CS 351: Design of Large Programs
Project 2: Domino Game

Brooke Chenoweth

Spring 2025

Game Description

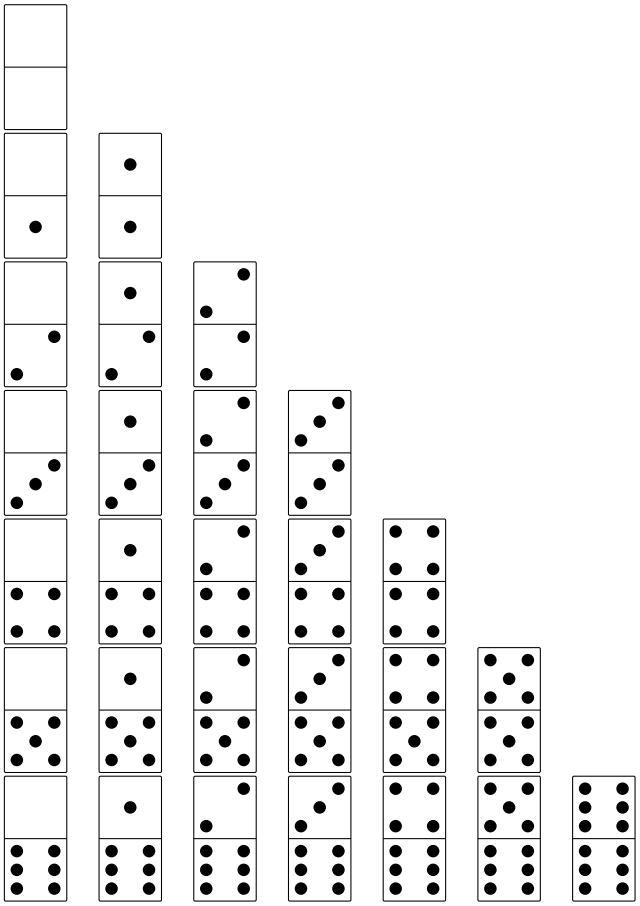
The game of dominos has its origins in China and made its way into Europe sometime in the 18th century. There are many versions of the game and some use different number of pieces. A domino, the piece being played, has two numbers separated by a line. The most common version of the game uses 28 pieces. The rules of the game also vary a great deal. The version outlined below assumes two players:

- The game starts with all the dominos being placed face down on the table and mixed around. This is called the boneyard.
- Each player selects 7 dominos and places them on the long edge so as to be able to see them without being visible to the opponent.
- One player starts the game by downing (putting down) a domino face-up on the table.
- The players take turns.
- On each turn, the player downs a single domino matching the configuration on the table.
- The dominos will form two parallel rows shifted by half a domino.
- Two adjacent dominos located on different rows must have the same value in the overlapping halves.
- A blank is used as a wild card and thus matches any value next to it.
- If the player has a domino that can extend the line of play in either direction, they put it down and their turn ends.
- If the player does not have a domino able to extend the line of play, they must pick one up from the boneyard and must continue to do so until a matching domino is found or the boneyard is empty.

- If the player has a domino able to extend the line of play, they cannot pick any pieces from the boneyard.
- If the boneyard is empty and the player cannot extend the line of play, they end their turn.
- The game ends when the boneyard is empty and either
  - The current player places their last domino.
  - Both players have taken a turn without placing a domino.
- At the end of the game, the dots on each player's dominos are counted and the player with the lower total wins. (This means that is one player ends the game without any dominos, they're pretty much guaranteed to be the winner.)

  If both players have the same number of dots, the player who last played a domino is the winner.

Tiles Used in Play



Assignment

- Write at least two versions of a Java program, each implementing the Domino game as described above.
- One player should be the computer with the other player being the user.
- The user always makes the first move.
- The designs must be object oriented, highly modular, and amenable to change.
- Provide a design diagram for each version of the program.

  If two versions use essentially the same object design, you may choose to have a single diagram and just explain the differences in your description.

  Your final object design diagram must be digitally created.[1]
- Each solution must be sequential, i.e., it cannot use multithreading.
- Each new version should build on the previous one.
- The code must be built incrementally.

Required Versions

You must submit jars and object oriented design documents for at least the following two versions of your program, but you are welcome to include more for intermediate stages. Make sure you document the versions in your readme so we'll know how to play your game.

Build the code incrementally, always having a program that delivers a subset of the overall game functionality. Keep changes to the previous version's code to the minimum. You should not be completely rewriting your game logic when changing from console to GUI, for example.

- Version 1: Console version

  Use only standard input/output to communicate with the user, i.e., no graphics.

  The current turn in the console game must be viewable without scrolling up on a standard terminal.
- Version 2: JavaFX GUI version

  Add graphics to display the state of the game and configure the graphical interface so as to allow the user to interact directly with the game display.

  The GUI game must be playable using the mouse only. Additional keyboard options are okay, but the keyboard should not be required.[2]

  The GUI must display dots on the graphical dominos in the the playing area and the human's hand.

[1] We've had too many problems with illegible handwritten documents and poor quality photos.

[2] It doesn't really count as a GUI game if you just paste your console game into a text field.

Examples

The following examples are from my version of dominos to give you more of an idea of what I'm expecting visually, but you don't have to match my interface exactly.

You should visually stagger the dominos in both games, but I'd expect you to store them internally in some sort of linear structure. Remember, the visual representation and the logical representation of your data are not always the same.

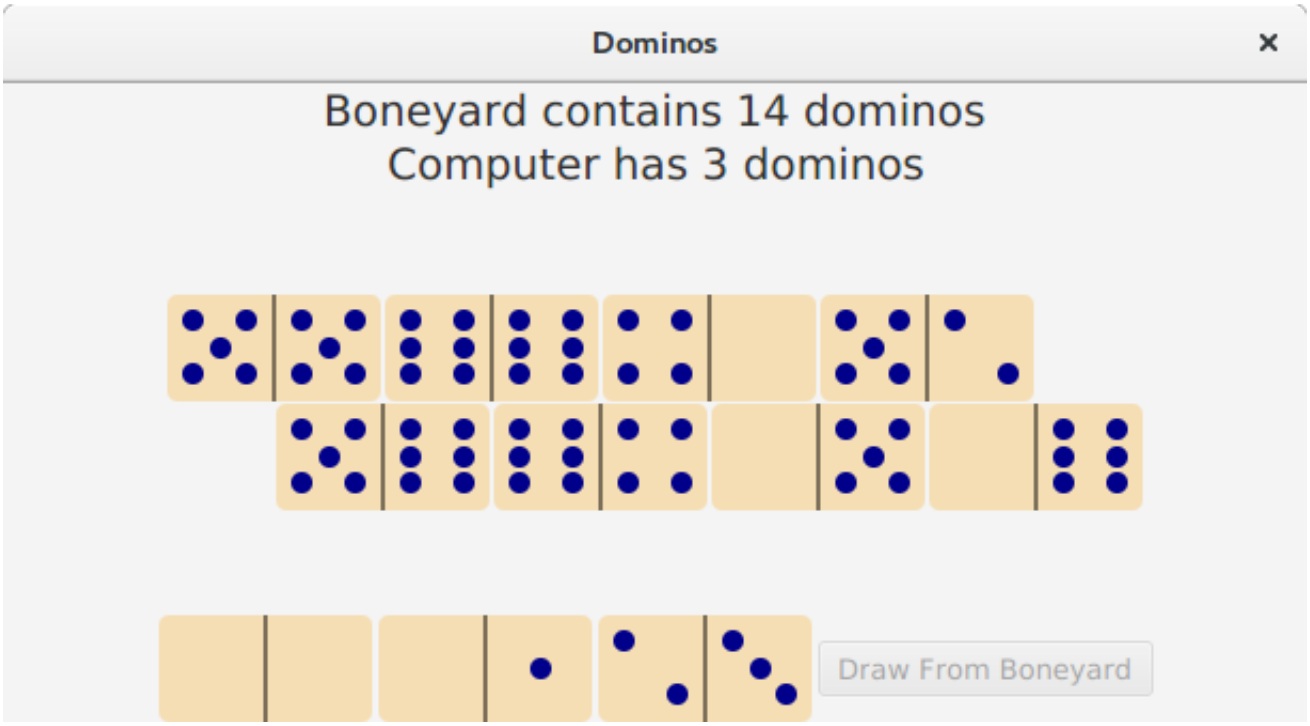
Console Game

```
Dominos!
Computer has 7 dominos
Boneyard contains 14 dominos


Tray: [[4  5], [0  2], [2  6], [3  6], [2  2], [1  5], [5  6]]
Human's turn
[p] Play Domino
[d] Draw from boneyard
[q] Quit
p
Which domino?
6
Left or Right? (l/r)
l
Rotate first? (y/n)
n
Playing [5  6] at left
Computer has 7 dominos
Boneyard contains 14 dominos
[5  6]

Tray: [[4  5], [0  2], [2  6], [3  6], [2  2], [1  5]]
Computer's turn
Computer plays [2  5] at left
Computer has 6 dominos
Boneyard contains 14 dominos
   [5  6]
[2  5]
Tray: [[4  5], [0  2], [2  6], [3  6], [2  2], [1  5]]
Human's turn
[p] Play Domino
[d] Draw from boneyard
[q] Quit
p
```

```
Which domino?
2
Left or Right? (l/r)
l
Rotate first? (y/n)
y
Playing [6  2] at left
Computer has 6 dominos
Boneyard contains 14 dominos
[6  2][5  6]
   [2  5]
Tray: [[4  5], [0  2], [3  6], [2  2], [1  5]]
Computer's turn
Computer plays [0  4] at right
Computer has 5 dominos
Boneyard contains 14 dominos
[6  2][5  6]
   [2  5][0  4]
Tray: [[4  5], [0  2], [3  6], [2  2], [1  5]]
Human's turn
[p] Play Domino
[d] Draw from boneyard
[q] Quit
p
Which domino?
0
Left or Right? (l/r)
r
Rotate first? (y/n)
n
Playing [4  5] at right
Computer has 5 dominos
Boneyard contains 14 dominos
[6  2][5  6][4  5]
   [2  5][0  4]
Tray: [[0  2], [3  6], [2  2], [1  5]]
Computer's turn
Computer plays [0  0] at right
Computer has 4 dominos
Boneyard contains 14 dominos
[6  2][5  6][4  5]
   [2  5][0  4][0  0]
Tray: [[0  2], [3  6], [2  2], [1  5]]
Human's turn
[p] Play Domino
```

```
[d] Draw from boneyard
[q] Quit
p
Which domino?
1
Left or Right? (l/r)
r
Rotate first? (y/n)
n
Playing [3  6] at right
Computer has 4 dominos
Boneyard contains 14 dominos
[6  2][5  6][4  5][3  6]
   [2  5][0  4][0  0]
Tray: [[0  2], [2  2], [1  5]]
Computer's turn
Computer plays [0  1] at right
Computer has 3 dominos
Boneyard contains 14 dominos
[6  2][5  6][4  5][3  6]
   [2  5][0  4][0  0][0  1]
Tray: [[0  2], [2  2], [1  5]]
Human's turn
[p] Play Domino
[d] Draw from boneyard
[q] Quit
```

GUI Game



New Requirement: Command Line Argument for Size

Both the console and GUI programs should support an optional integer command line argument giving the size of the domino set used in the game.

- If no command line argument is given or if an argument of 6 is given, the game will proceed as originally specified using dominos with up to 6 dots.
- If the command line argument is anywhere from 3 to 9, the domino set will be generated with that maximum number of dots.
- You may choose to support values smaller than 3 (though it wouldn't be a very interesting game) or larger than 9, but you may choose to reject values outside that range with an error message. Document your choice in your readme file.
- Both versions of the game must still support the staggered playing area display. (You might not want to support more than 9 dots per side just to make the string representation easier to align in the console version.)
- The GUI version must be able to display dominos with dots for all allowed sizes. (You probably want to have a maximum size to be able to specify the dot layout.)