For this assignment, you are going to create a **Deal or No Deal** program. The rules of the game are as follows:

1. At the start of each game, 26 cash amounts ranging from $0.01 to $1,000,000 are each randomly placed inside one of 26 briefcases. Each briefcase displays a number on the outside of the case from 1 to 26.
2. The contestant must select one briefcase to be placed on the contestant's podium.
3. Through a series of rounds, the contestant is asked to select a number of the other cases still in play; each case is opened and the cash value inside the case is revealed before it is taken out of play.
4. After completing the selection of cases for that round, the Banker, will offer a “deal” to the contestant. The deal is a cash offer, the amount of which depends on the values of the cases remaining in play, in exchange for leaving the game.
5. If the player accepts the deal, he or she presses the button to end the game and win the amount of the deal. Otherwise the player declares “No deal!” thus requiring the player to continue into the next round.
6. Each round progressively removes fewer cases from the game; the first round begins with six cases to be removed, the second round with five, diminishing subsequently until the final rounds requiring the removal of one case at a time.
7. Should the player refuse the final Banker’s offer, with the selected case and one other case left in play, the player is given the opportunity to swap cases, values unseen, and win whatever case they end up keeping at that point.

When the program starts, the game board should appear and should look something like this:
The first thing the user will need to do is select one briefcase which the user keeps until the end of the game. Once the user selects the briefcase, it should appear at the bottom-left hand corner.
Next the user will need to select six briefcases from the board. As each case is selected, the cash value inside the case must be revealed and taken off the board.

After completing the selection of cases for that round, the Banker, will offer a “deal” to the contestant. The deal is a cash offer, the amount of which depends on the values of the cases remaining in play, in exchange for leaving the game. The formula you will use to calculate the banker’s offer is as follows:

\[
\text{Banker’s offer} = \frac{\text{Average value of the money remaining} \times \text{round number}}{10}
\]

The round number goes up by one each time the banker makes an offer. So when the user makes an offer, the offer should appear in the form of an option dialog:

To create an option dialog with your own custom options (i.e. DEAL and NO DEAL) you would first need to create an array of options:

```java
Object[] options = {"DEAL", "NO DEAL"};
```

You would then need to create an option dialog that uses the above options:
JOptionPane.showMessageDialog(null, "The banker's offer is " + df2.format(offer) + "!
Deal or no deal?", "Banker's offer", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, options, options[0]);

Keep in mind that an option dialog returns an int representing the option the user selects. In this case, 0 if they select DEAL or 1 if they select NO DEAL.

If the contestant selects DEAL, the user goes home with the banker’s offer.

You must also reveal to the player how much they would’ve gone home if they stayed with their own suitcase before the program exits:

If the user selects NO DEAL, the game resumes with the contestant selecting more suitcases.

The 26 briefcases are opened according to the following system:

ROUND 1: 6 cases are opened. The Bank makes an offer based on the remaining 20 closed cases.
ROUND 2: 5 cases are opened. The Bank makes an offer based on the remaining 15 closed cases.
ROUND 3: 4 cases are opened. The Bank makes an offer based on the remaining 11 closed cases.
ROUND 4: 3 cases are opened. The Bank makes an offer based on the remaining 8 closed cases.
ROUND 5: 2 cases are opened. The Bank makes an offer based on the remaining 6 closed cases.
ROUND 6-9: The remaining unopened briefcases are then opened one at a time. In this portion of the game, one bank offer is made to the contestant after each individual case is opened.

When there is only one suitcase left on the board, users have the option to give up their case for the last unopened case. If users choose to keep their original case, they win the amount in that case. If, on the other hand, users choose to swap cases, they go home with the amount in the remaining case.

The case the user selects in the end should appear in a message dialog together with the amount that was in the case:

![Screen shot of the interface showing the option to keep or swap the case](image)

When the game ends, the user should have the option to play again. If the user chooses to play again, the game should restart and random numbers should be reassigned to the suitcases.

![Screen shot of the interface asking the player if they would like to play again](image)

If the user chooses not to play again, the program should just exit.

Save the project as **Deal or No Deal** in your UNIT 2 folder.
## PROGRAMMING ASSIGNMENT:
**Deal or No Deal**

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>CRITERIA</th>
<th>&lt; LEVEL 1 0 – 49%</th>
<th>LEVEL 1 50 – 59%</th>
<th>LEVEL 2 60 – 69%</th>
<th>LEVEL 3 70 – 79%</th>
<th>LEVEL 4 80 – 100%</th>
<th>MARK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Knowledge and Understanding</strong></td>
<td>Demonstrates an understanding of how to create a program using two-dimensional arrays</td>
<td>0-4.9</td>
<td>5.0-5.9</td>
<td>6.0-6.9</td>
<td>7.0-7.9</td>
<td>8.0-10</td>
<td></td>
</tr>
<tr>
<td>Thinking</td>
<td>The program meets all the specifications required</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Validates program to ensure the program produces correct results</td>
<td>0-4.9</td>
<td>5.0-5.9</td>
<td>6.0-6.9</td>
<td>7.0-7.9</td>
<td>8.0-10</td>
<td></td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>Provides internal documentation that clearly explains the methods and the program logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Documents program logic with no success</td>
<td>0-4.9</td>
<td>5.0-5.9</td>
<td>6.0-6.9</td>
<td>7.0-7.9</td>
<td>8.0-10</td>
<td></td>
</tr>
<tr>
<td><strong>Application</strong></td>
<td>Effectively applies knowledge and understanding of how to create algorithms to process elements in two-dimensional arrays</td>
<td>0-4.9</td>
<td>5.0-5.9</td>
<td>6.0-6.9</td>
<td>7.0-7.9</td>
<td>8.0-10</td>
<td></td>
</tr>
</tbody>
</table>

### OVERALL CURRICULUM EXPECTATIONS THAT ARE COVERED IN THIS ASSIGNMENT:

- **A1.** Demonstrate the ability to use different data types and expressions when creating computer programs.
- **A2.** Describe and use modular programming concepts and principles in the creation of computer programs.
- **A3.** Design and write algorithms and subprograms to solve a variety of problems.
- **A4.** Use proper code maintenance techniques when creating computer programs.
- **B1.** Demonstrate the ability to manage the software development process effectively, through all of its stages – planning, development, production, and closing.
- **C1.** Demonstrate the ability to apply modular design concepts in computer programs.
CATHOLIC GRADUATE EXPECTATIONS THAT ARE COVERED IN THIS ASSIGNMENT:

- Presents information and ideas clearly and honestly and with sensitivity to others.
- Uses and integrates the Catholic faith tradition, in the critical analysis of the arts, media, technology and information systems to enhance the quality of life.
- Creates, adapts, and evaluates new ideas in light of the common good.
- Demonstrates flexibility and adaptability.
- Thinks reflectively and creatively to evaluate situations and solve problems.
- Sets appropriate goals and priorities in school, work and personal life.
- Achieves excellence, originality, and integrity in one's own work and supports these qualities in the work of others.