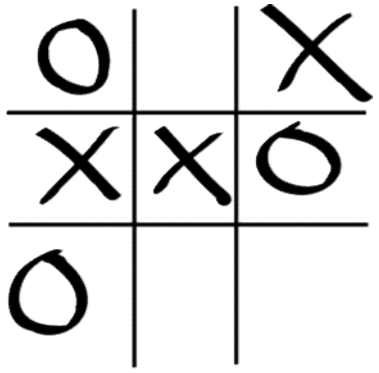
Building Tic Tac Toe with SWING API

CSCI 2300

Simple game of tic tac toe



- GUI front-end
- Object-oriented
- Can play against another player or against computer

Where do we start?

Top down
- Develop a prototype
- Determine sequence of events
- Implement main sequence
  - Break problem into classes
- Implement next sequence
  - May need to restructure classes
- Continue until all requirements are met

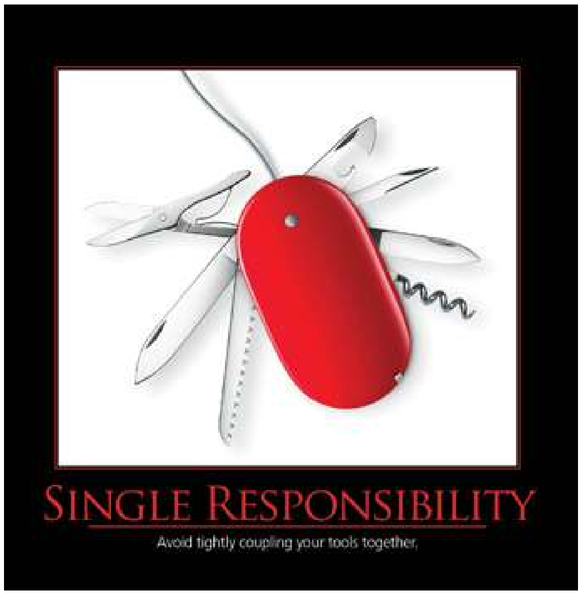
Bottom up
- Break the problem into classes
- Define class interfaces
- Integrate classes together: make sure they work with each other
- Develop the GUI layer
- Connect your classes to the GUI layer
  - May need to restructure classes

Develop a prototype

- In your git repos you will find a tic_tac_toe directory with GameGUI.java
- Non-functional prototype
  - Presents the look
  - Does not work
- What is GameGUI class responsible for?
- GameGUI has too many responsibilities

Single Responsibility Principle (SRP)

- Each class has one responsibility (and one reason to change)



GameGUI has many responsibilities

- Arrange GUI components
- Implementation of "announcement panel"
- Implementation of "score board panel"
- Implementation of "board panel"
- Many reasons to change GameGUI class

How can we fix it?

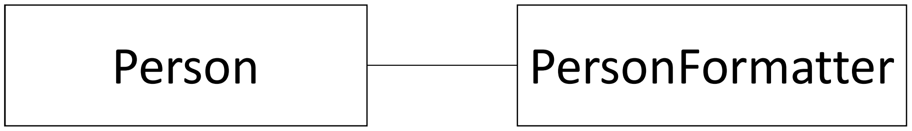
- Split each component into a separate class
  - AnnouncementGUI
  - ScoreBoardGUI
  - BoardGUI
- Each class is responsible for implementing the behavior of the component it represents
- GameGUI simply arranges GUI compoenents.
- Look in tic_tac_toe/v1 directory

```
class Person {
    protected String firstName;
    protected String lastName;
    protected Gender gender;
    protected DateTime dateOfBirth;
    public string Format(string formatType)
    {
        switch(formatType)
        {
            case "XML":
               return xmlFormattedString; break;
            case "FirstAndLastName":
              return firstAndLastNameString; break;
            default:
              // implementation of default formatting
              return defaultFormattedString;
        }
    }
}
```

Does this code violate SRP?

A. This class violates SRP because it encapsulates multiple attributes of a person
B. This class violates SRP because it does not have a constructor
C. This class violates SRP because it is responsible for encapsulating "person" attributes and formatting them
D. This class does not violate SRP

How can we fix it?



Encapsulates Person attributes

Formats Person as String, based on desired format

Back to Tic Tac Toe

- Making the prototype work
- Scenario 1: When board button is clicked, current player's game piece gets placed on that button
  - GamePiece
  - BoardButton
  - Player
- Which class should be responsible for Scenario 1?

A. GameGUI
B. BoardGUI
C. Player
D. BoardButton
E. A new, not yet defined class

Design continued

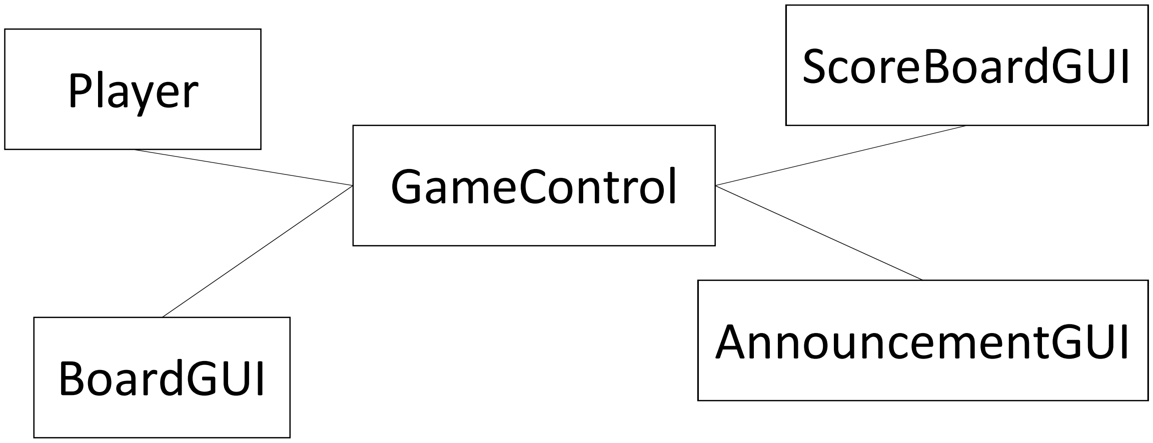
- Scenario 2: When the board contains three identical game pieces in a row, the game is over.
  - Check if there are three identical pieces in a row
  - If yes, disable all board buttons
  - Update announcement
  - Update score
- Which class should be responsible for implementing Scenario 2?

A. GameGUI
B. BoardGUI
C. ScoreBoardGUI
D. BoardButton
E. A new, not yet defined class

Need a new class to maintain game state

- GameControl
  - Maintains and updates game state (implements scenarios 1 & 2)
    - Who is the current player
    - What moves have been made (state of the game board)
  - Needs access to:
    - ScoreBoardGUI
    - AnnouncementGUI
    - Both players
    - Board buttons

Tight coupling: classes have a high degree of dependence of classes on each other. If one class changes, the other may have to change too



How can we fix it?

- Use Interfaces
- IBoard, IScore, IAnnouncement, IPlayer
- Delegate responsibilities to classes that implement these interfaces
- Example:
  - IBoard – implements methods for keeping the state of the board
  - IScore – implements methods for keeping track of score
  - etc

Lab 11: tic_tac_toe/v2

- Added GameControl class
- Added several interfaces
- TODO: have existing GUI classes implement the new interfaces
  - Update access modifier in interfaces to be **public**
  - ScoreBoardGUI implements IBcore
  - BoardGUI implements IBoard
  - AnnouncementGUI implements IAnnouncement