색 9:종료
원하는 메뉴를 입력하세요 : 1
- [입력] -
- x좌표를 입력하시오 : 4
- y좌표를 입력하시오 : 5
- 별의 이름을 입력하시오 : c
1:입력 2:주어진 별 삭제 3:임의의 별 삭제
4:출력 5:이름으로 검색 6:좌표로 검색 9:종료
원하는 메뉴를 입력하세요 : 1
- [입력] -
- x좌표를 입력하시오 : 1
- y좌표를 입력하시오 : 2
- 별의 이름을 입력하시오 : a
ERROR : 잘못된 입력입니다.

```
 1:입력 2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 2
- [주어진 별 삭제] -
- 별의 이름을 입력하시오 : b
X좌표 : 2
Y좌표 : 3
별의 이름 : b
1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 5
- [이름으로 검색] -
- 별의 이름을 입력하시오 : b
원하는 별이 존재하지 않습니다.
1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 1
- [입력] -
- x좌표를 입력하시오 : 3
- y좌표를 입력하시오 : 4
- 별의 이름을 입력하시오 : b
1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 5
- [이름으로 검색] -
- 별의 이름을 입력하시오 : b
b 별이 존재합니다.

1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 6
- [좌표로 검색] -
- x좌표를 입력하시오 : 4
- y좌표를 입력하시오 : 5
( 4, 5 ) 위치에 별이 존재합니다.
1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 3
- [임의의 별 삭제] -
X 좌표 : 3
Y 좌표 : 4
별의 이름 : b
1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력    5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 3
- [임의의 별 삭제] -
X 좌표 : 4
Y 좌표 : 5
별의 이름 : c
```

```
1:입력  2:주어진 별 삭제      3:임의의 별 삭제
4:출력     5:이름으로 검색   6:좌표로 검색   9:종료
원하는 메뉴를 입력하세요 : 9
9가 입력되어 종료합니다.
1개의 별이 존재합니다.
〈 별의 집합을 종료합니다 〉
```

2) 결과 분석

: 원하는 바와 같이 출력되었고, LinkedSet으로 실행해서 임의의 별을 삭제할 때 head 부터 삭제되었다.

4. 소스코드

1) DS1_05_201402395_이승희

```
public class DS1_05_201402395_이승희 {

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

  }

}
```

2) AppController

```
public class AppController {
  private AppView _appView;
  // private ArraySet _starCollector;
  private LinkedSet _starCollector;

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

  private void input() {
    this.showMessage(MessageID.Notice_InputStar);
    this.showMessage(MessageID.Notice_InputStarXCoordinate);
    int xCoordinate = this._appView.inputInt();
    this.showMessage(MessageID.Notice_InputStarYCoordinate);
    int yCoordinate = this._appView.inputInt();
    this.showMessage(MessageID.Notice_InputStarName);
    String starName = this._appView.inputString();
    if (!this._starCollector.add(new Star(xCoordinate, yCoordinate,
        starName)))
      this.showMessage(MessageID.Error_Input);
  }

  private void remove() {
    this.showMessage(MessageID.Notice_RemoveStar);
    this.showMessage(MessageID.Notice_InputStarName);

    String starName = this._appView.inputString();

    Star removeStar = this._starCollector.remove(new Star(starName));
```

```
    if (removeStar == null)
      this.showMessage(MessageID.Error_Remove);
    else {
      System.out.println("X좌표 : " + removeStar.xCoordinate());
      System.out.println("Y좌표 : " + removeStar.yCoordinate());
      System.out.println("별의 이름 : " + removeStar.starName());
    }
  }

  private void searchByName() {
    this.showMessage(MessageID.Notice_SearchByName);
    this.showMessage(MessageID.Notice_InputStarName);
    String starName = this._appView.inputString();
    Star aStar = new Star(starName);

    if (this._starCollector.doesContain(aStar)) {
      this._appView.outputStarExistence(starName, 0, 0);
    } else {
      System.out.println("원하는 별이 존재하지 않습니다.");
    }
  }

  private void searchByCoordinate() {
    this.showMessage(MessageID.Notice_SearchByCoordinate);
    this.showMessage(MessageID.Notice_InputStarXCoordinate);
    int xCoordinate = this._appView.inputInt();
    this.showMessage(MessageID.Notice_InputStarYCoordinate);
    int yCoordinate = this._appView.inputInt();

    Star aStar = new Star(xCoordinate, yCoordinate);

    if (this._starCollector.doesContain(aStar))
      this._appView.outputStarExistence(null, xCoordinate, yCoordinate);
    else {
      System.out.println("원하는 별이 존재하지 않습니다.");
    }
  }


  public void run() {
    // this._starCollector = new ArraySet();
    this._starCollector = new LinkedSet();
    this.showMessage(MessageID.Notice_StartProgram);
    int command = 0;
    while (command != 9) {
      try {
        this.showMessage(MessageID.Notice_Menu);
        command = this._appView.inputInt();
        if (command == 1) {
          this.input();
        } else if (command == 2) {
          this.remove();
        } else if (command == 3) {
          this.showMessage(MessageID.Notice_RemoveRandomStar);
          Star removedStar = this._starCollector.removeAny();
          if (removedStar != null)
            this._appView.outputStar(removedStar.starName(),
                removedStar.xCoordinate(),
                removedStar.yCoordinate());
          else
            this.showMessage(MessageID.Error_Remove);
        } else if (command == 4) {
          this.showMessage(MessageID.Notice_Show);
          this._appView.outputNumOfStars(this._starCollector.size());

        } else if (command == 5) {
          this.searchByName();
        } else if (command == 6) {
          this.searchByCoordinate();
```

```
      } else if (command == 9) {
        this.showMessage(MessageID.Notice_EndMenu);
        this._appView.outputNumOfStars(this._starCollector.size());
        break;
      } else {
        this.showMessage(MessageID.Error_WrongMenu);
      }

    } catch (Exception ex) {
      System.out.println("ErrorMessage : " + ex.getMessage());
      continue;
    }
  }
  this.showMessage(MessageID.Notice_EndProgram);
}

private void showMessage(MessageID aMessageID) {
  switch (aMessageID) {
  case Notice_StartProgram:
    this._appView.outputMessage("〈 별의 집합을 시작합니다 〉₩n₩n");
    break;
  case Notice_EndProgram:
    this._appView.outputMessage("〈 별의 집합을 종료합니다 〉₩n");
    break;
  case Notice_InputStar:
    this._appView.outputMessage("- [입력] -₩n");
    break;
  case Notice_InputStarXCoordinate:
    this._appView.outputMessage("- x좌표를 입력하시오 : ");
    break;

case Notice_InputStarYCoordinate:
  this._appView.outputMessage("- y좌표를 입력하시오 : ");
  break;
case Notice_InputStarName:
  this._appView.outputMessage("- 별의 이름을 입력하시오 : ");
  break;

case Notice_RemoveRandomStar:
  this._appView.outputMessage("- [임의의 별 삭제] -₩n");
  break;
case Notice_RemoveStar:
  this._appView.outputMessage("- [주어진 별 삭제] -₩n");
  break;
case Notice_Show:
  this._appView.outputMessage("- [출력] -₩n");
  break;
case Notice_SearchByName:
  this._appView.outputMessage("- [이름으로 검색] -₩n");
  break;
case Notice_SearchByCoordinate:
  this._appView.outputMessage("- [좌표로 검색] -₩n");
  break;

case Notice_Menu:
  this._appView.outputMessage("1:입력  2:주어진 별 삭제   3:임의의 별 삭제₩n"
      + "4:출력    5:이름으로 검색   6:좌표로 검색   9:종료₩n"
      + "원하는 메뉴를 입력하세요 : ");
  break;
case Notice_EndMenu:
  this._appView.outputMessage("9가 입력되어 종료합니다.₩n");
  break;
```

```
    case Error_Input:
      this._appView.outputMessage("ERROR : 잘못된 입력입니다.₩n ");
      break;
    case Error_WrongMenu:
      this._appView.outputMessage("ERROR : 잘못된 입력입니다.₩n");
      break;
    case Error_Remove:
      this._appView.outputMessage("ERROR : 잘못된 제거입니다.₩n");
    default:
      break;
    }
  }
}
```

3) AppView

```
import java.util.*;

public class AppView {
  private Scanner _scanner;

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

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

  public String inputString() {
    return this._scanner.next();
  }

  public void outputStar(String aStarName, int aX, int aY) {
    System.out.println("X 좌표 : " + aX);
    System.out.println("Y 좌표 : " + aY);
    System.out.println("별의 이름 : " + aStarName);
  }

  public void outputStarExistence(String aStarName, int aX, int aY) {
    if (aX == 0 && aY == 0)
      System.out.println(aStarName + " 별이 존재합니다.");

    else if (aStarName == null)
      System.out.println("( " + aX + ", " + aY + " ) 위치에 별이 존재합니다.");

  }

  public void outputNumOfStars(int aStarCollectorSize) {
    System.out.println(aStarCollectorSize + "개의 별이 존재합니다.");
  }

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

  }
}
```

4) ArraySet

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

  public ArraySet() {
    this._maxSize = DEFAULT_MAX_SIZE;
    this._elements = new Star[this._maxSize];
    this._size = 0;
  }

  public ArraySet(int aMaxSize) {
    this._maxSize = aMaxSize;
    this._elements = new Star[this._maxSize];
    this._size = 0;
  }

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

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

```
public Star remove(Star aStar) {
  Star removeStar = null;
  if (this.isEmpty() == true)
    return null;
  else {
    for (int i = 0; i < this._size; i++) {
      if (this._elements[i].equals(aStar)) {
        removeStar = this._elements[i];
        for (int j = i; j < this._size - 1; j++) {
          this._elements[j] = this._elements[j + 1];
        }
      }
    }
    this._elements[this._size - 1] = null;
    this._size--;
    return removeStar;
  }
}

public Star removeAny() {
  Star removeAnyStar = null;
  if (this.isEmpty())
    return null;
  else {
    removeAnyStar = this._elements[this._size - 1];
    this._elements[this._size - 1] = null;
    this._size--;
    return removeAnyStar;
  }
}


  public void clear() {
    for (int i = 0; i < this._size; i++) {
      this._elements[i] = null;
    }
    this._size = 0;
  }
}
```

5) LinkedSet

```
public class LinkedSet {
  private static final int DEFAULT_MAX_SIZE = 100;
  private int _maxSize;
  private int _size;
  private Node _head;

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

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

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

  public boolean isFull() {
    return (this._size == this._maxSize);
  }
```

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

}

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

    while (currentNode != null && !found) {
      if (currentNode.element().equals(anElement)) {
        found = true;
      } else {
        previousNode = currentNode;
        currentNode = currentNode.next();
      }
    }
    if (!found) {
      return null;
    } else {
      if (currentNode == this._head) {
        this._head = this._head.next();
      } else {
        previousNode.setNext(currentNode.next());
      }
      this._size--;
      return currentNode.element();
    }
  }
}

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

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

6) Node

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

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

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

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

  public Star element() {
    return this._element;

  }

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

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

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

}
```

7) Star

```
public class Star {
  private int _xCoordinate;
  private int _yCoordinate;
  private String _starName;

  public Star() {
  }

  public Star(int aX, int aY) {
    this._xCoordinate=aX;
    this._yCoordinate=aY;
  }

  public Star(String aStarName) {
    this._starName=aStarName;
  }

  public Star(int aX, int aY, String aStarName) {
    this._xCoordinate=aX;
    this._yCoordinate=aY;
    this._starName=aStarName;
  }

  public int xCoordinate() {
    return this._xCoordinate;
  }

  public int yCoordinate() {
    return this._yCoordinate;
  }
```

```
  public String starName() {
    return this._starName;
  }

  public void setXcoordinate(int aX) {
    this._xCoordinate = aX;
  }

  public void setYCoordinate(int aY) {
    this._yCoordinate = aY;
  }

  public void setStarName(String aStarName) {
    this._starName = aStarName;
  }

  public boolean equals(Star aStar) {
    if (this._xCoordinate == aStar.xCoordinate()
        && this._yCoordinate == aStar.yCoordinate())
      return true;
    else if (aStar._starName != null
        && this._starName.equals(aStar.starName()))
      return true;
    else
      return false;
  }
}
```

8) MessageID

```
public enum MessageID {
  // Message IDs for Notices:
  Notice_StartProgram,
  Notice_EndProgram,
  Notice_Menu,
  Notice_EndMenu,
  Notice_InputStar,
  Notice_InputStarName,
  Notice_InputStarXCoordinate,
  Notice_InputStarYCoordinate,
  Notice_RemoveStar,
  Notice_RemoveRandomStar,
  Notice_Show,
  Notice_SearchByName,
  Notice_SearchByCoordinate,
  // message IDs for Errors:
  Error_WrongMenu, Error_Input, Error_Remove,

}
```

5. 정리

1) Set과 Bag의 구현상 차이점은 어떤 부분이 있는가?

: Bag의 경우 코인의 정보는 액수인 정수형밖에 없기 때문에 다루기 쉽지만, Set의 별자리는 이름은 String형으로, 좌표는 정수형으로 받기 때문에 더 복잡하다. Remove의 경우 Set에서는 String형으로 입력받은 별의 정보에 따라 이 별의 좌표값까지 가져와서 출력해야 했기 때문에 Bag과 차이점이 있었다. 또한 코인의 액수로 검색하는 Bag과 달리 Set은 이름별, 좌표별로 검색 가능하다. 그렇기 때문에 사용자로부터 입력받은 값이 String형인지 int형인지를 구분해서 구현해야했다.

2) 실습 자료의 초록색으로 되어 있는 부분의 구현 방법과 그렇게 프로그램을 작성한 이

유?

: 클래스 ArraySet과 LinkedSet을 구성하고 있는 함수들의 기능과 이름들이 각각 동일하기 때문이다.