자료구조: 2022년 1학기 [실습]

제 4 주:

동전 가방

강 지 훈
jhkang@cnu.ac.kr
충남대학교 컴퓨터융합학부

실습 목표

실습 목표

- 이론적 관점
  - Generic Type의 필요성
  - Bag 의 개념
- 구현적 관점
  - Class “LinkedBag”: Bag 을 연결체인으로 구현하는 방법

Generic Type

Why Generic Type ? [1]

- 창고 짓는 설계도를 그렸다:
  - 이 설계도가 컴퓨터 모니터 전용 창고 용도라면,
    이 설계도로 지은 창고는 컴퓨터 모니터만 보관 가능.
- 그런데, 모니터나 키보드나 보관 방법이 다르지 않다면?
  - 모니터 보관용 창고 설계도로, 키보드 보관용 창고도 지을 수 있을 것이다.
- 하나의 창고 설계도로 다양한 type 의 객체를 보관할 수 있는 창고를 지을 수 있다.

Why Generic Type ? [2]

- 예를 들어, class "ArrayList" 는 담아야 할 원소의 종류가 무엇이든지 그 기능은 동일한, 그러한 객체들의 설계도 이다.

- 그렇다면, 실제 생성된 ArrayList (창고)에 어떤 자료형의 원소를 담을 지는, 설계할 때 미리 결정해 놓을 것이 아니라, "ArrayList" 를 사용하는 시점에 결정할 수 있다면 편리할 것이다.

Why Generic Type ? [3]

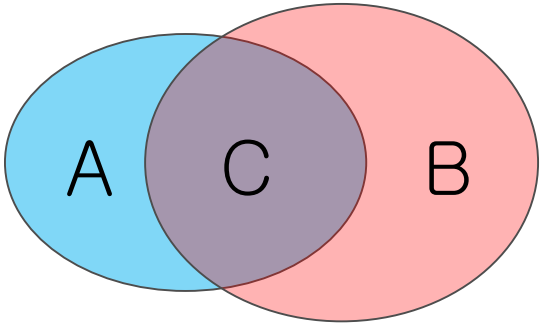
- 만일, "ArrayList" 를 정의하는 시점에 담을 원소의 자료형을 고정시켜야만 된다면 ?
  - 그 "ArrayList" 는 고정된 한 type 의 객체 만을 담을 수 있는 "ArrayList" 가 될 것이다.
  - 또한 우리는 원소의 type 마다 별도의 "ArrayList" 를 만들어 사용해야 할 것이다.
    - ArrayListForMoniter, ArrayListForKeyboard, ...

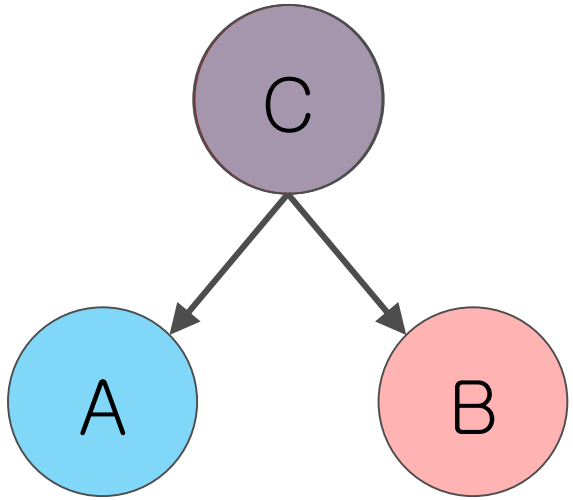
- 담아야 할 객체들의 다양한 type 마다,
  - 별도의 설계도를 작성하는 것이 아니다.
  - 하나의 설계도만 작성하면 된다.

- 그러므로, generic type 을 사용하는 것은, 코드의 생산성을 높이게 된다.

Why Generic Type ? [4]

- 설계도가 유사하지만, 똑같지는 않다면?
  - 이 문제는, generic type 의 문제는 아니다.
  - (설계도) 의 상속 (inheritance) 을 고려해 본다.
    - 공통 부분은 상위 class 로 선언.
    - 다른 부분은 상위 class 를 상속받은 class 로 구현한다.





Generic Type:

- Java 는, 클래스를 설계할 때 그 내부에서 사용되는 자료형을 특정하지 않은 채로 통칭적인 (generic) 그래서 형식적인 식별자 (identifier) 를 사용할 수 있게 해 준다.
  - 이러한 형식적인 식별자로 정의되는 자료형을 generic type 이라고 한다.
  - 이 generic identifier 는 나중에 실제로 사용하는 시점에 실제로 존재하는 자료형 이름으로 대체가 된다.

- (예) 다음과 같은 식별자를 종종 사용한다. 그러나 식별자의 이름은 사용자가 임의로 정할 수 있다.
  - Class ArrayList <**T**>
    - Class "ArrayList"에서 원소의 type으로 사용되는 generic class "**T**"
  - Class LinkedNode <**E**>
    - Class "LinkedNode"에서 원소의 type으로 사용되는 generic class "**E**"

- 사용하는 관점에서, 동일한 class 일지라도 사용 시점에 generic type 이 다르게 주어진 class 들은 서로 다른 class 라고 할 수 있다:
  - Class "ArrayList<Student>" 와 class "ArrayList<Coin>" 은 서로 다른 class 이다.

Generic Type 을 배열 원소의 자료형으로 사용 [1]

```
public class  ArrayList <T> {
    private T[]  _elements ;


    public  ArrayList () {
       this._elements = (T[]) new Object [100];
    }
}
```

- 객체를 생성하는 new 를 사용할 때에는 generic type 이 아닌, 반드시 구체적인 type 으로 언급되어야 한다.
- Class "ArrayList" 컴파일 하는 시점에, T 의 자료형은 결정되어 있지 않다. 따라서, new 에 T 를 사용하면 컴파일러는 오류를 내보낸다. (오류 메시지: "Cannot create a generic array of T")
- 보통 배열을 생성하는 이런 경우에는, T 대신에 최상위 Class 인 "Object" 를 사용한다. 즉, 아무 원소나 저장할 수 있는 배열을 만드는 것이다.
- 컴파일러는 생성된 배열이 _elements 의 자료형과 일치하지 않다고 여전히 오류 메시지 "Type mismatch: cannot convert from Object[] to T[]" 를 내보낸다.

Generic Type 을 배열 원소의 자료형으로 사용 [2]

```
public class  ArrayList <T> {
    private T[]  _elements ;


    public  ArrayList () {
       this._elements = (T[]) new Object [100] ;
    }
}
```

- 자료형 불일치를 해결하기 위해서, new 앞에 "(T[])" 를 삽입하여 생성된 배열의 자료형을 강제로 T[] 로 변환시킨다.
- 이렇게 하여 컴파일 오류는 제거할 수 있다.
- 그렇지만, 나중에 사용자가 잘못 사용하여 자료형이 T 가 아닌 원소를 배열에 넣을 가능성이 있다. (앞에서 언급하였듯이 생성된 배열은 원소의 자료형이 "Object" 이므로 어떠한 자료형의 객체도 넣을 수 있다.)
- 이렇게 잘못 사용할 가능성이 있으므로 컴파일러는 경고 메시지를 내보낸다. (경고 메시지: "Type Safety: Unchecked cast from Object[] to T[]")

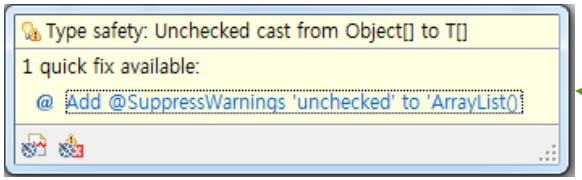
Generic Type 을 배열 원소의 자료형으로 사용 [3]

```
public class  ArrayList <T> {
    private T[]  _elements ;

    @SuppressWarnings("unchecked")
    public  ArrayList () {
       this._elements = (T[]) new Object [100] ;
    }
}
```

- 프로그램 작성자는 이러한 강제적인 자료형 변환으로 인하여 발생할 수 있는 위험성을 인지하고 대처할 수 있는 방안을 가지고 있어야 한다.
- 이를테면, 배열에 들어가는 객체의 자료형은 언제나 T 임 스스로 보장할 수 있고, 또 관련 프로그램 코드를 그에 합당하게 작성할 것임을 스스로 보장할 수 있어야 한다.
- 그렇다면, 프로그램 작성자는, "컴파일러는 이러한 강제적인 자료형 변환을 걱정할 필요가 없으므로, 자료형 변환으로 인한 점검을 하지 않아도 된다" 고 컴파일러 알려 더 이상 경고 메시지가 통보되지 않게 할 수 있다.

Generic Type 을 배열 원소의 자료형으로 사용 [4]

```
public class  ArrayList <T> {
    private T[]  _elements ;

    @SuppressWarnings("unchecked")
    public ArrayList () {
       this._elements = (T[]) new Object [100] ;
    }
}
```



오류를 클릭하면 이와 같은 창이 뜬다. 경고를 해결할 수 있는 방법을 선택하면 컴파일러가 자동으로 필요한 행위를 한다.

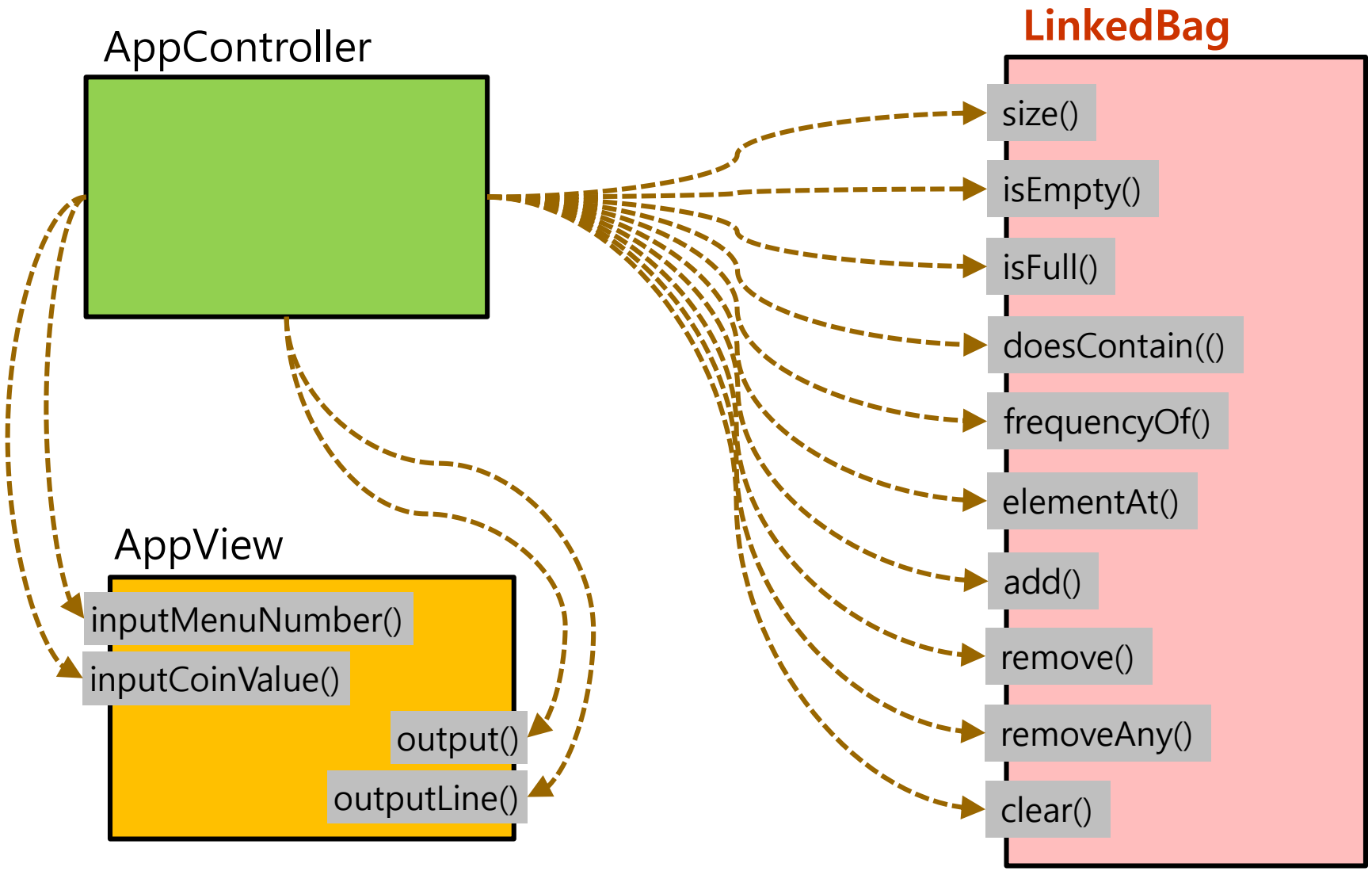
- 이러한 통보를 컴파일러에게 하기 위해, @SuppressWarning("unchecked") 를 삽입한다. 프로그램 작성자는 이 통보 사항을 직접 입력하기 보다는 오류를 클릭하여 해결책을 선택하는 방법으로 처리하게 된다.

과제에서 해결할 문제

문제

- 지난 주의 제 2 주차 문제와 동일
- 차이점:
  - ArrayBag 대신 LinkedBag 을 사용한다.
- 이번 주 과제는 이렇게 하자:
  - 지난주 과제를 복사하여, 이번 주 과제를 만든다.
  - Class "ArrayBag" 을 삭제한다.
  - Class “LinkedNode"와 "LinkedBag" 을 새로 추가한다.
  - AppController 에서 다음을 수정한다:
    - "ArrayBag" 으로 되어 있는 곳을, 모두 "LinkedBag" 으로 바꾼다.

MVC 구조


AppController
LinkedBag
size()
isEmpty()
isFull()
doesContain(()
frequencyOf()
elementAt()
add()
remove()
removeAny()
clear()
AppView
inputMenuNumber()
inputCoinValue()
output()
outputLine()

입출력에서 달라지는 점은?

❑ 출력의 예 [1]

```
<< 동전 가방 프로그램을 시작합니다 >>

? 동전 가방의 크기, 즉 가방에 들어갈 동전의 최대 개수를 입력하시오: 6

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
? 동전 값을 입력하시오: 5
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
? 동전 값을 입력하시오: 10
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
? 동전 값을 입력하시오: 20
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
? 동전 값을 입력하시오: 5
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
? 동전 값을 입력하시오: 30
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
? 동전 값을 입력하시오: 5
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 1
- 동전 가방이 꽉 차서 동전을 가방에 넣을 수 없습니다.
? 동전 값을 입력하시오: 100
- 주어진 값을 갖는 동전을 가방에 성공적으로 넣었습니다.
```

출력의 예 [2]

```
? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 2
? 동전 값을 입력하시오: 50
- 주어진 값을 갖는 동전은 가방 안에 존재하지 않습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 2
? 동전 값을 입력하시오: 5
- 주어진 값의 동전 하나가 가방에서 정상적으로 삭제되었습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 3
? 동전 값을 입력하시오: 20
- 주어진 값을 갖는 동전이 가방 안에 존재합니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 3
? 동전 값을 입력하시오: 70
- 주어진 값을 갖는 동전은 가방 안에 존재하지 않습니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 4
? 동전 값을 입력하시오: 5
- 주어진 값을 갖는 동전의 개수는 2 개 입니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 4
? 동전 값을 입력하시오: 80
- 주어진 값을 갖는 동전의 개수는 0 개 입니다.

? 수행하려고 하는 메뉴 번호를 선택하시오 (add: 1,  remove: 2,  search: 3,  frequency: 4,  exit: 9) : 9
- 가방에 대한 수행을 종료합니다.

가방에 들어 있는 동전의 개수: 6
동전 중에서 가장 큰 값: 100
모든 동전들의 값의 합: 170

<< 동전 가방 프로그램을 종료합니다 >>
```

AppController 에서 달라지는 점은?

AppController: run() 수정할 곳 [1]

```
public class AppController {
    // 상수
    ......

    // 비공개 인스턴스 변수들
    private LinkedBag<Coin> _coinBag ;

    // 생성자
    ......

    // 공개함수
```

"ArrayBag" 을 "LinkedBag" 으로 수정 할 것

AppController: run() 수정할 곳 [2]

```
// Public Method
    public void run() {
        AppView.outputLine("<<< 동전 가방 프로그램을 시작합니다 >>>") ;
        AppView.outputLine("");

        int coinBagSize = AppView.inputCapacityOfCoinBag ();

        this.setCoinBag ( new ArrayBag<Coin> (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) ;
            }
            menuNumber = AppView.inputMenuNumber() ;
        }
        this.showStatistics();
        AppView.outputLine("<<< 동전 가방 프로그램을 종료합니다 >>>") ;
    }
} // End of class  "AppController"
```

삭제할 것 (이유는?)

"ArrayBag<Coin> (coinBagSize)" 를
"LinkedBag<Coin>()" 으로 수정 할 것

이 과제에서 필요한 객체는?

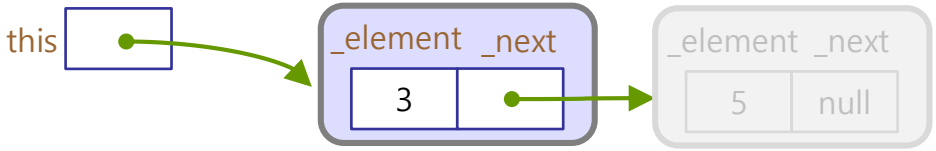
- AppController
- AppView
- Model
  - LinkedBag<E>
  - LinkedNode<E>
  - Coin

새롭게 추가할 Class 들

Class “LinkedNode<E>”

LinkedNode: 비공개 인스턴스 변수

```
public class LinkedNode<E> {
    // 비공개 인스턴스 변수
    private E                _element;    // 현재 노드에 있는 코인
    private LinkedNode<E> _next;          // 다음 노드
```



LinkedNode: 공개 함수

- LinkedNode의 Public Member function의 사용법
  - public LinkedNode()
  - public LinkedNode (E givenElement)
    - 원소 givenElement 를 갖는 LinkedNode 객체를 생성
  - public LinkedNode (E givenElement, LinkedNode<E> givenNext)
    - 원소 givenElement 와 다음 노드 givenNext 를 갖는 LinkedNode 객체를 생성
  - public E element()
    - LinkedNode 객체에 있는 element 를 얻는다.
  - public LinkedNode<E> next()
    - LinkedNode 객체의 다음 LinkedNode 객체를 얻는다.
  - public void setElement (E newElement)
    - LinkedNode 에 있는 element 를 newElement 로 변경한다.
  - Public void setNext (LinkedNode<E> newNext)
    - LinkedNode 객체의 next 를 newNext 로 변경한다.

LinkedNode: Member Functions의 구현

생성자

- public LinkedNode ()
  - _element 를 null 로 초기화 한다.
  - _next 를 null 로 초기화 한다.
- public LinkedNode (E givenElement)
  - _element 를 givenElement 로 초기화 한다.
  - _next 를 null 로 초기화 한다.
- public LinkedNode (E givenElement, LinkedNode<E> givenNext)
  - _element 를 givenElement 로 초기화 한다.
  - _next 를 givenNext 로 초기화 한다.

LinkedNode: Getter/Setter

LinkedNode 의 Getter/Setter

- public E element()
  - _element 의 값을 돌려준다.
- public void setElement (E newElement)
  - _element 의 값을 newElement 의 값으로 설정한다.

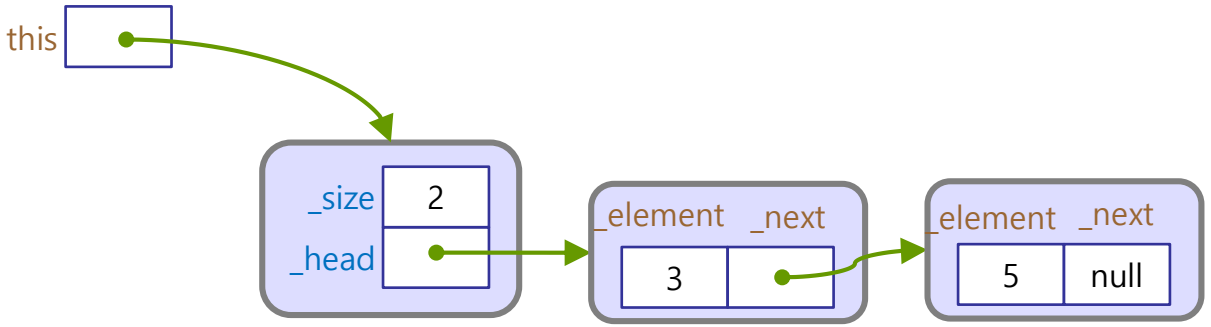
- public LinkedNode<Element> next()
  - _next 의 값을 돌려준다.
- Public void setNext(LinkedNode<Element> newNext)
  - _next 의 값을 newNext 의 값으로 설정한다.

Class “LinkedBag<E>”

LinkedBag: 비공개 인스턴스 변수

```
public class LinkedBag<E> {
    // 비공개 인스턴스 변수
    private int     _size ; // 가방이 가지고 있는 원소의 갯수
    private LinkedNode<E> _head ; // 연결 체인의 맨 앞 노드를 소유한다.
```

이 두 instance variable 에 대한 getter/setter 를 반드시 만들 것. 공개함수 "size()"를 제외한 나머지 모두 "private" 임의 유의할 것.



LinkedBag: 공개함수 [1]

LinkedBag 의 Public Member function의 사용법

- public LinkedBag ()
  - LinkedBag의 생성자
- public int size ( )
  - Bag에 들어있는 개수를 확인한다.
- public boolean isEmpty ()
  - Bag이 비어있는지 확인한다.
- public boolean doesContain (E anElement)
  - Bag 안에 주어진 원소가 존재 하는지 확인한다.
- public int frequencyOf (E anElement)
  - Bag 안에 주어진 원소가 몇 개 있는지 확인한다.
- public E elementAt (int anOrder)
  - Bag 안에 주어진 순서의 원소를 얻는다.
  - 가방의 개념상, 원소의 순서는 의미가 없다. 단지 가방 안의 원소를 하나씩 차례로 얻기 위한 수단으로 "elementAt()" 을 사용한다.
  - Class LinkedBag 을 위한 반복자 (iterator) 를 구현하여 사용하게 되면, "elementAt()" 은 필요 없으며, 따라서 삭제하는 것이 적절하다.

LinkedBag: 공개함수 [2]

- LinkedBag 의 Public Member function의 사용법
  - public boolean add (E anElement)
    - Bag 에 원소를 추가한다.
  - public boolean remove (E anElement)
    - Bag 에서 원소를 삭제한다.
  - public E removeAny ()
    - Bag 에서 원소 하나를 무작위로 삭제한다.
  - public void clear ()
    - Bag 을 초기화 한다.
    - 모든 원소를 삭제한다.

LinkedBag: Method 구현

- 강의 슬라이드를 볼 것.

요약

확인하자

- 다음의 내용을 잘 이해했는지 확인하자.
  - Model-View-Controller
  - LinkedBag 의 구현:
    - 연결 체인 취급 방법
  - ArrayBag 과 LinkedBag 의 구현의 차이점
    - 각 함수의 처리 시간은?
      - (예) elementAt(anOrder) 에서, anOrder 의 값에 따라 함수의 처리 시간은?
    - removeAny() 에서, 각각 어느 원소를 얻는 것이 좋았던 것인가?
      - ArrayBag 에서, 맨 앞 원소를 삭제한다면?
      - Linkedbag 에서, 맨 뒤 원소를 삭제한다면?
  - Generic Type 이름:
    - <E> 와 <Element> 는?

생각해 볼 점

⇒ 확인 내용 중에서,

class “ArrayBag” 과 class “LinkedBag” 의 구현의 차이점에 대해, 특히 성능의 차이점에 대해, 자신의 의견을 보고서에 작성하시오.

[실습 끝]