HW#1. 카드 매칭 게임

20132927 정재명

누구나 한번쯤, 카드 맞추기 게임을 해본 적이 있을 것이다. 카드 두 개의 모양이 똑같고, 이런 쌍이 여러장 뒤집혀 있다. 앞 면을 모른채, 두 명이 번갈아 카드를 두 장씩 앞면으로 뒤집어 확인한다. 그리고 두 장이 똑같은 모양이면 점수를 얻고 그대로 앞면으로 둔다. 두 카드가 모양이 다르면 다시 뒷면으로 둔다. 그리고 모든 카드가 앞면이 되면 게임은 종료된다. 이런 게임은 누구나 한번 쯤 해보았을 것이다. 이번 과제에서는 이 게임을 콘솔에서 구현해 본다.

게임은 다음과 같은 구성을 가진다.

- 16장의 카드 사용. 1번부터 8번까지의 숫자카드가 각각 2장씩 존재함.
- 게임을 시작하면 16장의 카드를 무작위로 섞어 4 X 4 로 배치한 후, 뒷면이 보이게 둠.
- 두 플레이어가 번갈아가며 카드를 두 장씩 선택하여 확인함. 만약 동일한 숫자가 나올 경우, 1점을 획득하며 해당 카드는 앞면이 보이게 둠.
- 모든 카드쌍을 찾으면, 두 플레이어의 점수를 보여주며 게임 종료.
- 특별규칙: 7번 카드쌍은 2점.

우리가 이미 알고 있는 카드 매칭 게임과 거의 똑같다. 다만, 다른 카드를 골랐을 경우 다시 앞면으로 뒤집지 않는다. 이것은 콘솔이라는 환경 때문으로 보인다. 뒤집어봤자 앞선 결과가 그대로 보이기 때문에 다시 앞면으로 돌려 놓는 것이 의미가 없다. 물론, 화면은 모두 지우고 다시 출력할 수도 있지만, 이번 과제에서 거기까지는 생각하지 않고 진행하기로 한다.

그리고 다음과 같은 제약 사항을 가진다.

- 카드의 위치는 행과 열을 표현한 두개의 정수로 입력받는다.
- 카드의 앞면은 1~8 숫자, 뒷면은 x로 표시한다.
- 카드 배치는 고정 배치와 무작위 배치 버전 모두를 지원한다.

여기까지가 게임 구현을 위한 요구 조건이다. 이제 본격적으로 게임을 만들어 보겠다.

1. 문제 정의(필요한 기능)

먼저, 게임에 필요한 기능들을 게임의 진행 순서대로 살펴보겠다. 가장 먼저, 게임이 시작되면, 게임 보드판을 출력해야 한다. 이 보드판은 4*4 카드 배열을 포함하고 행과 열의 번호도 포함된다. 카드는 모두 뒷판을 의미하는 x로 출력한다.

그리고 바로 유저로부터 입력을 받는다. 입력은 두 개의 정수를 받는데, 범위를 벗어나거나 이미 정답인 카드를 선택할 시에 잘못 에러처리를 해야한다. 입력을 다시 받도록 할 것이다. 또한 이 입력값은 정수이지만 사실 십의 자리 숫자가 row를 의미하고 일의 자리 숫자가

column을 의미한다. 그래서 이 값으로 변환하는 기능이 필요하다.

이렇게 입력을 받고나면, 선택된 카드 두 장이 모두 같은 카드인지 따져본다. 이를 위해 출력을 위한 배열과 앞면 데이터를 가진 배열 두 개를 따로 관리할 것이다. 출력을 위한 배열은 실제 눈에 보이는 게임 보드판을 의미하고, 앞면 데이터는 실제 카드의 값을 의미한다. 이렇게 하면, 출력할 때 마다 배열을 새로 만들지 않아도 되고, 프로그램이 끝날 때 까지 수정되면 안되는 앞면 데이터도 쉽게 관리할 수 있다.

선택된 카드가 같은지 확인할 때는, 선택한 위치의 앞면 데이터를 읽어 와서 확인한다. 그리고 같으면, 출력 배열에 이 값을 대입하여, 다음 출력 때 부터는 앞면으로 보이게 출력할 수 있게 한다. 그리고 같지 않으면, 출력 배열을 수정하지 않는다. 그러면 다음 출력때는 선택했던 위치의 카드가 다시 뒷면으로 출력된다. 그리고 출력할 때, 해당 위치가 이번에 선택된 위치인지 확인하고 만약 그렇다면 앞면이 보이게 바꾸어서 출력하도록한다. 물론 이 경우에는 출력 배열을 수정하는 것이 아니기 때문에 다음 출력에 영향을 미치지 않는다.

이렇게 카드를 확인하고 나면, 맞춘 경우에 유저의 점수를 올려주고 그렇지 않은 경우에는 해당 유저가 실패했다는 메시지를 출력한다. 이 때 서로 다른 두 유저의 데이터가 식별될 필요가 있다. p1의 순서인데 p2라고 출력되거나, p2의 점수가 올라가면 안된다. 또한 7이 적힌 카드를 맞추면 2점을 올려주어야 한다.

이렇게 다음 턴이 되면, 다시 입력을 받고 똑같이 반복한다. 모든 카드가 앞면이 되면 게임을 종료한다.

여기에 카드 배치를 랜덤으로 하는 버전도 추가한다. 처음 게임을 시작하기 전에 카드를 섞을 것인지 고정 배치 모드를 그대로 사용할 것인지 사용자로부터 선택할 수 있도록 한다. 카드 앞면 데이터를 저장하고 있는 배열은 디폴트로 특정 순서로 할당되어 있다. 고정 배치모드를 선택하면 이 배열이 그대로 사용된다. 랜덤 배치를 하겠다고 하면 그때 이 배열을 섞어준다.

2. 문제 해결(기능 구현)

앞선 기능들을, 몇 개의 클래스로 구성해 보았다.

User 클래스

사용자의 점수와 식별자를 가지고 있다. 생성시에 아이디를 부여한다. 나중에 화면에 사용자의 이름을 출력할 때 이 식별자를 사용한다. 점수를 올리는 메서드에는 7번 카드인지 아닌지 확인하는 로직을 추가했다.

CardLocation 클래스

사용자가 입력한 위치 값을 관리한다. 입력 받은 두 정수를 나누기와 모듈러 연산을 통해 좌표로 사용할 수 있는 값으로 바꾼다. 예를 들어 13이 입력되었다면, 13/10과 13%10연산을 통해 각각 1과 3을 얻어낼 수 있다. 이렇게 얻어낸 값을 이 클래스의 멤버변수에 할당하여 보관하고 필요할 때 메서드로 접근해서 사용한다.

CardManager클래스

앞면 카드 값을 가지고 있다. 출력을 2차원 배열의 형태로 되지만, 이 카드 값들은 1차원 배열에 저장했다. 이게 섞기가 쉽다고 판단했다. 우선 1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1 이 순서대로 값을 할당한다. 하지만, 섞는 버전을 선택한 경우에는 shuffle함수를 통해 섞어준다.

16개의 요소를 가지는 1차원 배열을 섞는 방법은 간단하다. 우선 두 개의 인덱스 변수를 준비하고, rand()%16 모듈러 연산을 통해 0~15값 중 하나를 인덱스 변수에 할당한다. 이 인덱스를 카드 배열의 인덱스로 사용하여 두 개의 요소에 접근한다. 그리고 temp값을 사용하여 두 개의 변수에 있는 값을 swap하는 로직을 사용한다. 그러면 16개의 요소에 랜덤하게 접근하여 두 값을 바꾸게 되는 것이다. 이 과정을 반복문을 통해 여러번 해주면 충분히 섞이게 된다.

이 클래스 객체가 가진 카드 리스트 배열은 프로그램이 종료될 때까지 변하지 않는다. 오직 이 클래스는 카드 앞면 데이터를 유지하고, 필요하면 확인하기 위해 해당 위치의 카드 값을 반환할 뿐이다.

BoardManager 클래스

실제 카드 게임이 이루어 지는 보드판이라고 생각하면 된다. 출력할 보드판 2차원 배열 데이터를 가지고 있고 동시에 CardManager의 객체를 가지고 있다, 앞면을 확인할 필요가 있을 때 마다 CardManager 객체를 참조한다. 이렇게 CardManager 객체를 참조하는 것은 카드를 뒤집어 앞면을 확인하는 행위라고 생각하면 된다. 이러한 역할을 하는 BoardManager 클래스는, 카드의 앞면을 확인하여 두 카드가 같은지 확인하거나, 또는 게임판을 출력하는 기능을 제공한다.

여기서 printBoard를 오버로딩하여 두 개의 메서드를 제공한다. 매개변수가 없는 print는 이미 가지고 있는 게임 보드는 그대로 출력한다. 매개변수로 CardLocation 객체를 사용하는 print함수는, 해당 location에 있는 카드 값만 앞면으로 바꿔서 출력한다. 하지만 원본 게임보드 데이터는 수정하지 않는다. 이것은, 정답이 아닌 경우에 출력 보드는 수정하지 않고 다만,

선택했다는 의미로 선택한 카드를 앞면으로 출력하기 위해 필요하다.

GameManager 클래스

마지막으로 이 모든 기능을 적절히 배치하여 게임을 진행시키는 클래스다. 게임이 진행되는 순서에 맞게 필요한 기능들을 가지고 있다. User객체와 BoardManager 객체를 필요한 순간에 적절히 활용하여 게임을 진행시킨다.

여기서 중요한 것은 두 개의 User객체를 식별하는 방법이다. User타입의 배열을 만든다. 그리고 멤버변수 turnCheckNumber를 활용하여 인덱스값으로 사용하면, 턴이 바뀔 때 마다 사용되는 User 객체가 바뀌게 할 수 있다. turnCheckNumber는 턴이 지나갈 때 마다 1씩 증가한다. 그리고 이 값을 %2 연산을 하면, 0 1 0 1 ... 과 같이, 매 턴 마다 0과 1이 번갈아 나온다. 이 값을 인덱스로 사용하면 User 배열의 두 객체를 번갈아 사용할 수 있다. 이렇게 하면 굳이 두 개의 각체를 따로 만들고 턴마다 조건문으로 확인하여 식별하지 않아도 된다.

카드를 확인하는 checkRight 메서드에서는 정답 여부에 따라 보드판을 수정해주어야 한다. numberOfMatchedCardPair 멤버변수는 두 개의 똑같은 카드를 선택할 때 마다 1씩 증가한다. 이 변수의 값이 8이라면, 모든 카드가 앞면, 즉 게임이 끝났음을 의미한다. checkRight 메서드는 이 때 false를 반환한다. main 함수에 있는 while루프에서 if문으로 이 반환 값을 확인하여 while문을 빠져나올지 결정한다. numberOfMatchedCardPair 값이 8이면 false가 반환되어 while문을 빠져나와 게임이 끝나게 된다.