CSE1341 - Lab 6 Assignment

Overview

In Lab 5 you created the Deal or No Deal game using structured programming techniques. In this lab, you will rewrite your application using object-oriented programming.

Pre-Lab (5 Points)

Create the class named *Case* found in the instructions. Include the four attributes, constructor and getter and setter methods for each of the attributes.

Lab (95 Points)

Create the DealGame system using object oriented programming, following the design provided on the following pages. Your output should match the format shown on the last page, although your actual output will vary based on the the outcome of the game.

Submit the *java* and *class* files via Canvas (as a single zip-file). Include a comment block at the top of each *Java* file that includes your name, student id number, and "Lab 6 - Spring 2019".

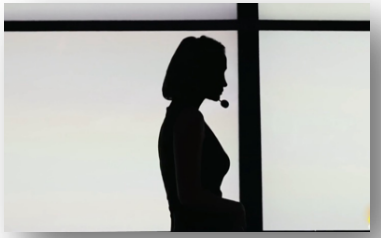
NOTES:

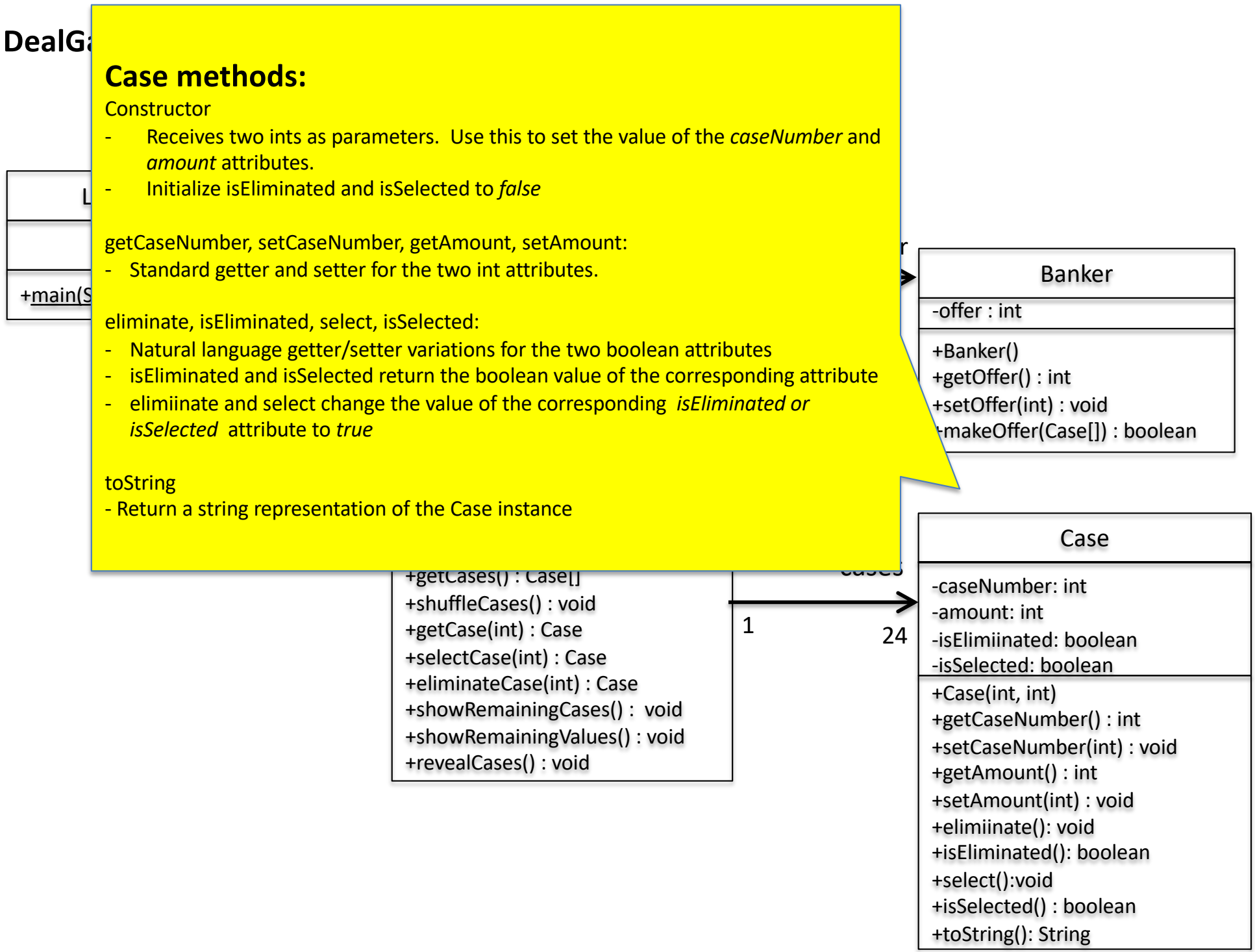
Each program should include comments that explain what each block of code is doing. Additionally, the programs should compile without errors, and run with the results described in the exercise. The following deductions will be made from each exercise if any of the following is incorrect or missing:

Proper formatting [5 points]
Proper names for classes and variables [5 points]
Comments [5 points per class]
Program doesn't compile [5 points for each minor error up to 5 errors provided that after fixing the errors the program compiles. If the program does not compiler after the 5 errors are fixed, partial credit will be given not to exceed 50 points]
Source code (java file) missing [10 points]
Executable (class file) missing [10 points]
Missing array where an array was required [5 points each]
Missing loop where a loop was required [5 points each]
Missing class from the design provided [10 points each]
Missing method from the design provided [5 points each]

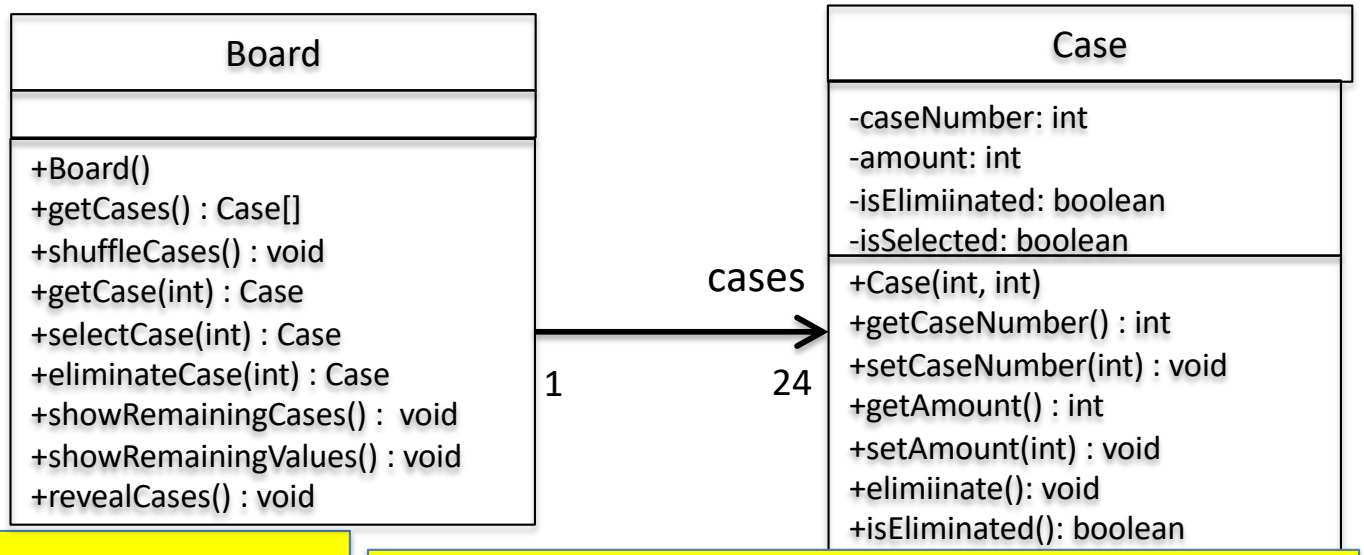
This Lab is due Saturday April 13 at 6:00am.

DealGame Design




DealGa
L
+main(S
Case methods:
Constructor
- Receives two ints as parameters. Use this to set the value of the caseNumber and amount attributes.
- Initialize isEliminated and isSelected to false
getCaseNumber, setCaseNumber, getAmount, setAmount:
- Standard getter and setter for the two int attributes.
eliminate, isEliminated, select, isSelected:
- Natural language getter/setter variations for the two boolean attributes
- isEliminated and isSelected return the boolean value of the corresponding attribute
- elimiinate and select change the value of the corresponding isEliminated or isSelected attribute to true
toString
- Return a string representation of the Case instance
Banker
-offer : int
+Banker()
+getOffer() : int
+setOffer(int) : void
+makeOffer(Case[]) : boolean
+getCases() : Case[]
+shuffleCases() : void
+getCase(int) : Case
+selectCase(int) : Case
+eliminateCase(int) : Case
+showRemainingCases() : void
+showRemainingValues() : void
+revealCases() : void
1
24
Case
-caseNumber: int
-amount: int
-isElimiinated: boolean
-isSelected: boolean
+Case(int, int)
+getCaseNumber() : int
+setCaseNumber(int) : void
+getAmount() : int
+setAmount(int) : void
+elimiinate(): void
+isEliminated(): boolean
+select():void
+isSelected() : boolean
+toString(): String

DealGame Design



Board methods:

Constructor

- Create the cases array, create the 24 cases in a loop. Use the Case constructor to set the case number and the amount. Save the array in the *cases* attribute.
- Hint: Use the array initializer to create an int array containing all 24 values. Use this temporary array in your loop when creating the 24 Case inistances.

getCases:

- Returns the entire cases array

shuffleCases:

- Use the algorithm you created in Lab 5 to shuffle the cases. In this version, don't relocate the Case objects in the array. Instead, just change the value of the *amount* attribute in the two cases during each swap. Use the getter/setter methods for amount to accomplish this.

getCase:

- Return the Case object for the case number passed as a parameter. Remember to subtract 1 to use as an index number.

selectCase:

- Find the Case in the cases array using the int passed in. Remember that the number passed in (1-24) needs to be decremented by one to use as the index number in the array.
- Once found in the array, send the select message to that Case.

eliminateCase:

- Works like the *selectCase* method, but send the *eliminate* message to the Case instead of *select*.

showRemainingCases:

- As in Lab 5, loop through the cases array and print the cases that have not been selected or eliminated. E.g., [CASE 12]
- For those that are selected or elminated, print an equivalent number of blank spaces so the cases line up neatly.
- After every sixth case (or blanks) is printed, print a CRLF so there are six cases (or blanks) on each of four rows.

showRemainingValues: ***(SEE NOTES ON THE NEXT PAGE)***

DealGame Design

Board methods: (continued)

showRemainingValues:

- Create an int array with room for 24 integers.
- Loop through the cases array and for each Case, retrieve the case dollar amount using *getAmount.*
  - Use the *isElimiinated* method in Case, and copy the Case value to the corresponding element in the new int array IF the case was NOT eliminated.
  - IF the case was eliminated, put a -1 in the int array at that location.
- After populating the int array, sort it using Arrays.sort
- Finally, loop through the int array and print all of the values that are NOT -1.
  - For the -1 values, just skip them, don't print spaces. The printed grid of remaining dollar values should shrink after every turn. (See sample output at the end of the instructions.)

Optional:

revealCases:

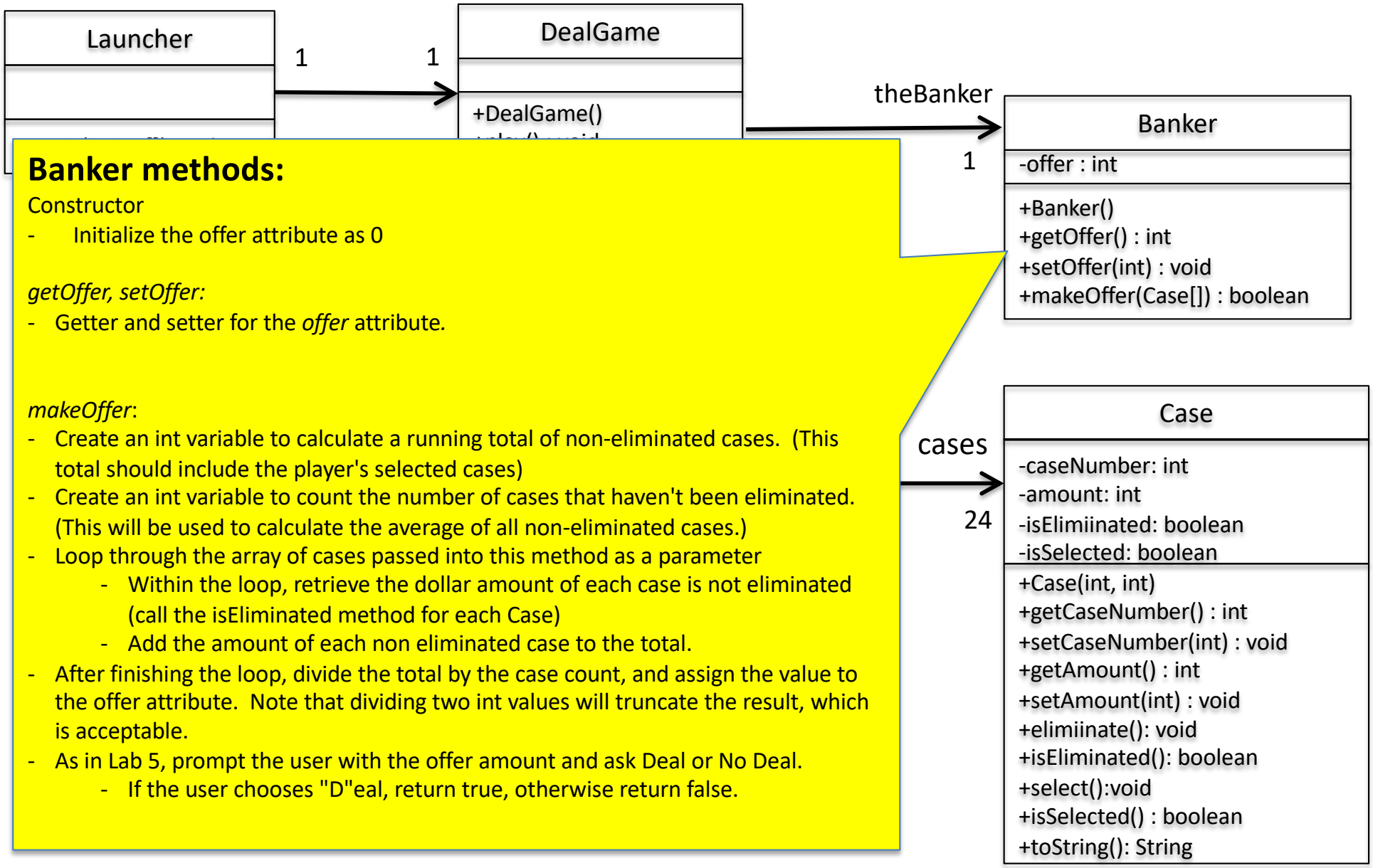
- Create a loop that prints the case number and dollar amount of all the Case instances in the cases array.
- This method is not used in the game, but may be useful for testing while building your solution.

```
//show the contents of all cases
//for testing purposes. not used in the game
public void revealCases()
{
int columnCount = 1;

        for(int i = 0; i < cases.length; i++)
        {
           System.out.printf("%15s ",cases[i]);
           columnCount++;
           if(columnCount == 5)
             {
                System.out.println();
                columnCount = 1;
             }
        }//end if

}//end revealCases
```

DealGame Design



Banker methods:

Constructor

- Initialize the offer attribute as 0

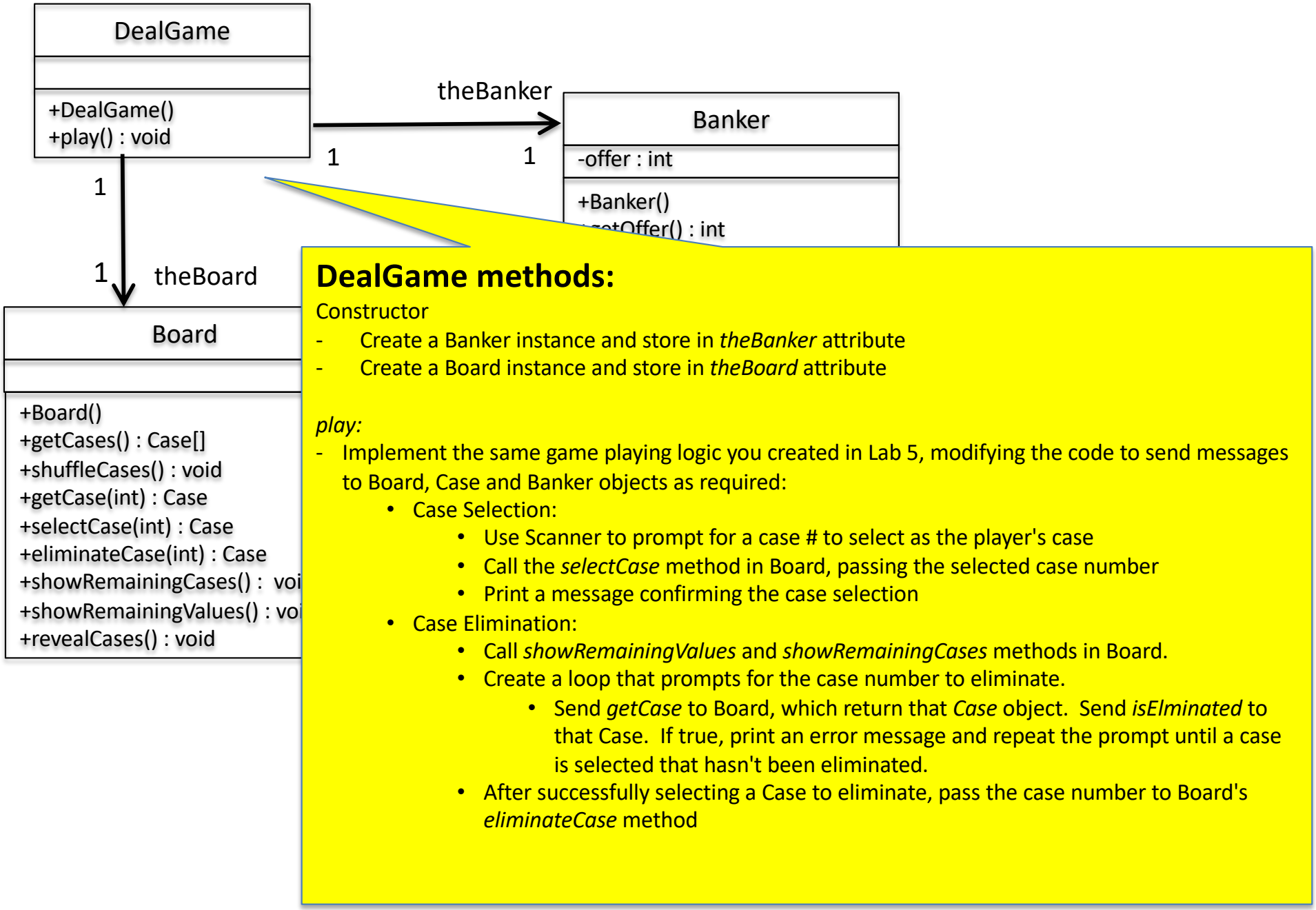
getOffer, setOffer:

- Getter and setter for the *offer* attribute.

makeOffer:

- Create an int variable to calculate a running total of non-eliminated cases. (This total should include the player's selected cases)
- Create an int variable to count the number of cases that haven't been eliminated. (This will be used to calculate the average of all non-eliminated cases.)
- Loop through the array of cases passed into this method as a parameter
    - Within the loop, retrieve the dollar amount of each case is not eliminated (call the isEliminated method for each Case)
    - Add the amount of each non eliminated case to the total.
- After finishing the loop, divide the total by the case count, and assign the value to the offer attribute. Note that dividing two int values will truncate the result, which is acceptable.
- As in Lab 5, prompt the user with the offer amount and ask Deal or No Deal.
    - If the user chooses "D"eal, return true, otherwise return false.

DealGame Design



DealGame methods:

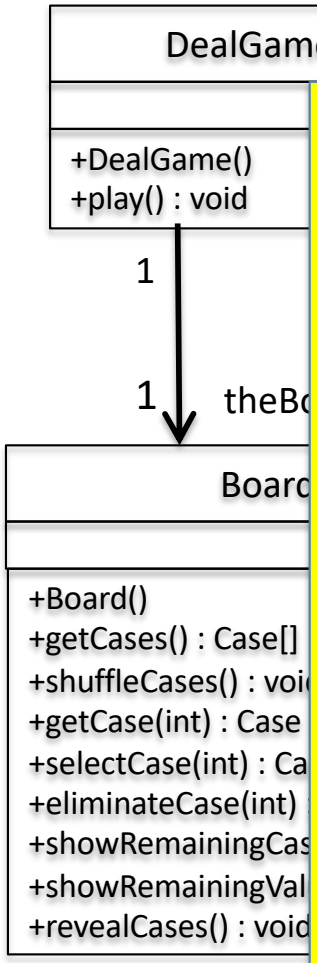
Constructor

- Create a Banker instance and store in *theBanker* attribute
- Create a Board instance and store in *theBoard* attribute

play:

- Implement the same game playing logic you created in Lab 5, modifying the code to send messages to Board, Case and Banker objects as required:
    - Case Selection:
        - Use Scanner to prompt for a case # to select as the player's case
        - Call the *selectCase* method in Board, passing the selected case number
        - Print a message confirming the case selection
    - Case Elimination:
        - Call *showRemainingValues* and *showRemainingCases* methods in Board.
        - Create a loop that prompts for the case number to eliminate.
            - Send *getCase* to Board, which return that *Case* object. Send *isElminated* to that Case. If true, print an error message and repeat the prompt until a case is selected that hasn't been eliminated.
        - After successfully selecting a Case to eliminate, pass the case number to Board's *eliminateCase* method

DealGame Design

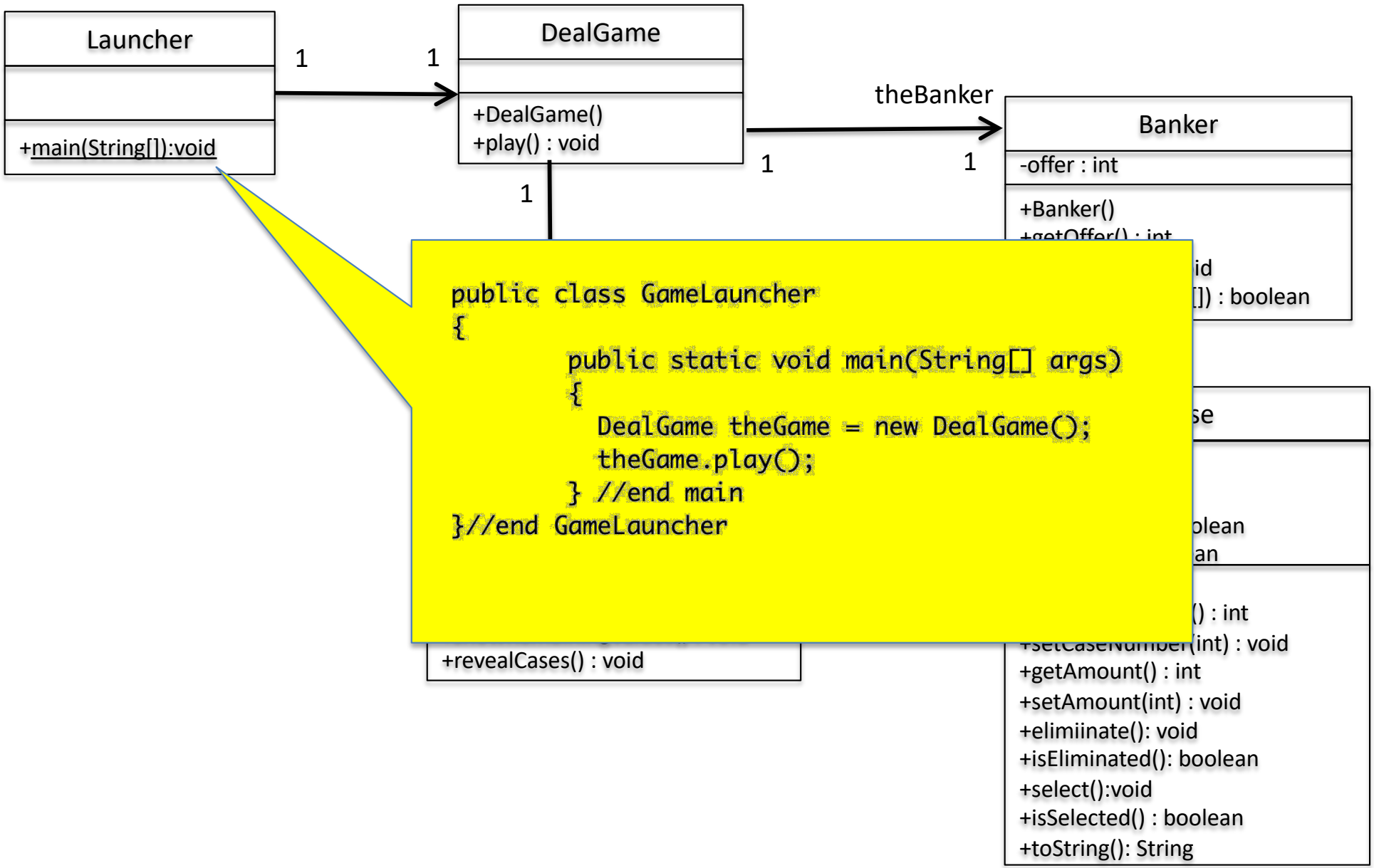


DealGame methods: (continued)

play:

- Implement the same game playing logic you created in Lab 5, modifying the code to send messages to Board, Case and Banker objects as required:
    - Case Selection:
        - Use Scanner to prompt for a case # to select as the player's case
        - Call the *selectCase* method in Board, passing the selected case number
        - Save the Case returned by *selectCase* in a temporary variable to be used later in the *play* method.
        - Print a message confirming the case selection
    - Case Elimination:
        - Call *showRemainingValues* and *showRemainingCases* methods in Board.
        - Create a loop that prompts for the case number to eliminate.
            - Send *getCase* to Board, which return that *Case* object. Send *isElminated* to that Case. If true, print an error message and repeat the prompt until a case is selected that hasn't been eliminated.
        - After successfully selecting a Case to eliminate, pass the case number to Board's *eliminateCase* method, which will return the newly eliminated *Case* object
        - Print a message with the case number and it's amount, invoking methods in *Case* eliiminated in the previous step.
        - Call the *makeOffer* method in *Banker,* passing the *cases* array for the banker to use in creating the offer. The *makeOffer* method will return a boolean.
            - If true is returned by *makeOffer*, print a "Game Over" message. Call *getOffer* in *Banker* to get the offer amount, and use this for the final message which includes calculating whether or not the player made a good deal. (Compare the offer with the amount in the selected case, which you saved in a temporary variable near the beginning of this method.)
                - Break out of the play loop
            - If false is returned by *makeOffer*, play continues within the play loop

DealGame Design



Sample Output:

$ java GameLauncher

```
Welcome to Deal or No Deal!

Please select your case (1-24): 10
You now have case #10 and it's time to eliminate cases.


$         1 $         5 $        10 $        25
$        50 $        75 $       100 $       200
$       300 $       400 $       500 $     1,000
$     5,000 $    10,000 $    25,000 $    50,000
$    75,000 $   100,000 $   200,000 $   300,000
$   400,000 $   500,000 $   750,000 $ 1,000,000


AVAILABLE CASES:
----------------
[CASE  1]  [CASE  2]  [CASE  3]  [CASE  4]  [CASE  5]  [CASE  6]
[CASE  7]  [CASE  8]  [CASE  9]             [CASE 11]  [CASE 12]
[CASE 13]  [CASE 14]  [CASE 15]  [CASE 16]  [CASE 17]  [CASE 18]
[CASE 19]  [CASE 20]  [CASE 21]  [CASE 22]  [CASE 23]  [CASE 24]

Please select a case to eliminate: 1
You eliminated case #1 which contained $ 1,000,000
I am offering you $105,115.   Do you accept? (Y/N) n

$         1 $         5 $        10 $        25
$        50 $        75 $       100 $       200
$       300 $       400 $       500 $     1,000
$     5,000 $    10,000 $    25,000 $    50,000
$    75,000 $   100,000 $   200,000 $   300,000
$   400,000 $   500,000 $   750,000
AVAILABLE CASES:
----------------
           [CASE  2]  [CASE  3]  [CASE  4]  [CASE  5]  [CASE  6]
[CASE  7]  [CASE  8]  [CASE  9]             [CASE 11]  [CASE 12]
[CASE 13]  [CASE 14]  [CASE 15]  [CASE 16]  [CASE 17]  [CASE 18]
[CASE 19]  [CASE 20]  [CASE 21]  [CASE 22]  [CASE 23]  [CASE 24]


Please select a case to eliminate: 1
Case 1 not available for selection
Please select a case to eliminate: 2
You eliminated case #2 which contained $ 25
I am offering you $109,892.   Do you accept? (Y/N) y
Game over!  Your case contained $750,000
Not a great deal!
```