

# 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?

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

## Two-dimensional arrays

```
/**
 * Constructs a 3x3 TicTacToe board with all squares blank.
 */
```

```
public TicTacToe() {
    this.rows = 3;
    this.columns = 3;
```

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

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

```
    for (int r = 0; r < this.rows; r++) {
```

Could have used:  
`this.board.length`

```
        for (int c = 0; c < this.columns; c++) {
```

```
            this.board[r][c] = " ";
```

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

```
        }
```

```
    }
```

```
}
```

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

# 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);
```

Starting position in *oldArray*

Starting position in *newArray*

Number of characters to copy

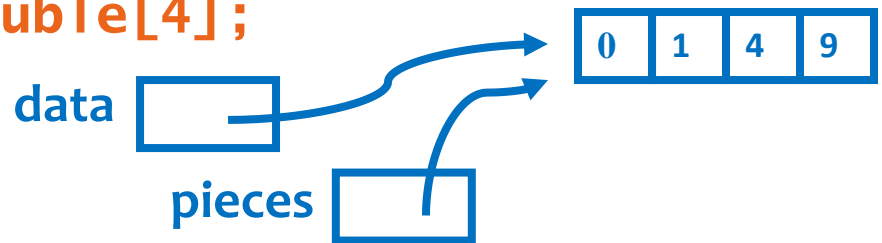
The key point is that all of these except possibly the first make **shallow copies** – see next slide

# Copying Arrays – Shallow copies

- ▶ Can copy whole arrays in several ways:

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

- `...`  
`pieces = data;`

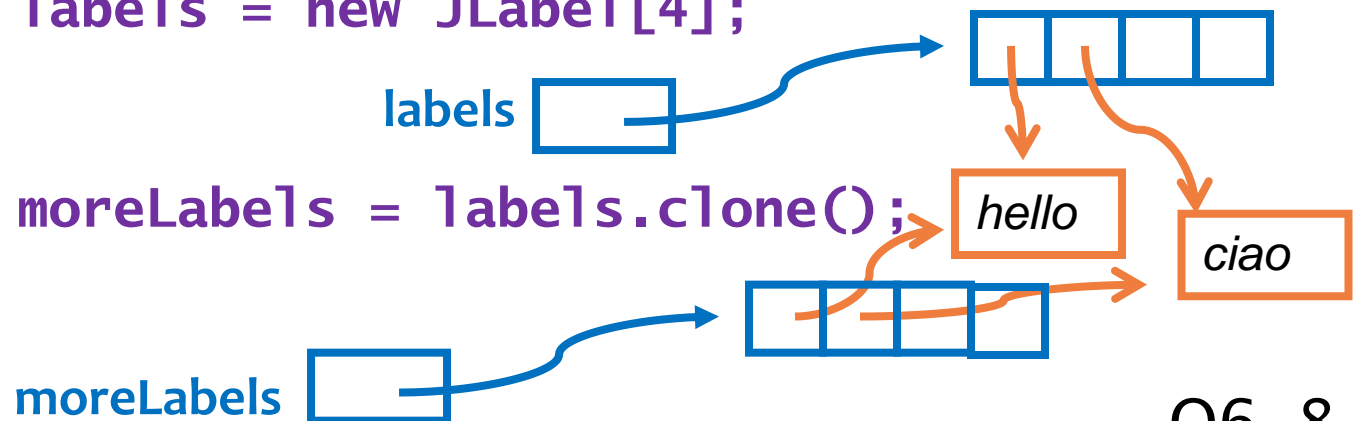


- `double pizzas = data.clone();`

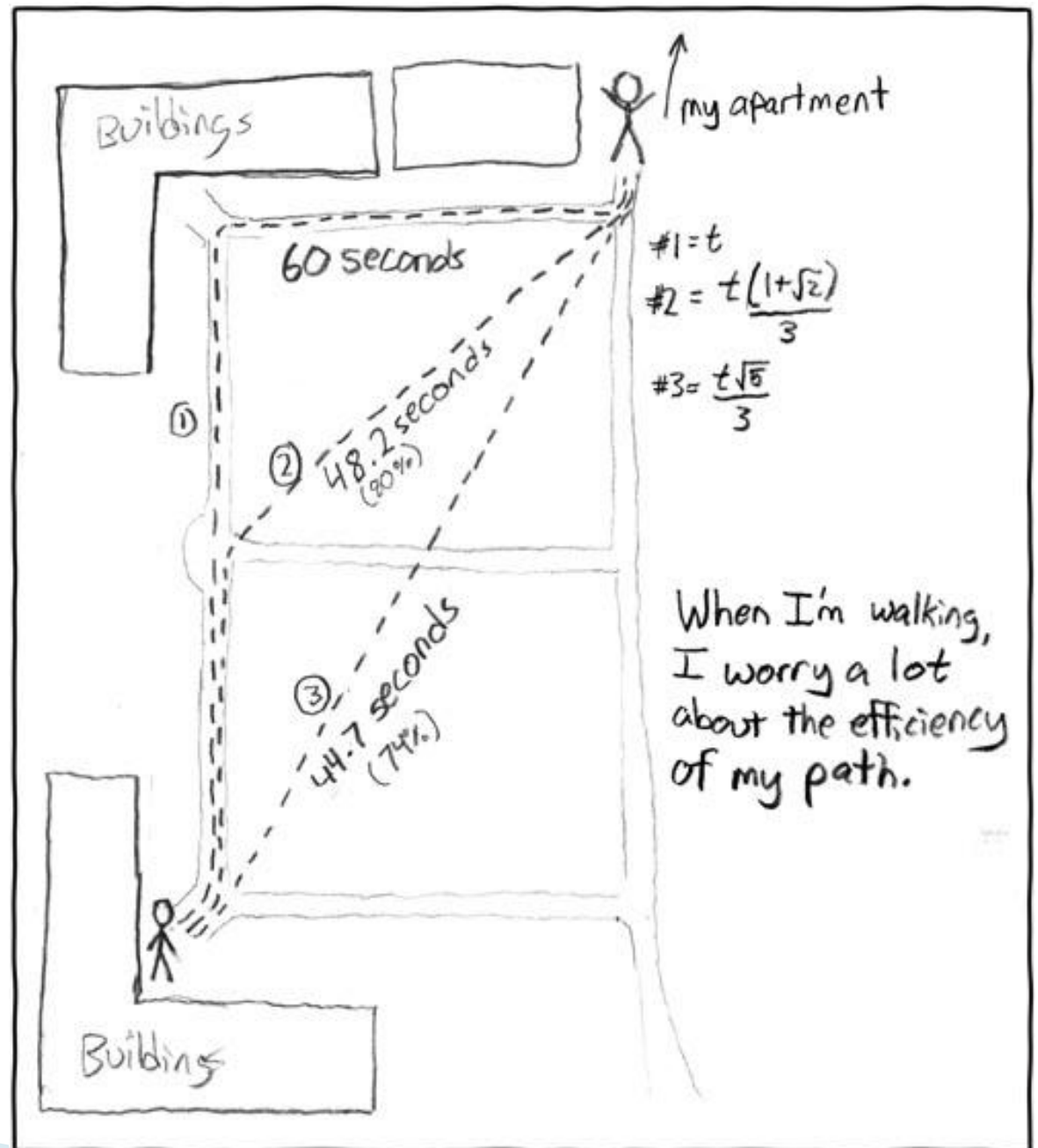


- `JLabel[] labels = new JLabel[4];`

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



# Interlude:



<http://xkcd.com/85/>



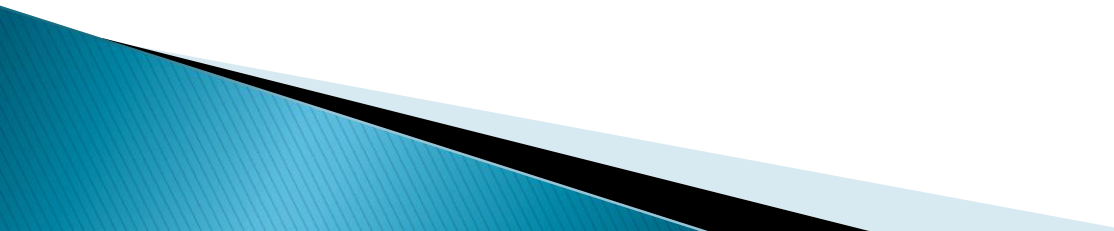
# Quality Tip – “Avoid parallel arrays”

- ▶ 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?
- ▶ 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

# Software Engineering Techniques

- ▶ Regression testing
  - ▶ Pair programming (next class)
  - ▶ Team version control (next class)
- 

# Regression Testing

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

# Exam Tomorrow

See the [Schedule page](#), Session 12, for a link to a document that lists the topics covered by this exam

- ▶ Test Friday
  - In class but you may have up to 50 mins of extra time. You can work from at 7:10–8:00 am or any of hours 1–4 that you are free.
  - If you can't do it in one contiguous chunk, you can only leave between the two parts of the exam – plan accordingly.
- ▶ 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 have listed ALL the possible topics for this portion of the exam
  - A programming part: 1–2 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

# Homework

- ▶ There is nothing to turn in for Homework 12
  - It just points you toward resources you might find helpful in preparing for the exam (or in taking it!)
  - You can email csse220–staff if you need help.
- ▶ **There IS something to do for Homework 13**
  - Reading and assessment on Angel over the reading
  - **Due at the beginning of Session 13 (Tuesday), as usual**