CSSE 220 Day 11

Two-dimensional arrays,
Copying arrays (shallow copies),
Software Engineering Techniques
(regression testing, pair programming, team version control)

Check out *TwoDArrays* from SVN

Questions?

Exam Coming!

See the Schedule page, Session 12, for a link to a document that lists the topics covered by this exam

- Test next Monday
  - Evening exam! Schedule page says where and when.
  - Exam is 7–9 p.m. but you may start the exam up to 1 hour early and stay up to 1 hour late (or both)
- Topics from Chapters 1–7
- Will include:
  - A closed-book paper part: short answer, fill-in-the-blank, trace-code-by-hand, draw box-and-pointer diagrams, find-errors-in-code, write short chunks of code
    - We will list in advance ALL the possible topics for this portion of the exam
  - A programming part: a few small programs, unit tests provided for some of them, you write unit tests for others
- Review in class Thursday
  - Bring questions
  - I won't prepare anything but am happy to cover whatever you want, including working examples

Q1

Two-dimensional arrays

```
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

Q2

Exercise

Complete the TODO items in TicTacToe and TicTacToeTest

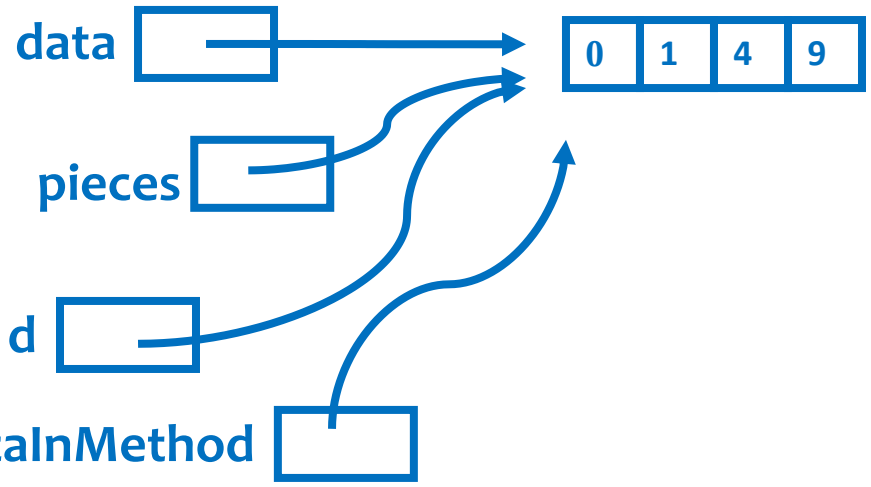
They're numbered; do 'em in order.

- The Tasks tab lists the TODO's.

The stub of the non-default constructor that we gave to you has a compile-time error; that is purposeful – you'll correct that error as part of your TODO 1.

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;
    ...
}
```

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

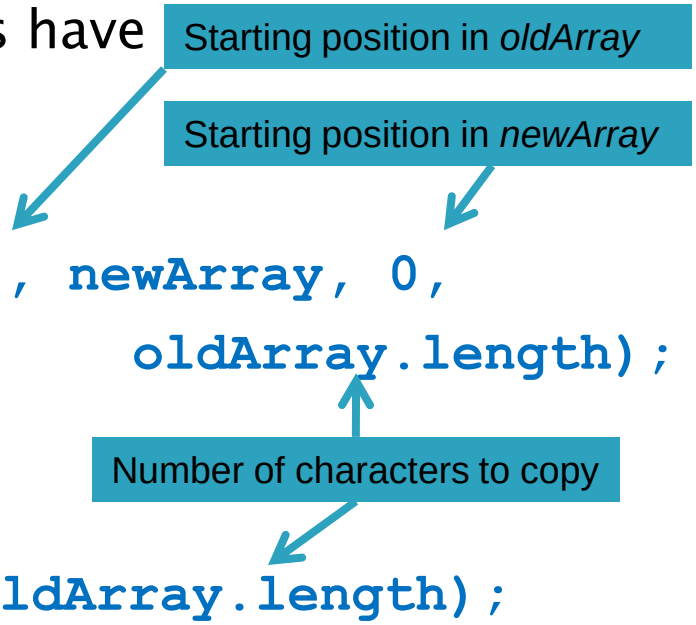
```
newArray = oldArray.clone();
```

  3. Use the *System.arraycopy* method:

```
System.arraycopy(oldArray, 0, newArray, 0,
                 oldArray.length);
```

  4. Use the *Arrays.copyOf* method:

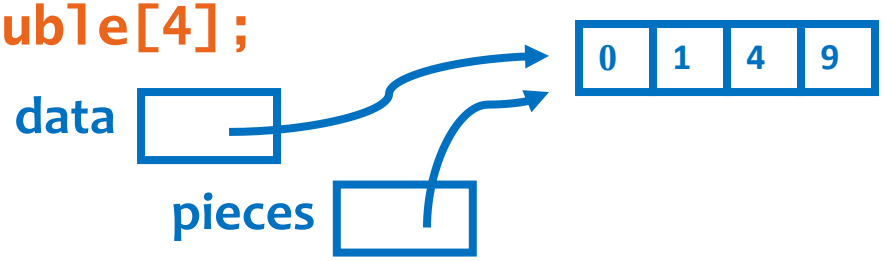
```
newArray = Arrays.copyOf(
                 oldArray, oldArray.length);
```



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

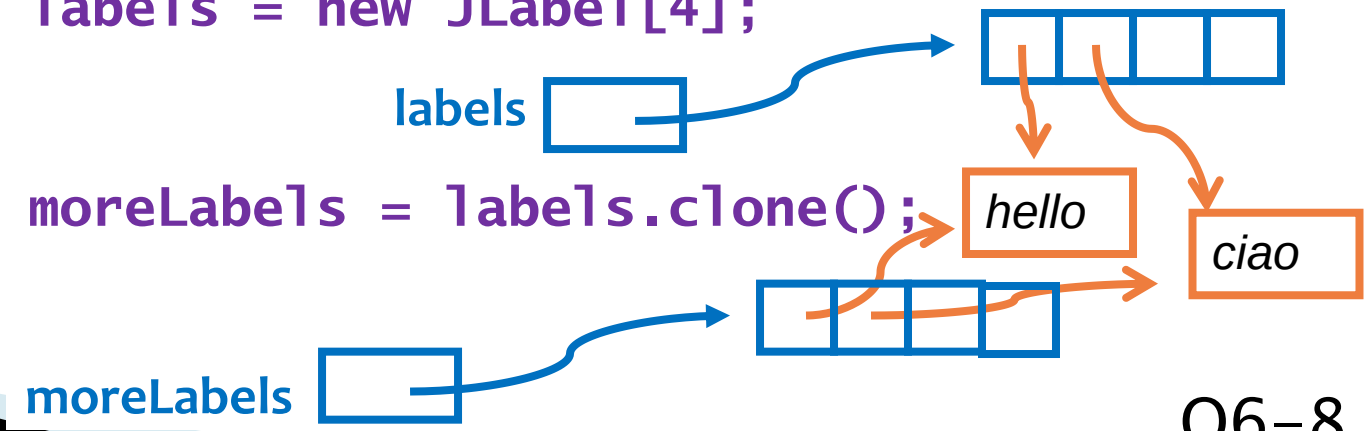
Copying Arrays – Shallow copies

- 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();`



Quality Tip –"Avoid parallel arrays"

- We avoided parallel arrays in our 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?
- We did (unwisely?) use parallel arrays in StateListTest:

```
this.inputs = new ArrayList<String>();
this.correctResults = new ArrayList<String>();
```

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
  - More general because it can be extended

- 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, the keep a bug from coming back
- Remember:
  - You can right-click the project in Eclipse to run all the unit tests

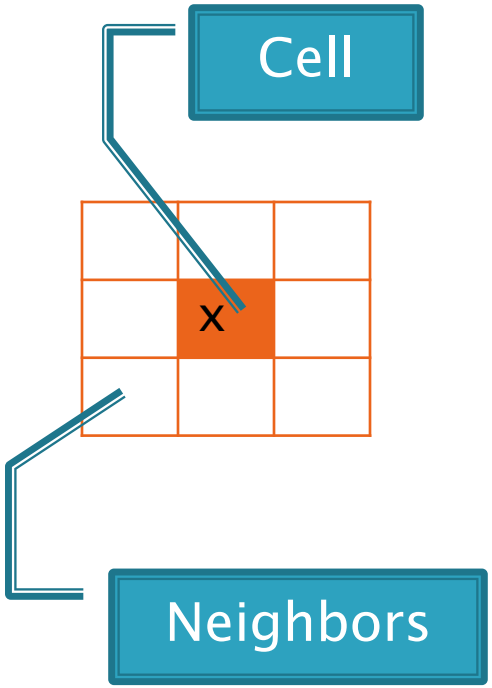
Pair Programming

- Video

http://agile.csc.ncsu.edu/pairlearning/educators.php#ppvideo

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



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

Game of Life Teams – Boutell

n	Team
01	lint,roserrm
02	klaassmj,baldwicd
03	wardsr,zimmerka
04	degrotpc,evansea
05	ernsteac,houstoef
06	audretad,geislekj
07	lamantds,maderli
08	wieganda,vermiljb
09	draycs,lapresga
10	weavergg,fryjc

n	Team
11	knightbk,cahilltr
12	channmn,hopkinaj
13	hannantt,kautzjr
14	shumwanm

Driver (and ONLY the Driver):
Check out *GameOfLife* from SVN

- The Navigator will check out the project in the next session, after today's changes are committed.

The TODO's are numbered – do them in the indicated order.

Follow the practices of pair programming!

Team number used in repository name:
http://svn.csse.rose-hulman.edu/repos/csse220-201020-life-teamXX

Game of Life Teams – Mutchler

n	Team
21	Ahmed Alshaali & Ian Cundiff
22	Kyle Apple & Alex Mullans
23	Tom Atnip & George Mammarella
24	Jeremy Bailey & Ryan Fuller
25	Devon Banks & Chase Mathison
26	Susan Cisneros & Katie Greenwald
27	Brian Collins & Jackson Melling
28	Alex Gumz & Richard Thai
29	Elizabeth Hines & Ben McDonald
30	Rebecca McCarthy & Ann Say

n	Team
31	Brad Quamme & Franklin Totten
32	Ruben Rodriguez & Nathan Varner

Driver (and ONLY the Driver):
Check out *GameOfLife* from SVN

- The Navigator will check out the project in the next session, after today's changes are committed.

The TODO's are numbered – do them in the indicated order.

Follow the practices of pair programming!

Team number used in repository name:
http://svn.csse.rose-hulman.edu/repos/csse220-201020-life-teamXX