**자료구조 실습 보고서**

[제03주] 동전가방(ArrayBag)

**2021년 3월 22일**

**201702039 오명주**

1. **프로그램 설명서**
2. **프로그램의 전체 설계 구조**

* MVC (Model – View – Controller) 구조

Model : 프로그램이 “무엇”을 할 것인지 정의. 사용자의 요청에 맞는 알고리즘을 처리하고 DB와 상호작용하여 결과물을 산출하고 Controller에게 전달.

View : 화면에 무엇인가를 “보여주기 위한” 역할. 최종 사용자에게 “무엇”을 화면으로 보여줌.

Controller : 모델이 “어떻게” 처리할 지 알려주는 역할. 사용자로부터 입력을 받고 중개인 역할. Model과 View는 서로 직접 주고받을 수 없음. Controller을 통해 이야기함.

* 동전가방(ArrayBag) 프로그램에서의 각 클래스 별 MVC 구조 역할

Model :

* ArrayBag<E> : Generic Type으로 정의하여 가방의 구조와 무관하게 가방에 넣을 원소는 필요에 따라 결정되도록 한다. 동전 가방의 기능을 구성한다.
* Coin : 가방에 넣을 ‘동전’을 구체화하는 클래스. 동전의 변수, 속성으로 이루어져 있다.

View :

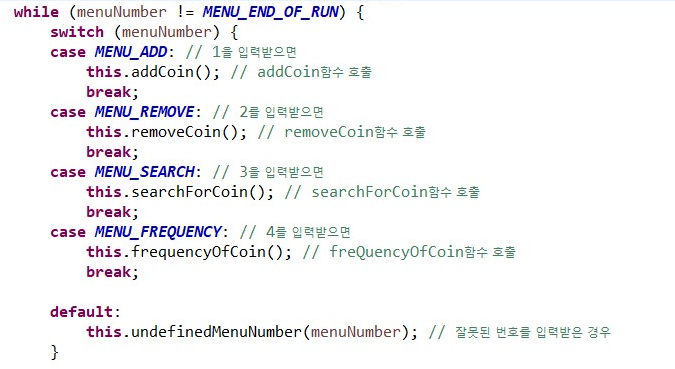
* AppView : 프로그램의 입/출력을 담당한다.

Controller :

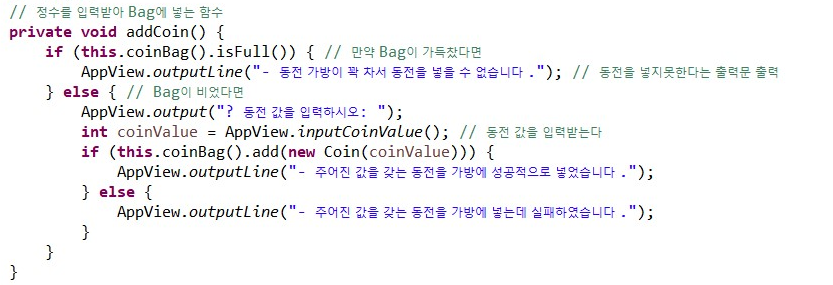
* AppController : AppView를 통해 수행할 메뉴 번호를 입력 받아 Model에 해당하는 클래스들에 전달하고 결과물을 AppView를 통해 출력한다.

1. **함수 설명서**

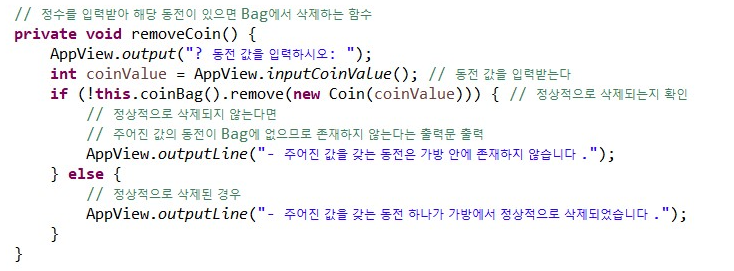
* 주요 알고리즘



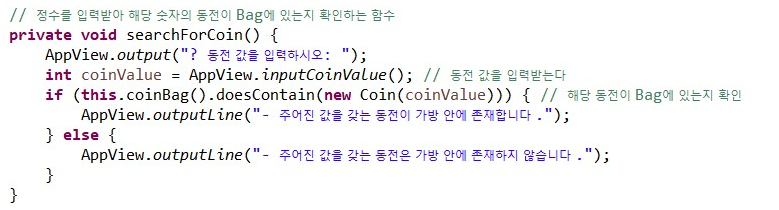
사용자로부터 수행할 메뉴 번호를 입력 받는다. 가방에 동전을 넣는 MENU\_ADD, 가방에서 해당 동전을 제거하는 MENU\_REMOVE, 해당 동전이 있는지 없는지 확인하는 MENU\_SEARCH, 해당 동전에 가방에 몇 개 있는지 확인하는 MENU\_FREQUENCY로 구성되어 있다. 해당 상수들은 Enum 클래스로 정의해도 되지만 switch문을 실행하는 AppController클래스에 정의하였다.



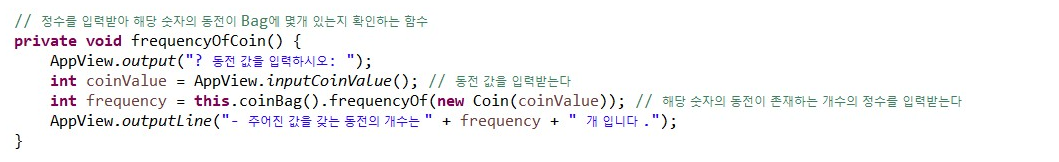
‘1’을 입력 받으면 호출되는 addCoin 함수이다. 가방이 모두 찼는지 확인하는 isFull 함수 호출 후 비어 있다면 동전 값을 입력 받아 동전 객체 생성 후, 성공적으로 넣었다는 출력문을 출력한다. 이때, 출력문 역시 AppView를 활용한다. 실제 넣는 함수 행위는 ArrayBag 클래스의 add 함수에서 행한다.



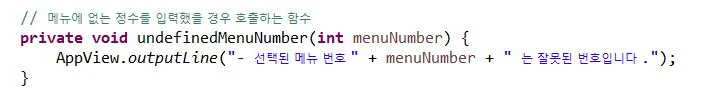
‘2’를 입력 받으면 호출되는 removeCoin 함수이다. 삭제할 동전 값을 입력 받고 해당 값의 동전 객체 생성 후 ArrayBag 클래스의 remove 함수를 이용하여 삭제가 되는지 확인한다. 정상적으로 삭제되지 않는다면 주어진 값의 동전에 Bag에 없으므로 존재하지 않는다는 출력문을 출력한다. 정상적으로 삭제된 경우 삭제되었다는 출력문을 출력한다.



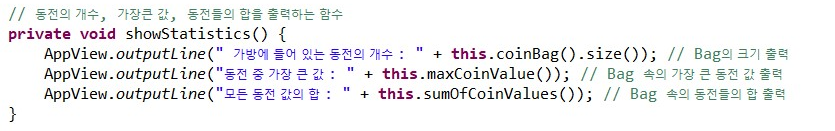
‘3’을 입력 받으면 호출되는 searchForCoin 함수이다. 찾을 동전 값을 입력 받고 해당 동전이 Bag에 있는지 확인한다. 이때 역시 실제 찾는 행위는 ArrayBag 클래스의 doesContain 함수에서 행한다.



‘4’를 입력 받으면 호출되는 frequencyOfCoin 함수이다. 찾을 동전 값을 입력 받고 해당 동전에 Bag에 몇 개 있는지 알아본다. 실제 찾는 행위는 ArrayBag 클래스의 frequencyOf 함수를 이용한다.



잘못 된 번호를 입력한 경우 호출되는 함수.

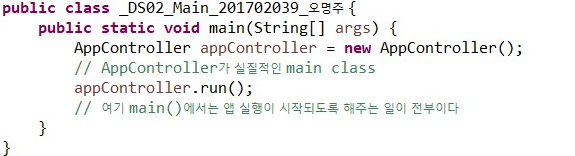


‘9’를 이용하여 프로그램이 종료될 때 출력되는 통계를 나타내는 함수.

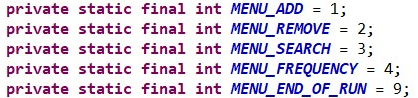
* maxCoinValue 함수에서는 Bag 속의 가장 큰 동전 값을 찾기 위해 for문을 Bag의 Size만큼 반복하여 원소 값을 모두 비교하여 max 값을 저장하여 반환한다.
* sumOfCoinValues 함수에서는 마찬가지로 for문을 Bag의 Size만큼 반복하여 원소 값을 모두 더하여 반환한다.

1. **종합 설명서**

* 프로그램 실행 순서대로 설명해보자.

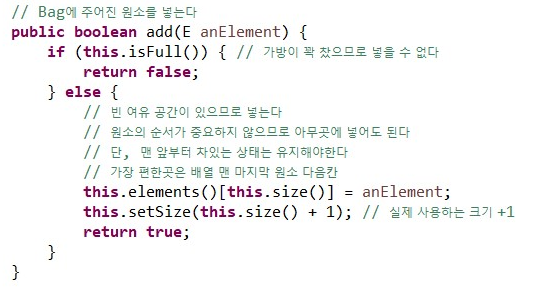
****

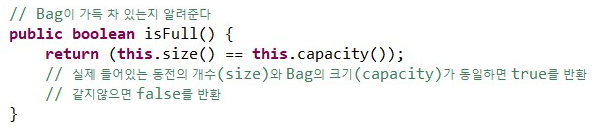
Main 클래스에서는 AppController 객체 생성 후 run함수만 호출한다.

정의 해 놓은 상수를 이용하여 번호를 입력 받고 해당 메뉴를 수행한다.

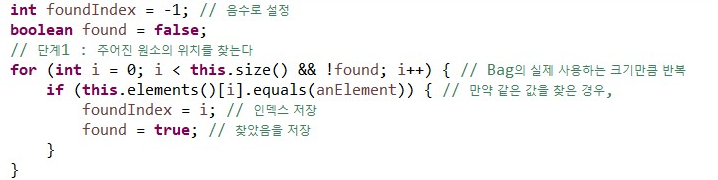
* ArrayBag 클래스

1. add 함수

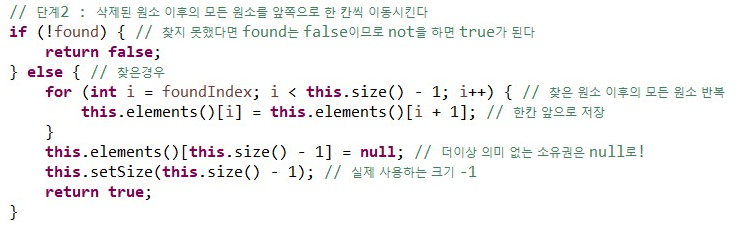


* isFull 함수로 가방이 가득 찼는지 확인하고 가득 찼다면 false를 반환
* 가득 차 있지 않다면 마지막 원소 다음 칸에 입력 받은 값(객체)를 넣는다.
* Size를 1 증가 시켜 다시 설정해준다. 그리고 true를 반환한다.
* Bag이 가득 차 있는지 확인하는 isFull 함수. Size가 Bag의 크기와 일치하는지 확인.

1. remove 함수

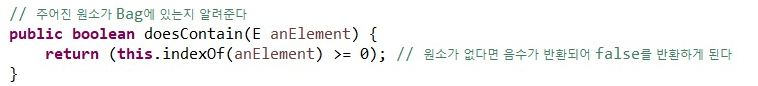


* 주어진 원소의 위치를 for문을 통해 찾는다. 이때, Coin 클래스의 equals 함수를 활용하는데 해당 객체를 찾은 경우 인덱스와 찾았는지 여부를 변수에 저장한다.



* 찾지 못한 경우 false를 반환
* 찾은 경우 찾은 원소 이후 인덱스부터 모든 원소를 한 칸 앞쪽으로 저장한다. For문 활용.
* Size는 1 감소하여 저장. 삭제한 부분은 null 처리. True 반환

1. doesContain 함수

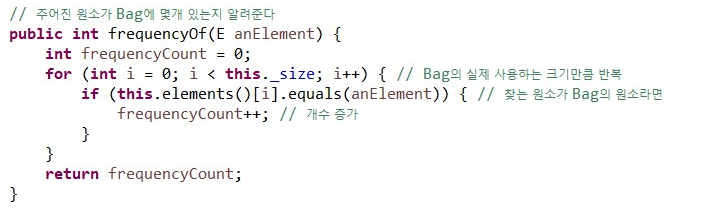


해당 함수에서는 indexOf 라는 private 함수를 호출하여 활용하였다.



* 인덱스를 저장할 foundIndex 변수를 생성 후 음수로 초기화한다.
* size만큼 반복하는 for문을 이용하여 찾는 원소가 있는지 확인한다.
* 만약 찾는다면 해당 원소의 index를 저장한다. return했을 때 찾은 경우 양수가 되고 못 찾은 경우 음수가 된다.

1. frequencyOf 함수



주어진 원소가 몇 개 있는지 저장할 frequencyCount 변수를 선언하고 0으로 초기화한다. Bag의 size만큼 반복하여 해당 원소를 찾는다. 찾는다면 frequencyCount 변수를 개수 증가한다. 하나도 없다면 그대로 0을 반환한다. 이때도 Coin 클래스의 equals 함수를 활용하였다.

* Coin 클래스



객체의 값을 비교하는 equals 함수이다. getClass 함수를 이용하여 Class 정보를 받아와 먼저 비교하고 Coin 클래스와 동일하지 않다면 false를 반환한다. 만약 동일하다면 해당 객체의 값을 비교하는데 값을 비교하여 일치하면 true를, 일치하지않으면 false를 반환하도록 구현하였다.

1. **프로그램 장단점 / 특이점 분석**

* **장점**
* MVC 모델을 이용하여 가독성과 생산성이 뛰어나다. 각 클래스, 함수의 역할이 분명해서 코드와 프로그램을 잘 이해할 수 있다.
* 유사한 기능을 하는 다른 프로그램에도 재사용할 수 있다. 객체 지향 프로그램의 가장 큰 장점이라고 할 수 있다.
* 수정이 편리하다. 데이터나 기능을 수정하려고 하면 해당 메소드만 수정하면 되기 때문에 편리하다.
* Generic 타입을 활용하여 프로그램 성능저하를 유발하는 Type Casting을 제거한다.  
  코드 절약 및 코드 재사용성을 증진시켜 유지보수가 편하다  
  엄격한 데이터 타입 체크를 가능하게 한다.
* **단점**
* 처음에 클래스와 함수 역할을 뚜렷하게 나누는 것이 쉽지않다. 객체 지향 프로그램 설계할 때 시간이 오래 걸린다.
* 순서가 상관이 있는 ArrayBag이었다면 함수를 수정해야 하고 추가해야 할 것이다. 재사용성이 상황에 따라 감소한다.
* 입/출력까지 모두 분리 하다 보니 코드양이 많아지고 시간이 오래 걸린다.

1. **실행 결과 분석**
2. **입력과 출력 (화면 capture하여 제출)**

****

****

1. **결과 분석 (자신의 논리적 평가, 기타 느낀 점)**

* Generic Type을 이용한 구현이 쉽지는 않았고 다시 처음부터 구현하라고 한다면 다시 할 수 있을까라는 생각이 들 정도로 익숙하지 않은 구현이었다. 그래도 Generic Type을 이용하니 코드가 유연해지고 깔끔하다는 생각이 들었다. 이 과제에서는 Coin 클래스를 정의하여 이용했지만 이를 바꾸어 구현한다면 재사용성 면에서도 꽤 유용하다고 생각된다.
* 지난 주 과제와 동일하게 객체 지향적인 과제를 수행했는데 AppView를 재사용하면서 다시한번 객체 지향 설계의 장점을 되새긴 것 같았다. 출력문을 출력하는 함수는 따로 구현없이 재사용하였다.

1. **소스코드**

**[main class]**

