Principles of Java Language with Applications, PIC20A
Minh Pham
Spring 2022

Homework 2
Due 11:59pm, Tuesday, April 26, 2022

Download the starter code `PlayCard.java`. It should give you an idea of how we intend to use the `class MatchCardGame`. We may take off up to 20% of the total marks for poor style; make sure to name your variables reasonably, indent properly, and comment sufficiently. Put `PlayCard.java` and `MatchCardGame.java` in the package `hw2`. Zip the folder `hw2` under the name with format `FirstName_LastName.zip` and submit this zip file.

This homework is worth 130 points which is corresponding to 13% of your total grade. **This is a must-do homework**.

Problem 1: (Pairs)

In a card game of *Pairs* the goal is to turn over pairs of matching cards.
`https://en.wikipedia.org/wiki/Concentration_(game)`
Here are the rules for the variation of Pairs we consider.

At the start of the game, there are `n` cards face-down, where `n` is a multiple of `4`. There are `4` cards of each type, and the cards are labeled with letters `a`, `b`, ... For example, if `n==24`, there are 6 types of cards: `a`, `b`, `c`, `d`, `e`, and `f`. Say `4=1*4<=n` and `n<=4*26=104`.

At each turn, the player flips 2 cards, one at a time, that are face-down. If the 2 flips are of the same type, the matched cards are left face-up. If the 2 flips mismatch, the mismatched cards are returned to the face-down position. The game ends when all cards are matched, and the score is the total number of flips made. (Flipping a pair of cards counts as 2 flips, so the best possible score is `n` flips.)

Write a `pubic class` titled `MatchCardGame` with the following members that implement this game of Pairs. The starter code in `PlayCard.java` should give you an idea of how `MatchCardGame` is used.

`MatchCardGame` should have no other `public` fields, methods, and constructors aside from the ones specified. However, it may and should have additional `private` fields, methods, or constructors.

The field

```
public final int n;
```

is the size of the game set by the constructor.

The field

```
private final char[] CardValues;
```

is the array containing the `char` values of `n` cards of the game.

The constructor

```
public MatchCardGame(int n);
```

initializes a card game with a total of `n` cards. Assume `n` is a multiple of `4` and that `4<=n && n<=4*26`. Without shuffling (explained in Problem 2) cards `0`,`1`,`2`, and `3` should be `a`, cards `4`,`5`,`6`, and `7` should be `b`, and so on.

The method

```
public String boardToString();
```

converts the state of the board to an appropriate `String` representation. You have freedom to choose your own representation, but it must reasonably represent the state of the game.

The method

```
public boolean flip(int i);
```

plays card number `i`. If card `i` cannot be played because it's face-up, or if `i` is an invalid card number, then `return false`. If `i` is a card number that can be played, play card `i` and `return true`.

The method

```
public boolean wasMatch();
```

`return`s `true` if the previous pair was a match and `return`s `false` otherwise. This method should be called only after `flip` has been successfully called an even number of times and before `flipMismatch` is called. (A successful call of `flip` is a call that results in a flip and `return`s `false`.)

The method

```
public char previousFlipIdentity();
```

returns the face of the previously flipped card as a `char`. This method should only be called after a card has been flipped.

The method

```
public void flipMismatch();
```

reverts the a mismatched pair to face-down position. This method should only be called after a 2 calls of `flip` results in a mismatch.

The method

```
public boolean gameOver();
```

`return`s `true` if all cards have been matched and the game is over and `return`s `false` otherwise.

The method

```
public int getFlips();
```

`return`s the total number of card flips that have been performed so far.

Remark. MatchCardGame represents the physical state of the game, not the player. MatchCardGame has nothing to do with game strategies a human or AI player might employ.

Remark. The problem specifies how these methods should be used. For example, the input n of MatchCardGame's constructor needs to be a multiple of 4. You do not have to do any sort of error handling when these requirements are violated, although using asserts will help debugging. Note that main of PlayCard.java uses MatchCardGame in compliance to the spefications.

Remark. We will not use PlayCard or MatchCardGame's main function in the grading.

Remark. You might need some helper functions for your program. It is your choices and try to be creative, logical and optimize your code if possible.

Problem 2: (Shuffle)

For the class MatchCardGame write the method

```java
public void shuffleCards();
```

This method shuffles the cards using the Fisher-Yates shuffle. This method should be called before any flips have been made.

Problem 3: (Game AIs)

Within the `class PlayCard`, write the following methods that automatically play the game.

The first method

```
public static int playRandom(MatchCardGame g);
```

plays the game by flipping a legal random card. I.e., at each turn, play one of face-down cards with equal probability independent of any past plays. (This "AI" is the least intelligent player.) The method plays until the game is over and `return`s the total number of flips.

The second method

```
public static int playGood(MatchCardGame g);
```

plays the game with perfect memory. After an even number of flips, if there is a known face-down pair, play the pair. Otherwise, randomly play, with equal probability, one of the unknown face-down cards. After an odd number of flips, if there is a known face-down match, play the match. Otherwise, randomly play, with equal probability, one of the unkown face-down cards. The method plays until the game is over and `return`s the total number of flips.

Remark. These AIs should perform the same whether or not the cards are shuffled.

Remark. When `n` is the board size, the good AI will always score `2n` or better.

Problem 4: (Monte Carlo)

Within `PlayCard`, write the following methods that play the game many times with the AIs.

The first method

```
public static double randomMC(int N);
```

plays shuffled `MatchCardGame`s of size `32` a total of `N` times using `playRandom` method. The method `return`s the average number of flips to complete the games.

The second method

```
public static double goodMC(int N);
```

plays shuffled `MatchCardGame`s of size `32` a total of `N` times using `playGood` method. The method `return`s the average number of flips to complete the games.