**Assignment 3**

**Maximum Marks: 64 Weight: 25%**

**------------------------------------------------------------------------------------------------------------------------------------------**

This is a group assignment. **You MUST work in groups of two to complete this assignment**. If you can’t find a partner, please email me and I will pair you up with another loner.

Your submitted work must be "ENTIRELY" your work. Make sure you understand the academic policies and procedures as detailed at:

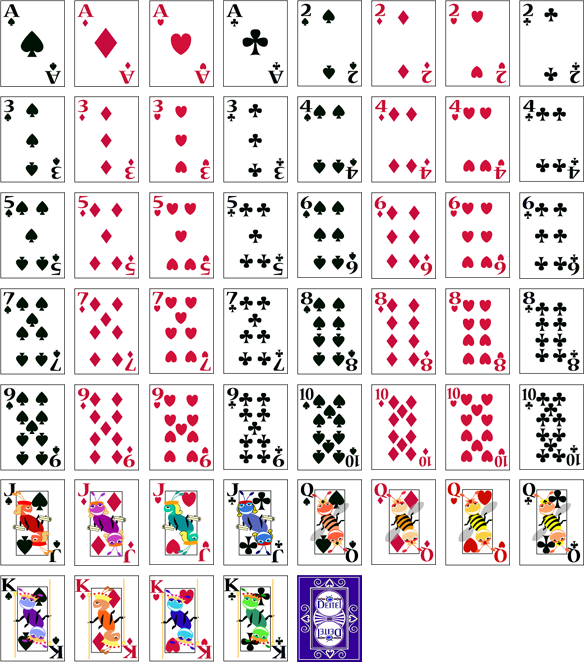
<http://www.georgianc.on.ca/admissions/policies-procedures/academic-misconduct>

Pay special attention to warnings about cheating, plagiarism, and acceptable use of college computer resources.

If your submitted work is found out to have been re-produced from any human or non-human resource (for example, the web), I will strictly be following the procedure as in the above document.

**How to submit:** Please use the Eclipse IDE to create this game application. I will be using the same to mark it. Please do not use any plug-in to create the interface. It should be coded by you. Please zip all the files and submit via the Assignment 3 link on blackboard. Please name the zipped file *yournames\_*Assignment3.zip replacing *yournames* with the names of the two group members. Only one submission is required per group.

**Overview**: Create a Memory Game using the **Java** language. This game tests the user's memory. To keep it manageable, we will play with 16 cards instead of a pack of 52. Include a four-by-four grid for 16 playing cards that are face down (e.g. showing the back of the card). When the user touches the back of a playing card, a card is revealed. The user then touches the back of another playing card trying to find a match with the same card. If the two cards do not match, the cards are flipped back to the back side. If the two cards match, the user gets a point and the cards are removed from the game/screen. The Game should keep track of the user’s score. The game gives the user 60 seconds to play and accumulate the score. When the timer runs out the user can reset the game. The playing card images for this game are included in the file Card Images.zip.



**Instructions :**

Please write efficient and non-redundant code. Please keep the design code separate from the event handling code in the GUI class code. Use inner classes to handle the events. Further the code to ‘play’ the game must be in a separate code file. (5 Marks)

Create a GUI that includes the following components **(11 Marks: GUI, 31 Marks: Functionality):**

* 1. Sixteen images of **card backs** on **Buttons** are initially displayed on the screen in a **four by four grid** (4 rows and 4 columns) (4 Marks: GUI).
  2. A message **TextField** at the top of the screen will display a message to the user: “Pick a card, any card” (1 Marks: GUI, 1 Marks: Functionality).
  3. A score **TextField** at the top of the screen displays the users score (initially set to zero) (1 Marks: GUI, 2 Marks: Functionality).
  4. A timer **TextField** at the top of the screen displays the countdown (initially set to 60 seconds) (1 Marks: GUI, 3 Marks: Functionality).
  5. A “Play Again” **button** is displayed that allows the user to reset the game any time. Disable it initially and enable it as soon as the user starts to play. (1 Marks: GUI, 2 Marks: Functionality).
  6. A “Quit” **button**  is displayed that allows the user to quit the game at any time. (1 Marks: GUI, 1 Marks: Functionality).
  7. Eight cards from the 52 card deck will be **randomly selected** and two images of each card will be placed at **random locations** on the grid and only revealed when the user clicks on a **card back** (2 Marks: GUI, 6 Marks: Functionality).
  8. User clicks on a card back to select it. It is flipped over and revealed. (2 Marks: Functionality).
  9. When the user has selected his **first card** the message **TextField** at the top of the screen will change its message to: “Now pick another card” (1 Marks: Functionality)
  10. The user then makes a **second selection**. If the second card selected matches the first card selected, the score is **updated by 1 point**, the message **TextField** at the top of the screen displays: “Right!” and then both cards become **invisible**. If the card is not the same as the first, the message **TextField** displays “Wrong! Pick again…” and the second card the user chose flips over again (8 Marks: Functionality).
  11. The user can “De-select” his original card choice if he presses it a second time (1 Marks: Functionality).
  12. The user continues making choices until the timer runs out or the game has been won, that is all cards have been matched. The user’s final score is displayed in a message box. (4 Marks: Functionality).

1. Include **Internal Documentation** for your program **(17 Marks: Internal Documentation):**
   1. Please follow naming conventions for classes, methods, variables and constants.(1 Mark: Internal Documentation)
   2. Ensure you include **comment** **headers** in your **java** **file(s)** that indicate: the **Author names**, **Date** **and Description of the class** (4 Marks: Internal Documentation).
   3. Ensure you include a **comment header** for all of your **methods** that describe the purpose of the method (5 Marks: Internal Documentation)
   4. Ensure your program uses contextual variable names that help make the program human-readable (2 Marks: Internal Documentation).
   5. Ensure you include inline comments that describe your GUI Design and Functionality (5 Marks: Internal Documentation)

**Optional Features (i.e. Potential Bonus Marks).**

1. Add sound clips that provide feedback to the user when he selects a card, achieves a point, matches the final few seconds of the countdown, initiates a buzzing sound when the timer runs out, etc.
2. Have the game keep track of a high score and the initials of the player.
3. Allow the game to continue if the user completes the first 8 card matches by resetting the grid with 16 new hidden cards. Add an additional 10 seconds to his timer to allow him to collect more points. Continue to allow the user to match more cards until his time runs out.