CSSE 220 Day 6

Arrays, ArrayLists,
Wrapper Classes, Auto-boxing,
Enhanced *for* loop

Check out *ArraysAndLists* and *TwoDArrays* from SVN

Questions?

Exam 1 is Wednesday March27!

- Over chapters 1-7
- You'll have a chance to ask questions about anything in next Monday's class.
- See Session 10 on the Schedule Page schedule for **Exam 1 samples**

Part 1 – Written. You may bring an 8.5 x 11 inch sheet of paper (double-sided, hand-written or printed) with whatever you want on it.
Part 2 – Computer. Code that you must write and debug. You can use your textbook, the Java API documents, and any programs that you have written or we have given you.

Q1

So, what's the deal with primitive types?

- Problem:
  - ArrayList's only hold objects
  - Primitive types aren't objects
- Solution:
  - *Wrapper classes*—instances are used to "turn" primitive types into objects
  - Primitive value is stored in a field inside the object

Primitive	Wrapper
byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

Q2

Auto-boxing Makes Wrappers Easy

- Auto-boxing: automatically enclosing a primitive type in a wrapper object when needed
- Example:
  - You write: `Integer m = 6;`
  - Java does: `Integer m = new Integer(6);`

  - You write: `Integer answer = m * 7;`
  - Java does:

```
int temp = m.intValue() * 7;
Integer answer = new Integer(temp);
```

Auto-boxing Lets Us Use ArrayLists with Primitive Types

- Just have to remember to use wrapper class for list element type

- Example:

  -
    ```
    ArrayList<Integer> runs =
              new ArrayList<Integer>();
    runs.add(9); // 9 is auto-boxed
    ```

  -
    ```
    int r = runs.get(0); // result is unboxed
    ```

Enhanced For Loop and Arrays

- Old school

```
double scores[] = …
double sum = 0.0;
for (int i=0; i < scores.length; i++) {
    sum += scores[i];
}
```

- New, whiz-bang, enhanced for loop

```
double scores[] = …
double sum = 0.0;
for (double score : scores) {
    sum += score;
}
```

Say "in"

- No index variable (easy, **but limited in 2 respects)**
- Gives a name (**score** here) to each element

Enhanced For and ArrayList's

```
ArrayList<State> states = …
int total = 0;
for (State state : states) {
    total += state.getElectoralVotes();
}
```

Q3

Two-dimensional arrays

```java
public class TicTacToe {
    private final int rows;
    private final int columns;
    private String[][] board;

    /**
     * Constructs a 3x3 TicTacToe board with all squares blank.
     */
    public TicTacToe() {
        this.rows = 3;
        this.columns = 3;

        this.board = new String[this.rows][this.columns];

        for (int r = 0; r < this.rows; r++) {
            for (int c = 0; c < this.columns; c++) {
                this.board[r][c] = " ";
            }
        }
    }
```

What is the value of `this.board[1][2]` immediately after this statement executes?

Could have used: `this.board.length`

Could have used: `this.board[r].length`

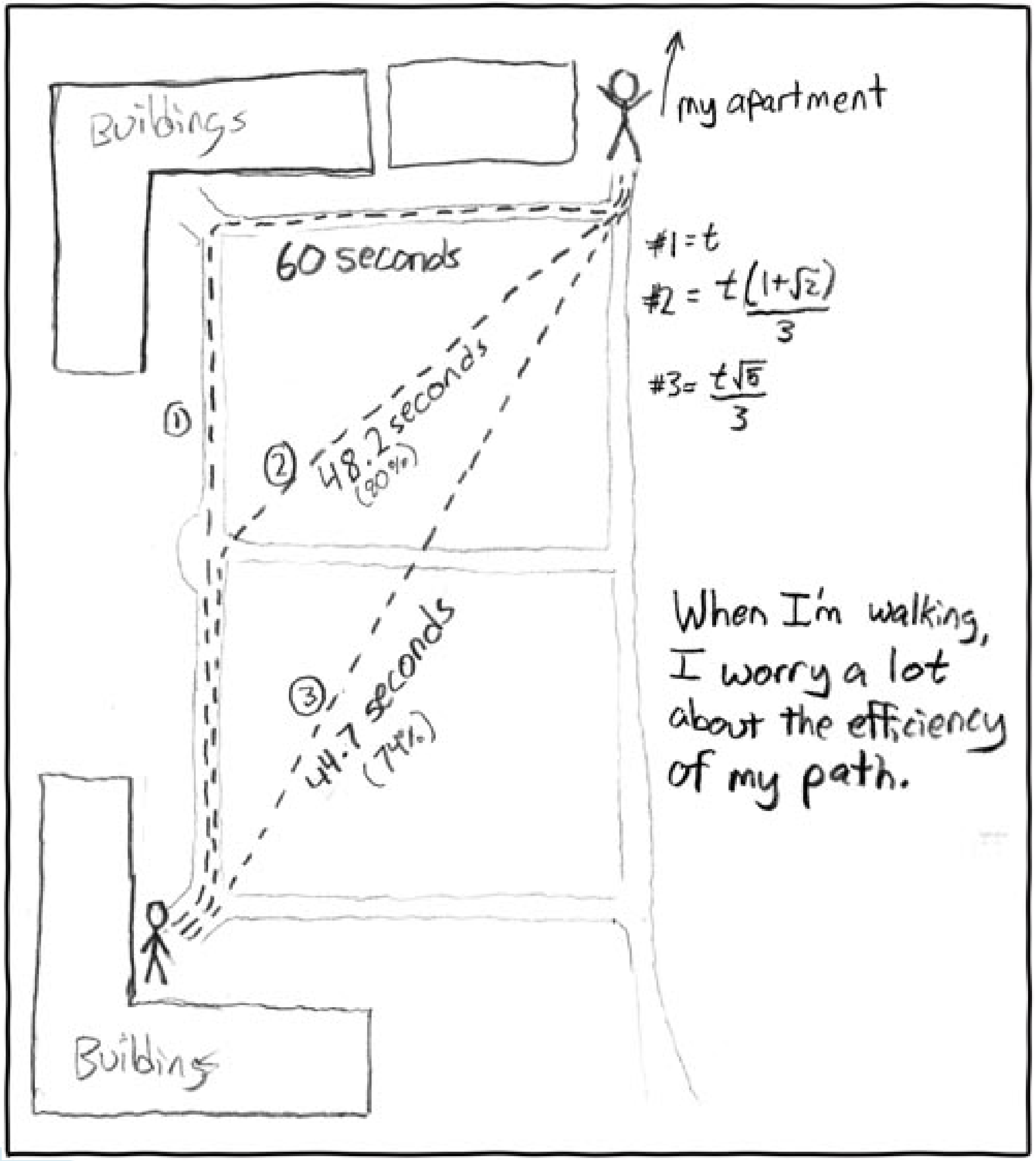
Note the (very common) pattern: loop-through-rows, for each row loop-through columns

Q4

Exercise

- Complete the TODO items in TicTacToe and TicTacToeTest
  They're numbered; do 'em in order.

Interlude:



Buildings
my apartment
60 seconds
#1 = t
$\#2 = \frac{t(1+\sqrt{2})}{3}$
$\#3 = \frac{t\sqrt{5}}{3}$
①
② 48.2 seconds (90%)
③ 44.7 seconds (74%)
When I'm walking, I worry a lot about the efficiency of my path.
Building

http://xkcd.com/85/

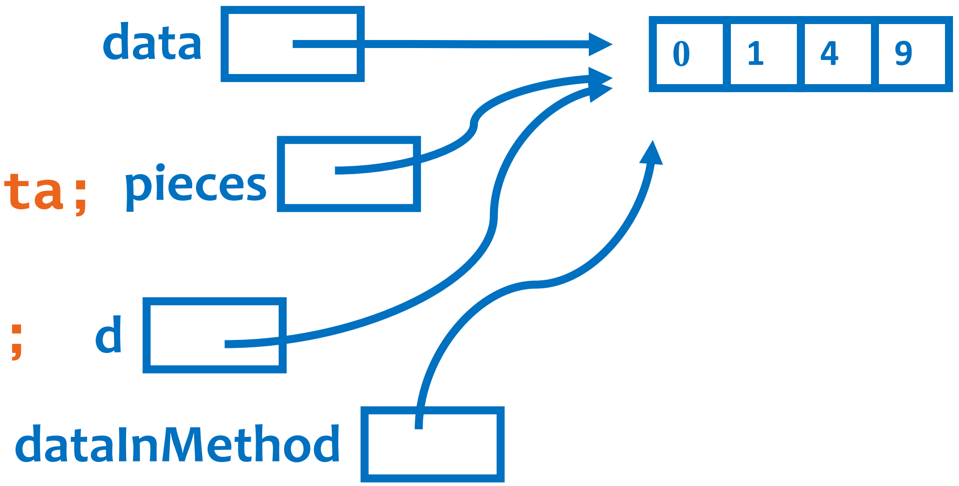
Copying Arrays – assignment

- Assignment uses *reference* values:

  -
    ```
    double[] data = new double[4];
    for (int i = 0; i < data.length; i++) {
        data[i] = i * i;
    }
    ```

  -
    ```
    double[] pieces = data;
    ```

  -
    ```
    foo.someMethod(data);
    ```



This makes the field a reference to (NOT a copy of) a list that exists elsewhere in the code. Think carefully about whether you want this or a clone (copy).

```
public void someMethod(double[] d) {
    this.dataInMethod = d;
    ...
}
```

Q5-6

Copying Arrays – many ways

- You can copy an array in any of several ways:
  1. Write an explicit loop, copying the elements one by one
  2. Use the *clone* method that all arrays have

     ```
     newArray = oldArray.clone();
     ```
  3. Use the *System.arraycopy* method:

     Starting position in *oldArray*

     Starting position in *newArray*

     ```
     System.arraycopy(oldArray, 0, newArray, 0,
                                       oldArray.length);
     ```
  4. Use the *Arrays.copyOf* method:

     Number of elements to copy

     ```
     newArray = Arrays.copyOf(
                     oldArray, oldArray.length);
     ```

The key point is that all of these except possibly the first make ***shallow copies*** – see next slide

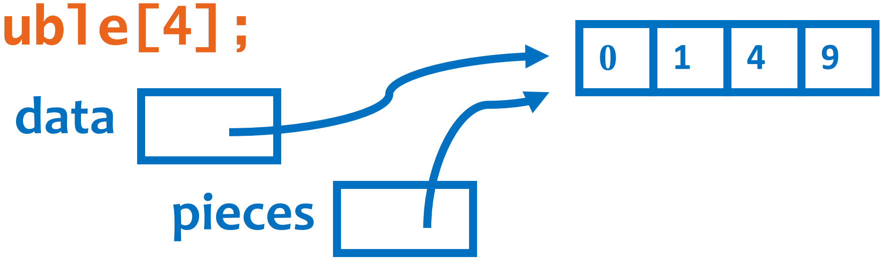
Copying Arrays – Shallow copies

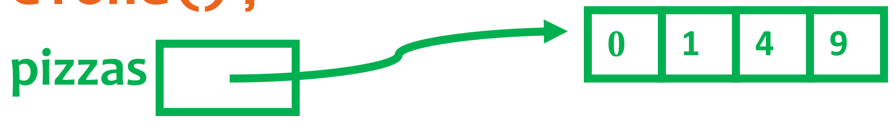
- Can copy whole arrays in several ways:

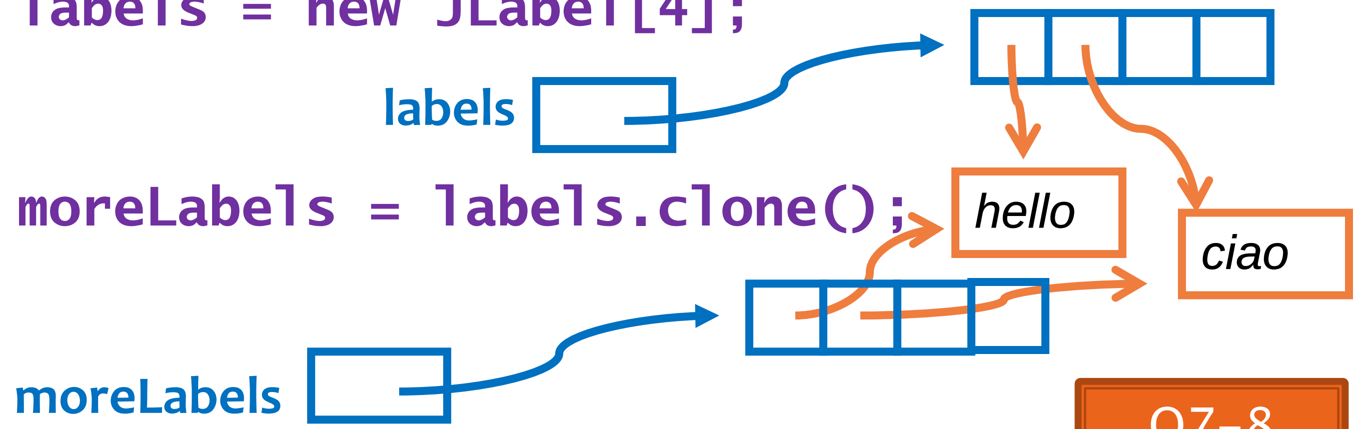
  -
    ```
    double[] data = new double[4];
        ...
    pieces = data;
    ```

  -
    ```
    double[] pizzas = data.clone();
    ```

  -
    ```
    JLabel[] labels = new JLabel[4];
        ...
    JLabel[] moreLabels = labels.clone();
    ```







Q7-8

Quality Tip –"Avoid parallel arrays"

- Consider an ElectionSimulator:
  - Instead of storing:

    -
      ```
      ArrayList<String> stateNames;
      ArrayList<Integer> electoralVotes;
      ArrayList<Double> percentOfVotersWhoPlanToVoteForA;
      ArrayList<Double> percentOfVotersWhoPlanToVoteForB;
      ```
  - We used:
    - `ArrayList<State> states;`
      and put the 4 pieces of data inside a State object
- Why bother?

Q9

Pick the Right Data Structure

- Array or ArrayList, that is the question

- General rule: use ArrayList
  - Less error-prone because it grows as needed
  - More powerful because it has methods

- Exceptions:
  - Lots of primitive data in time-critical code
  - Two (or more) dimensional arrays

Q10

Software Engineering Techniques

- Regression testing
- Pair programming
- Team version control

Regression Testing

- Keep and run old test cases
- Create test cases for new bugs
  - Like antibodies, to keep a bug from coming back
- Remember:
  - You can right-click the project in Eclipse to run all the unit tests

Q11–12

Pair Programming Video

- Let's watch the video together

Pair Programming

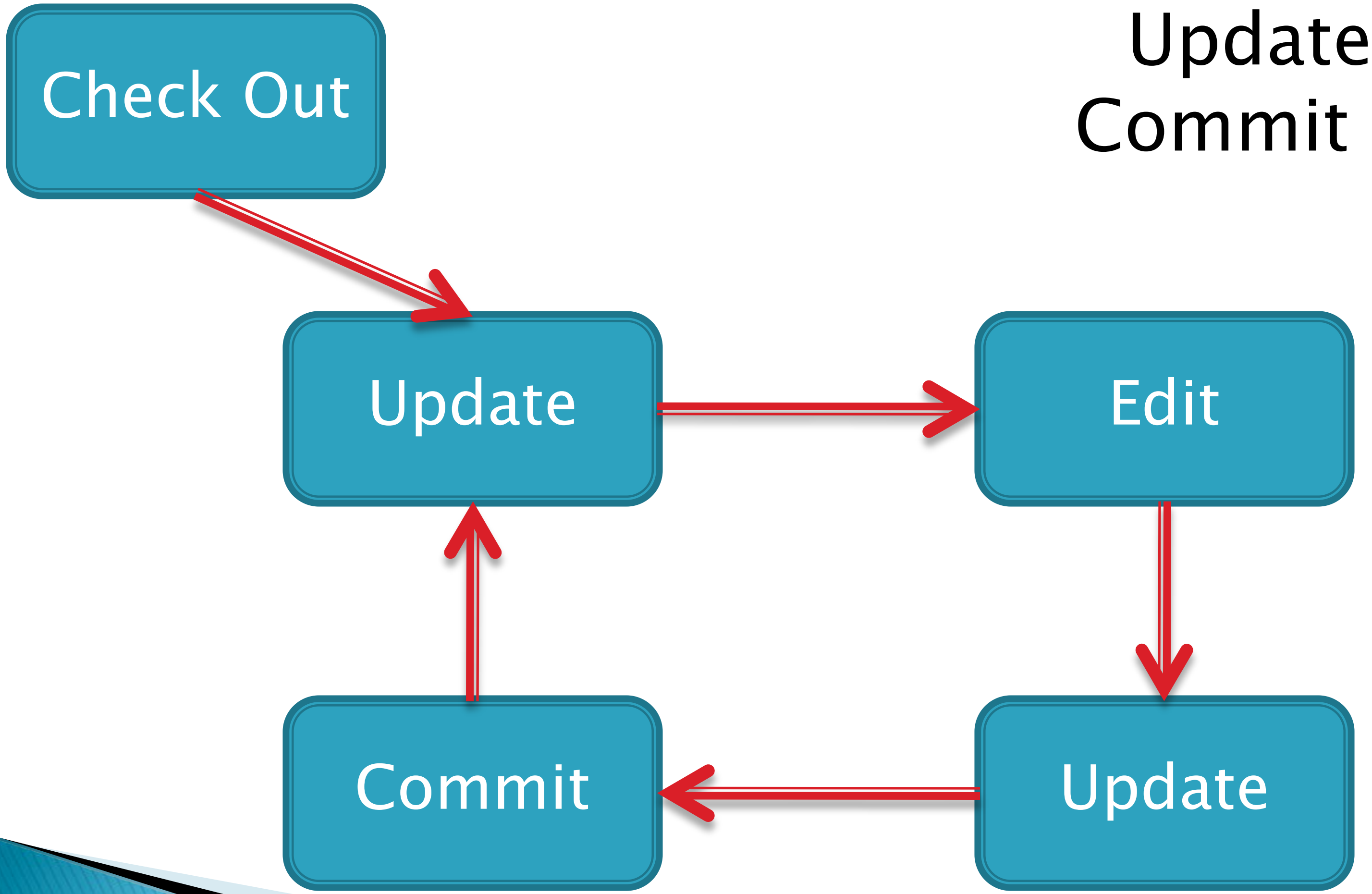
Becoming a common interview technique!

- Working in pairs on a single computer
  - One person, the *driver*, uses the keyboard
  - The other person, the *navigator*, watches, thinks, and takes notes
- For hard (or new) problems, this technique
  - Reduces number of errors
  - Saves time in the long run
- Works best when partners have similar skill level
  - If not, then student with most experience should navigate, while the other student drives.

Team Version Control

- **Always**:
  - **Update before** working
  - **Update again** before committing
  - **Commit often** and with good messages
- **Communicate** with teammates so you don't edit the same code simultaneously
  - Pair programming eliminates this issue

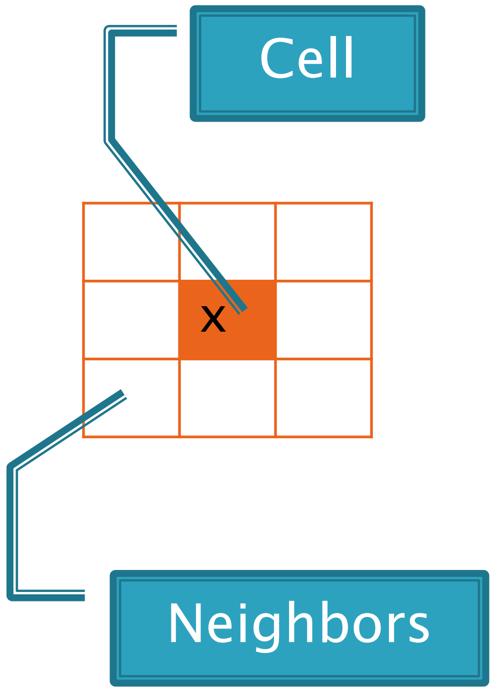
Team Version Control



Update and Commit often!

Game of Life

1. A new cell is born on an empty square if it has exactly 3 neighbor cells
2. A cell dies of overcrowding if it is surrounded by 4 or more neighbor cells
3. A cells dies of loneliness if it has just 0 or 1 neighbor cells



Developed by John Conway, 1970

Team Repositories

- ◦ http://svn.csse.rose-hulman.edu/repos/csse220-201330-teamXX

Game of Life Teams Section 1

Format: repositoryName,firstStudent,secondStudent

csse220-201330-team01,benshorm,woodjl
csse220-201330-team02,brynelnm,mcnelljd
csse220-201330-team03,daruwakj,shumatdp
csse220-201330-team04,gauvrepd,kadelatj
csse220-201330-team05,gouldsa,tebbeam
csse220-201330-team06,griffibp,heathpr
csse220-201330-team07,hazzargm,songh1
csse220-201330-team08,holzmajj,roccoma
csse220-201330-team09,litwinsh,plugerar
csse220-201330-team10,malikjp,olivernp

Check out *GameOfLife* from SVN

Game of Life Teams Section 2

Format: repositoryName,firstStudent,secondStudent

- csse220-201330-team11,adamoam,alayonkj
- csse220-201330-team12,bochnoej,wrightj3
- csse220-201330-team13,calhouaj,cheungnj
- csse220-201330-team14,evansc,wagnercj
- csse220-201330-team15,haloskzd,stephaje
- csse220-201330-team16,hullzr,phillics
- csse220-201330-team17,johnsoaa,kethirs
- csse220-201330-team18,johnsotb,tatejl
- csse220-201330-team19,liuj1,zhoup
- csse220-201330-team20,matsusmk,vanakema
- csse220-201330-team21,mookher,morrisrg
- csse220-201330-team22,naylorbl,winterc1
- csse220-201330-team23,nepoted,walthecn

Game of Life hints:

- Follow the TODO's. *Test as frequently as practical.*
  - If a part is hard, break it down into sub-parts and test each sub-part as you go.
- There are at least 3 clever ways to avoid cluttering code that references cells with IF's to ensure that they are not "off the edge of the board", namely:
  - "Wrap". For example, if the board is 10x10, attempts to reference **`board[10][3]`** are converted to **`board[0][3]`** (use the **`%`** operator).
  - Write a "getter" that gets the value of a cell and returns a sensible value (0?) if the reference is off the edge of the board. Ditto for a "setter" if needed.
  - For a 10x10 board, declare a 12x12 board and make the outer shell all empty cells. You will find that you never make them non-empty (loop from 1 to 10, not 0 to 11), so all is well.

Animating Game of Life

- How: use `Timer` class to automatically “click” button
- Details: in `GameOfLifeMain`:
  - Use local variable for `UpdateButton` object
  - Add timer code to end of main to repeatedly click button at regular intervals:

    -
      ```
      Timer mrClicker =
            new Timer(INTERVAL, updateButton);
      mrClicker.start();
      ```
- Learn more: Big Java, Ch. 9.9

Work Time

- Game of life due 11:59 PM on day of next class
- Work with your partner
  on the Game of Life project
  - Get help as needed

Before you leave today, make sure that you and your partner have ***scheduled a session to complete the Game of Life project***

- Where will you meet?
  - ***Try the CSSE lab F-217/225***
- When will you meet?
  - ***Consider this evening***,
    7 to 9 p.m. ***Exchange contact info*** in case one of you needs to reschedule.
- **Do it *with your partner.*** If your partner bails out, DON'T do it alone until you communicate with your instructor.

Work Time

- Work with your partner on the GameOfLife project
  - Get help as needed
  - The TODOs are numbered – do them in the indicated order.
  - *Follow the practices of pair programming!*
- *Don’t do any of the work without your partner!*
- Good exam prep.

Live Coding

- Finish **RollingDice**, then continue on HW 6.

Q13–Q14