* Introduction
  + Who am I
    - Who am I? LinkedIn says I’m a 31-year-old engineer at S&C Electric
  + What is my experience
    - What is my experience with Object Oriented Programming? I took a few classes on Java in undergrad, I did some professional work in C# and Java throughout my career
  + Thesis Statement:
    - Over the course of this project for the development of jPaint(CrappyPaint on GitHub), I will be going over what went right and wrong from a development standpoint, the design patterns I used to handle the various aspects of it, some functional information about the application itself, and the required deliverables(Documented Source Code, UML Diagrams, Maybe code metrics if I’m feeling fancy).
* Successes and Failures
  + What went right?
    - Got super comfortable with the development environment
    - Reasonable development pace
    - Good response to compilation and runtime errors to never commit bad
  + What went wrong?
    - Took way too long to get coloring to work
    - Had to scrap a ton of code early on because AWT supplied better code for it
    - Bugs that took too long to resolve hindered development
  + What could have been better?
    - If I had read the documentation on swing earlier, a lot of scrapped code wouldn’t have happened.
    - If I had documented project requirements correctly into tasks in GitHub Projects, I would have avoided wasting time on an improper implementation
    - Had reached out for help from other students earlier
* Design Patterns Used
  + What is a design pattern and why it’s important?
    - What is it?
      * It’s a reusable solution to a commonly occurring problem within a given context in software design
    - Why is it important?
      * Design patterns can speed up the development process by providing tested, proven development paradigms.
  + Adapter pattern
    - What is it
      * An Adapter method is a design pattern which makes incompatible objects adaptable to each other.
    - How I used it
      * Used it to convert mouse coordinates into coordinate points
    - What classes it impacted
      * MouseAdapter.java used the Adapter pattern to convert mouse movement on a coordinate plane into points in which operations can happen based on those inputs.
  + Factory pattern
    - What is it
      * A factory pattern is designed to make objects
    - How I used it
    - What classes it impacted
  + Strategy pattern
    - What is it
      * A strategy pattern is designed to modify objects
    - How I used it
      * I use it to set drawing modifiers
    - What classes it impacted
  + Null Object pattern
    - What is it
      * Abstract Null objects
    - How I used it
      * End Point Type is set with this
    - What classes it impacted
  + Singleton pattern
    - What is it
      * It is a way to provide one and only one object of a type
    - How I used it
      * I use it to return color
    - What classes it impacted
      * I originally made a color singleton class, but that didn’t work so another classmate had instructed using ShapeColor to hold my color singleton.
* Functional Notes
  + Miscellaneous
    - Tools used
      * IntelliJ IDEA Community 2020.1
      * Java jdk 1.8
      * Git v2.14.1.windows.1
      * Understand v5.1
    - GitHub
      * GitHub Information
        + URL
      * GitHub Releases
        + Tagged Commits
      * GitHub Projects
        + Sprint #1
        + Sprint #2
        + Sprint #3
        + Sprint #4
      * GitHub Actions
        + Would have been nice to setup automated unit and UI tests into my commit process
    - Troubleshooting Techniques
      * Google-Fu
      * Rubber Ducky Programming
  + Bugs (as of 08/08)
    - Move only moves the most recent shape
      * How to reproduce
    - Select changes the property of the shape
      * How to reproduce
  + Missing Features (as of 08/08)
    - Group/Ungroup – Missing because not enough time to add the functionality to app
  + Extra Credit
    - Not applicable because primary functionality was never reached
    - Stretch Goals
      * Automated Build Pipeline
      * Automated UI/Unit Tests
      * Add Keyboard shortcuts
* Deliverables
  + Zipped up code base
  + Attached UML Diagram
  + Attached Code Metrics
* Conclusion
  + What I learned
    - Design Patterns are an elegant solution to approaching common programming strategies.
  + What I will take away from this
    - Implement design patterns into my daily programming at work.