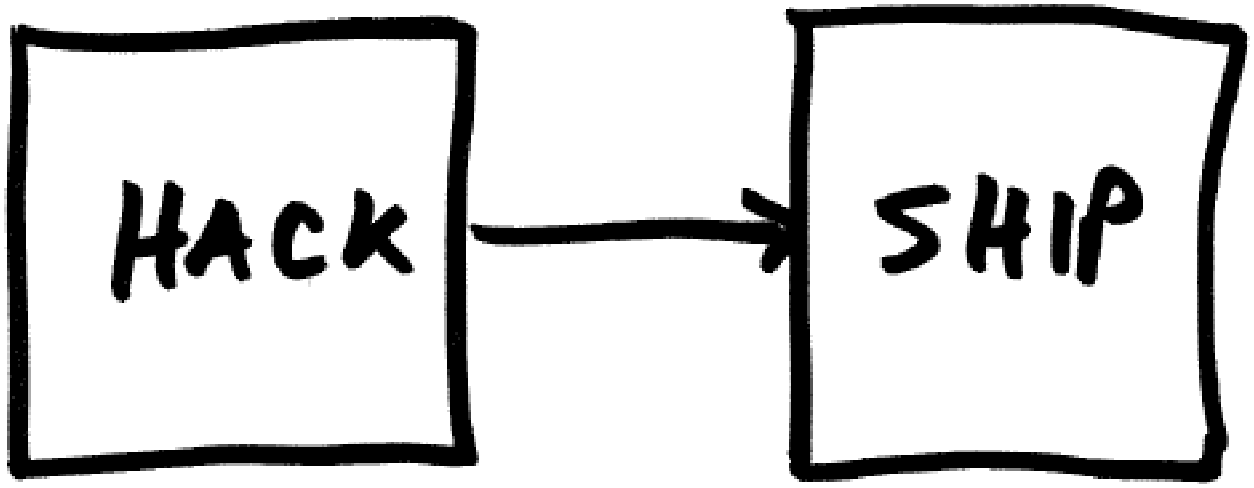
CSSE 220 Day 10

Some Software Engineering Techniques
(Class Diagrams, Pair Programming
& Version Control)
Game of Life Exercise

Software Process: The Early Days



So, what is Software Process?

Hint: software is the part of a computer system that is suppose to change!

- Take 15 seconds and think about it
- Turn to neighbor and discuss what you think for a minute
- Let's talk?

Iterative

Waterfall

Incremental

Spiral

Extreme Programming

Producing Software is an Elaboration and Refinement Process

- Starting with Abstract Requirements, successively *Elaborate* and *Refine* them into specifications, models, and more concrete implementation

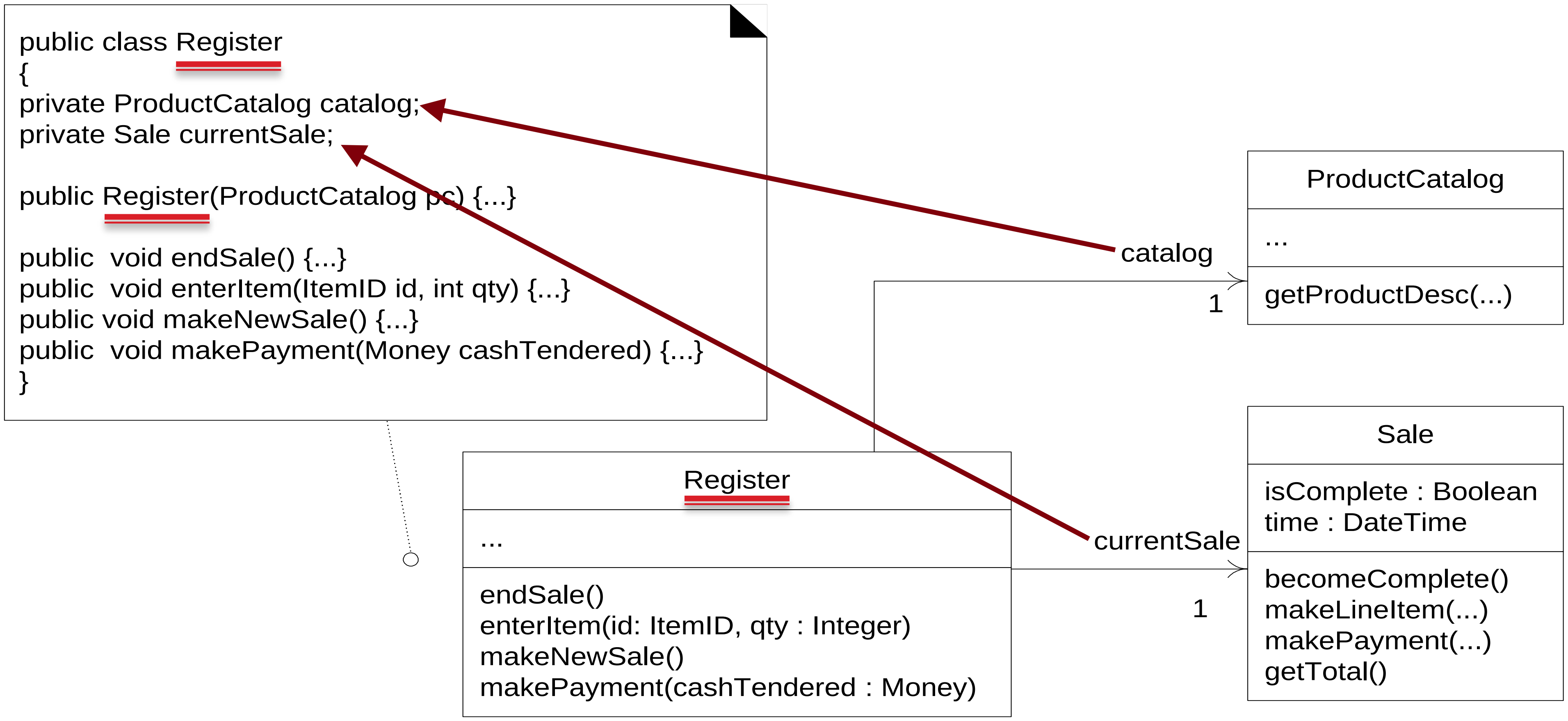
- A Software Process organizes the life cycle activities related to the creation, delivery, and maintenance/evolution of software systems

Q1, 2

Software Engineering Techniques

- Class Diagramming
- Pair programming
- Team version control
- Brief mention of Regression Testing

Diagramming Classes

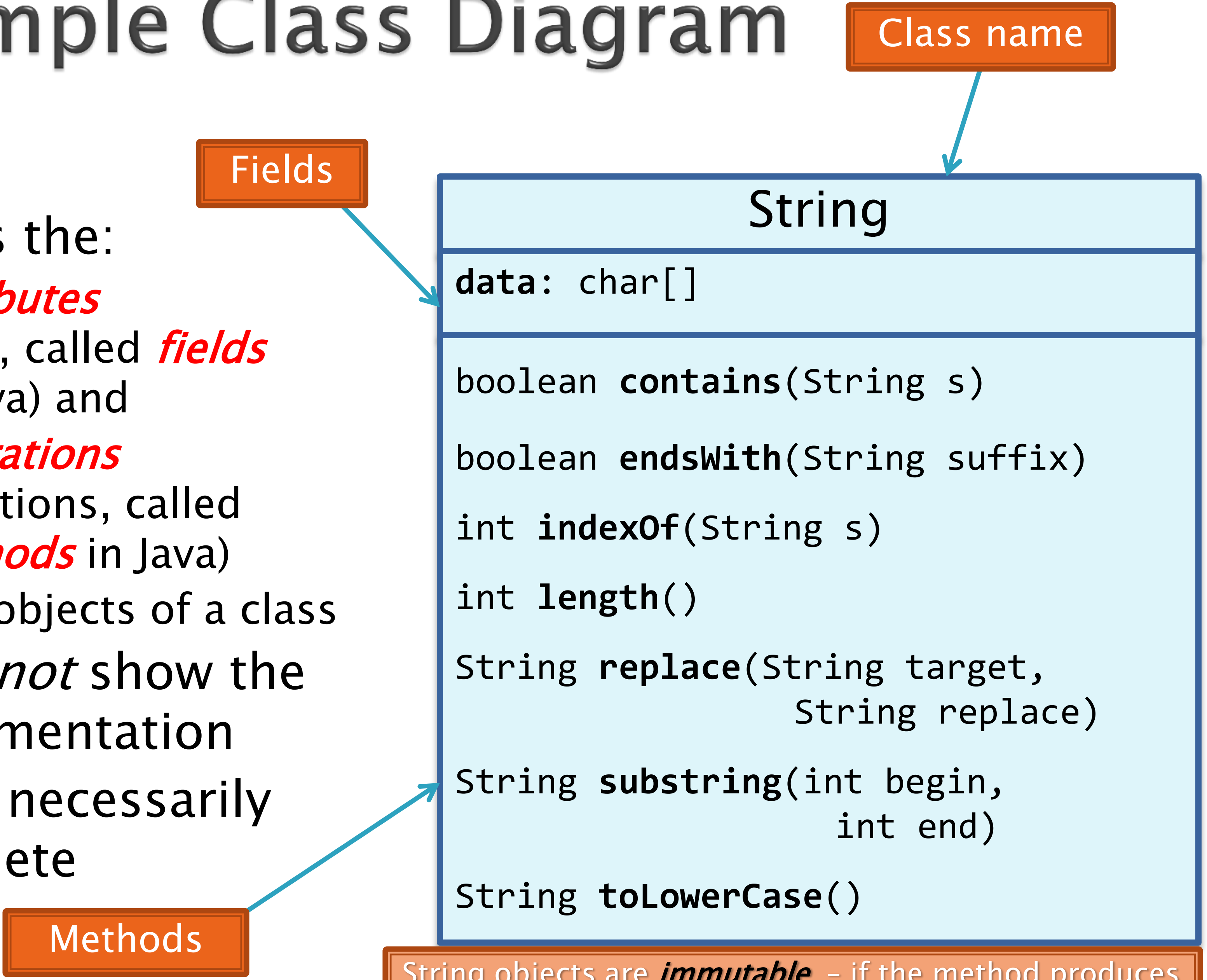


```
public class Register
{
private ProductCatalog catalog;
private Sale currentSale;

public Register(ProductCatalog pc) {...}

public  void endSale() {...}
public  void enterItem(ItemID id, int qty) {...}
public void makeNewSale() {...}
public  void makePayment(Money cashTendered) {...}
}
```

Example Class Diagram

- Shows the:
  - ***Attributes*** (data, called ***fields*** in Java) and
  - ***Operations*** (functions, called ***methods*** in Java)

  of the objects of a class
- Does *not* show the implementation
- Is *not* necessarily complete



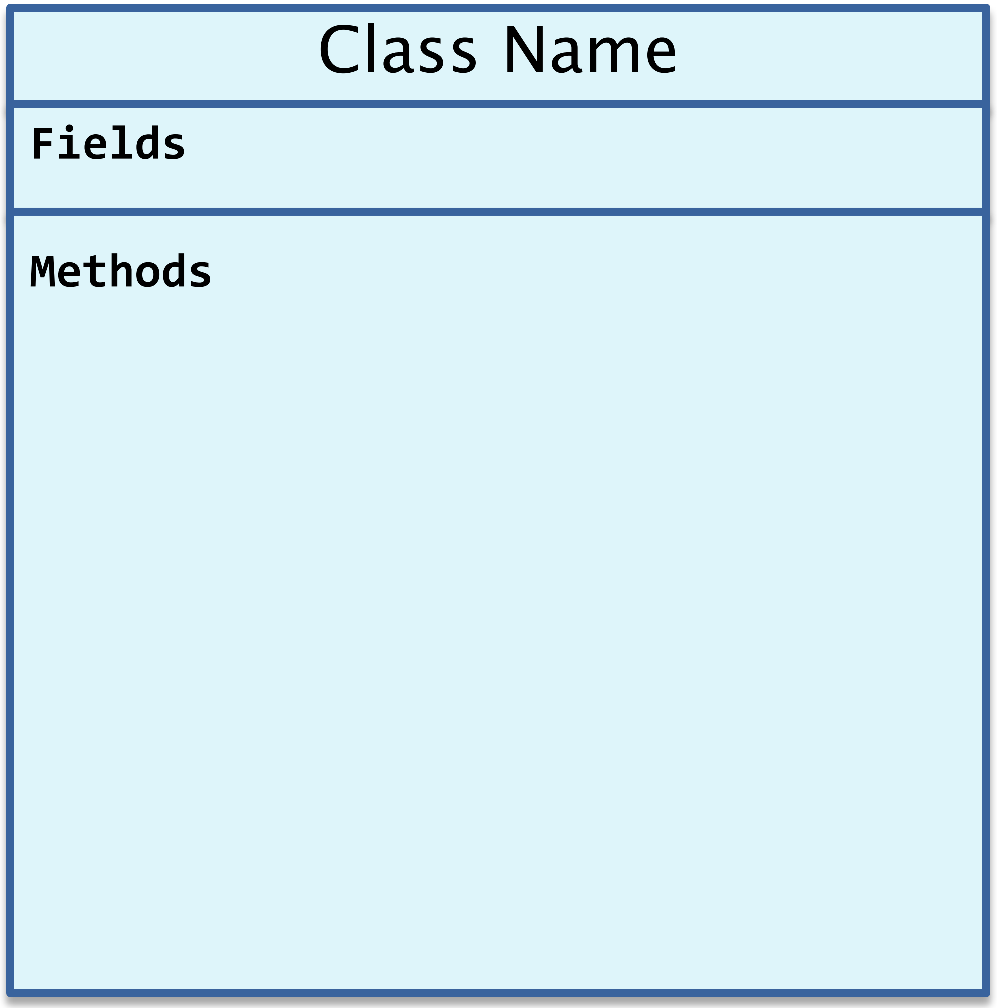
```
String
----------------------------------------
data: char[]
----------------------------------------
boolean contains(String s)
boolean endsWith(String suffix)
int indexOf(String s)
int length()
String replace(String target,
                String replace)
String substring(int begin,
                  int end)
String toLowerCase()
```

String objects are ***immutable*** – if the method produces a String, the method *returns* that String rather than mutating (changing) the implicit argument

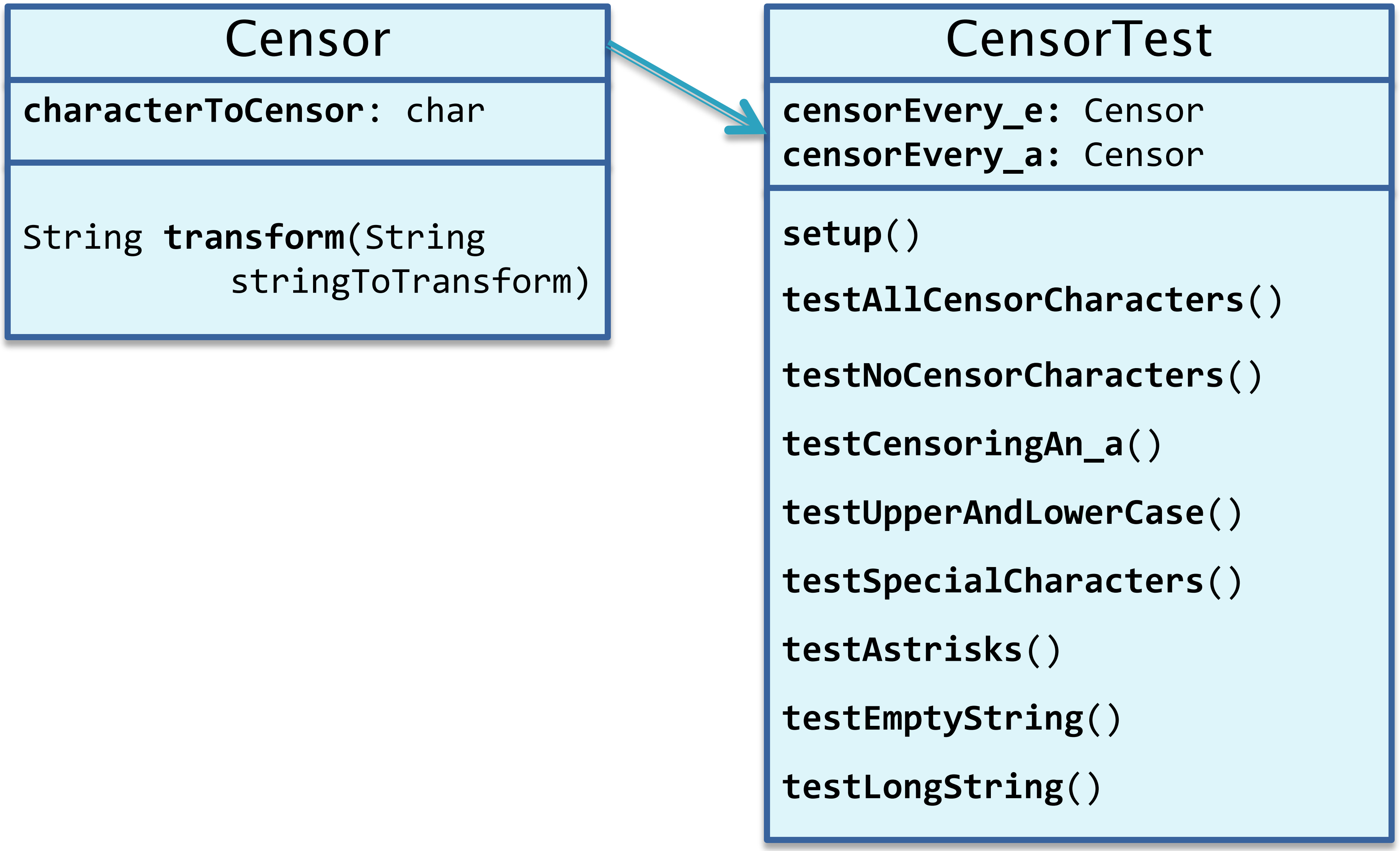
Q3

Exercise: Class Diagrams

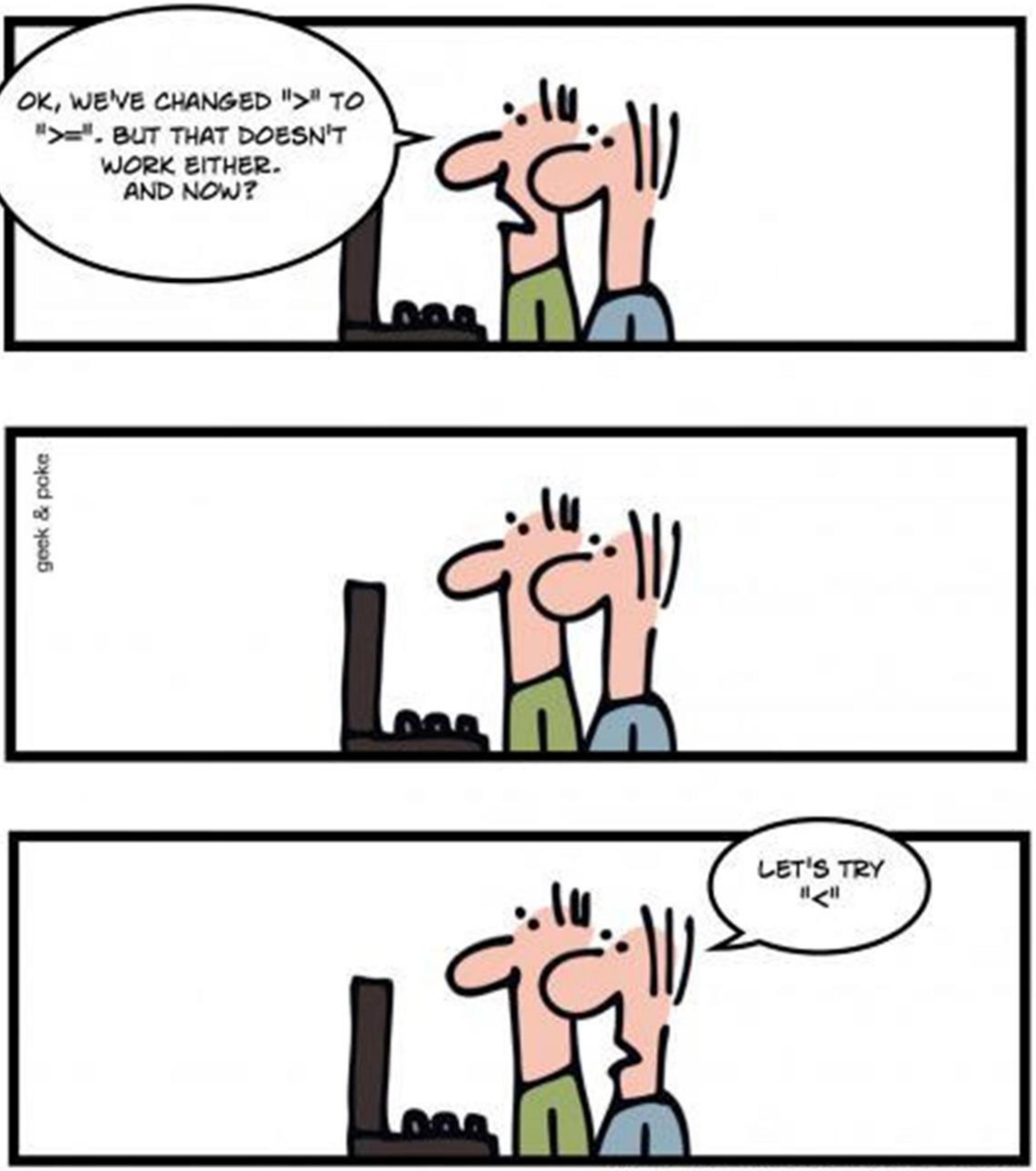
- Task: Make Class diagrams for the Censor and CensorTest classes from Word Games



Exercise: Class Diagrams



GOOD CODERS...


OK, WE'VE CHANGED ">" TO ">=". BUT THAT DOESN'T WORK EITHER. AND NOW?
geek & poke
LET'S TRY "<"

... KNOW WHAT THEY'RE DOING

What Is Pair Programming?

- Two programmers work side-by-side at a computer, continuously collaborating on the same design, algorithm, code, and/or test
- Enable the pair to produce higher quality code than that produced by the sum of their individual efforts
- Let's watch a video…

Q4

Pair Programming

- Working in pairs on a single computer
  - The *driver*, uses the keyboard, talks/thinks out-loud
  - The *navigator*, watches, thinks, comments, and takes notes
  - Person who really understands should start by navigating ☺
- For hard (or new) problems, this technique
  - Reduces number of errors
  - Saves time in the long run

Q5

How Does This Work? (1 of 2)

- Pair-Pressure
  - Keep each other on task and focused
  - Don’t want to let partner down
- Pair-Think
  - Distributed cognition:
    - Shared goals and plans
    - Bring different prior experiences to the task
    - Must negotiate a common shared of action
- Pair-Relaying
  - Each, in turn, contributes to the best of their knowledge and ability
  - Then, sit back and think while their partner fights on

Q6

How Does This Work? (2 of 2)

- Pair-Reviews
  - Continuous design and code reviews
  - Improved defect removal efficiency (more eyes to identify errors)
  - Removes programmers distaste for reviews (more fun)

- Debug by describing
  - Tell it to the “Rosie in the Room”

- Pair-Learning
  - Continuous reviews → learn from partners
  - Apprenticeship
  - Defect prevention always more efficient than defect removal

Q7

Abstracted from: Robert Kessler and Laurie Williams

Partnering the Pair

Expert paired with an Expert

Expert paired with a Novice

Novices paired together

Professional Driver Problem

Culture

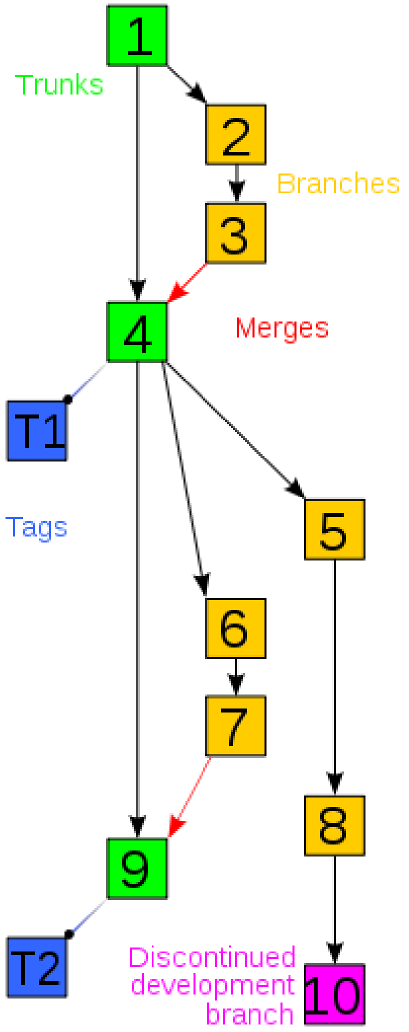
Source: Robert Kessler and Laurie Williams

What can go wrong when you are working with your team on the same system artifacts?

- Take 15 seconds and think about it
- Turn to neighbor and discuss what you think for a minute and list a few examples
- Let’s talk?

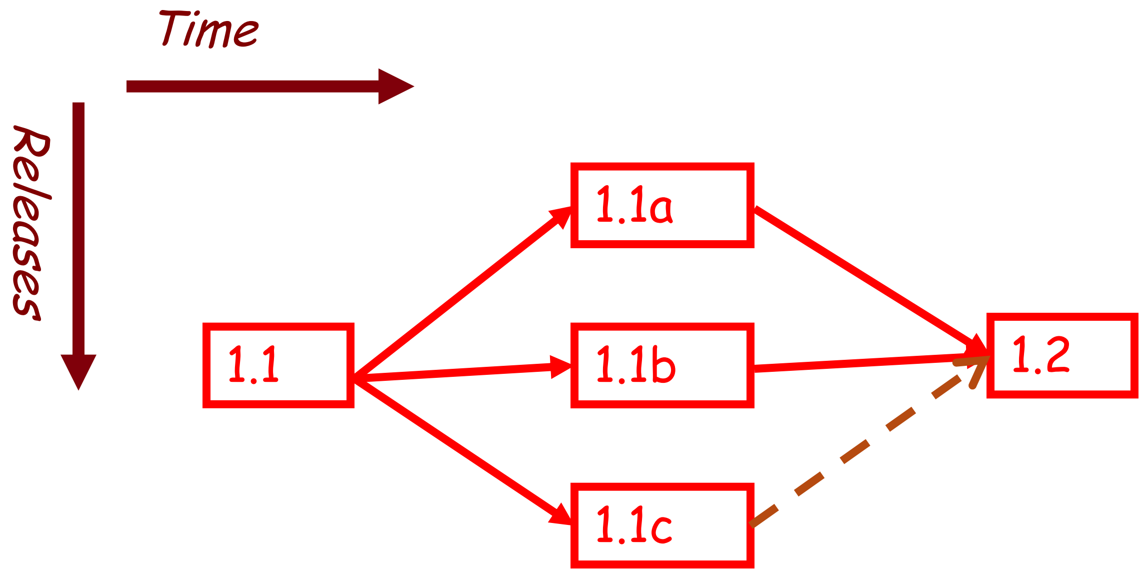
Software Has Multiple Versions

- Why? Again, software is suppose to change …
- Different releases of a product
- Variations for different platforms
  - Hardware and software
- Versions within a development cycle
  - Test release with debugging code
  - Alpha, beta of final release
- Each time you edit a program



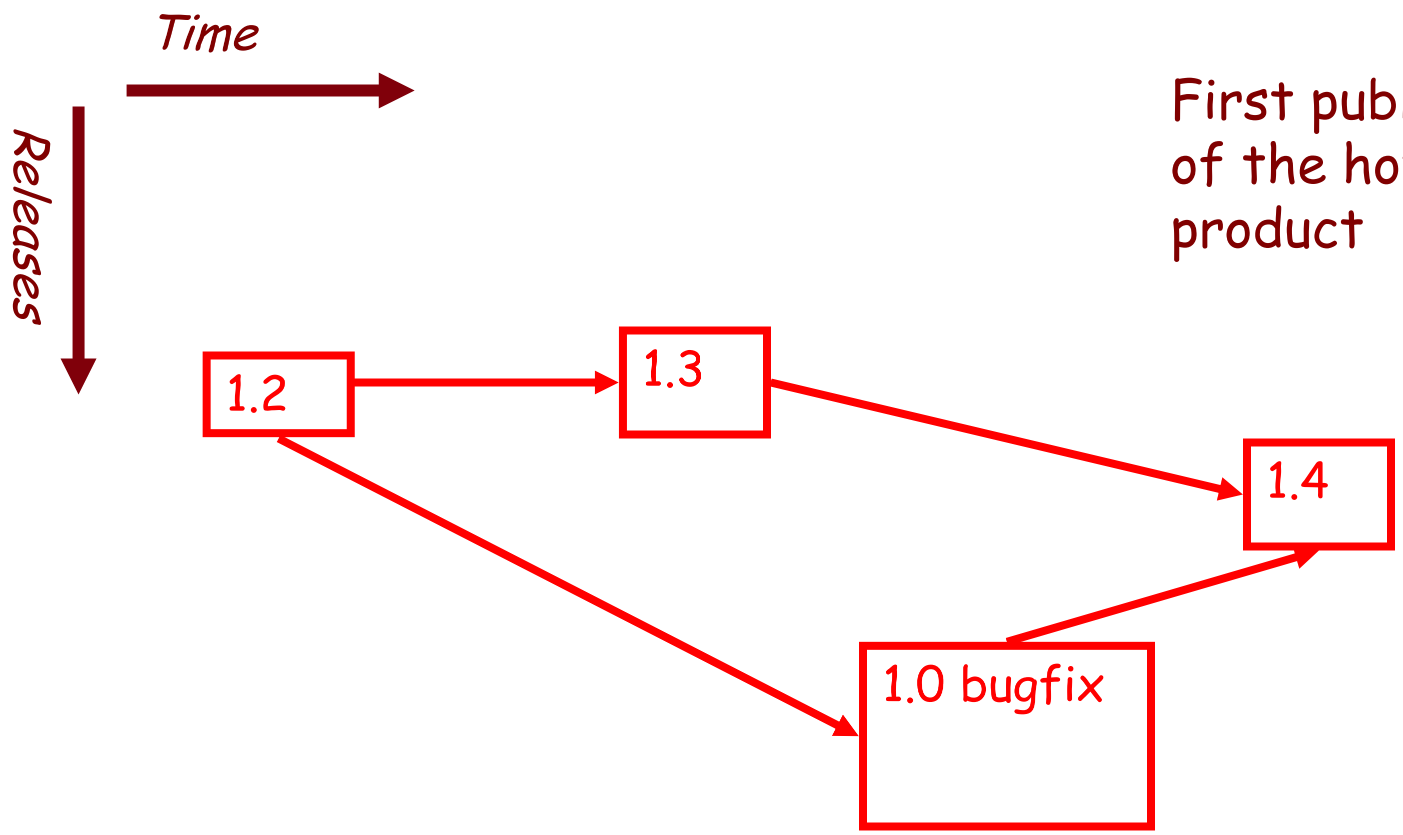
Q8

Scenario I: Normal Development



You are in the middle of a project with three developers named a, b, and c.

Version Control Scenario II: Bug Fix



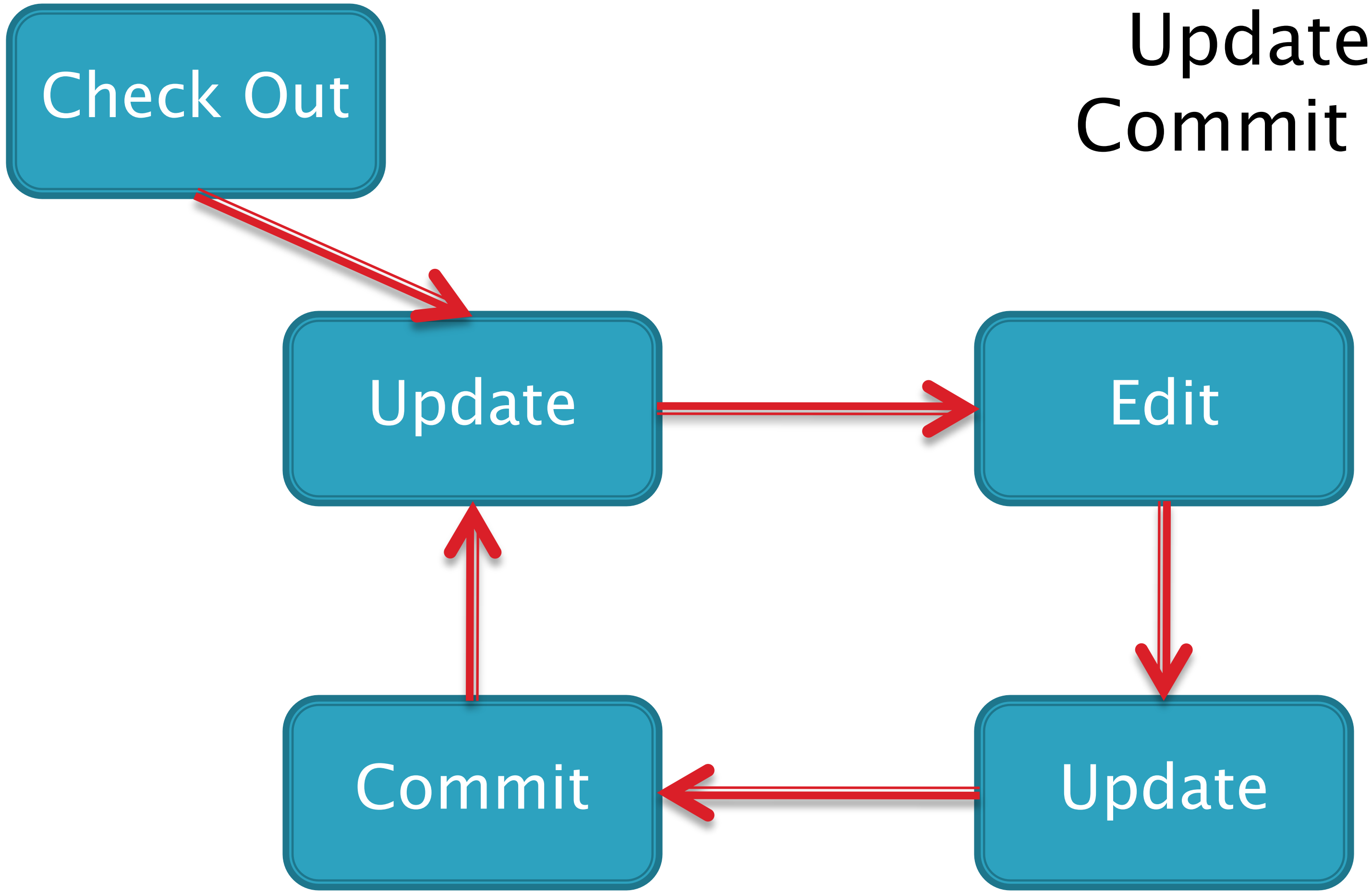
First public release of the hot new product

Team Version Control

- **Version control tracks multiple versions**
  - Enables old versions to be recovered
  - Allows multiple versions to exist simultaneously
- **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 ameliorates this issue ☺

Q9

Team Version Control



Update and
Commit often!

Why do you keep versions of the test suite under configuration management?

- Take 15 seconds and think about it
- Turn to neighbor and discuss what you think for a minute
- Let's talk?

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

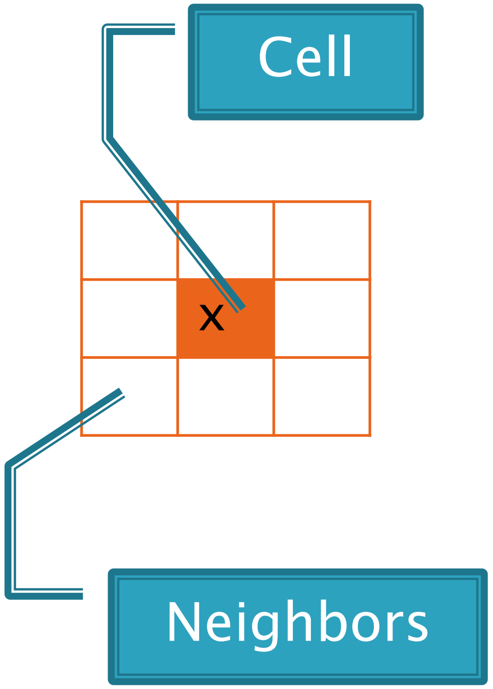
Checkout Today's work

- Go to SVN repository view at bottom of workbench
  - Window➔ show view➔ Other➔ SVN➔ SVN Repositories

- Right click in SVN View, then choose New SVN Repository Location
  - http://svn.csse.rose-hulman.edu/repos/csse220-201420-"your_team_repository"

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

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!*