

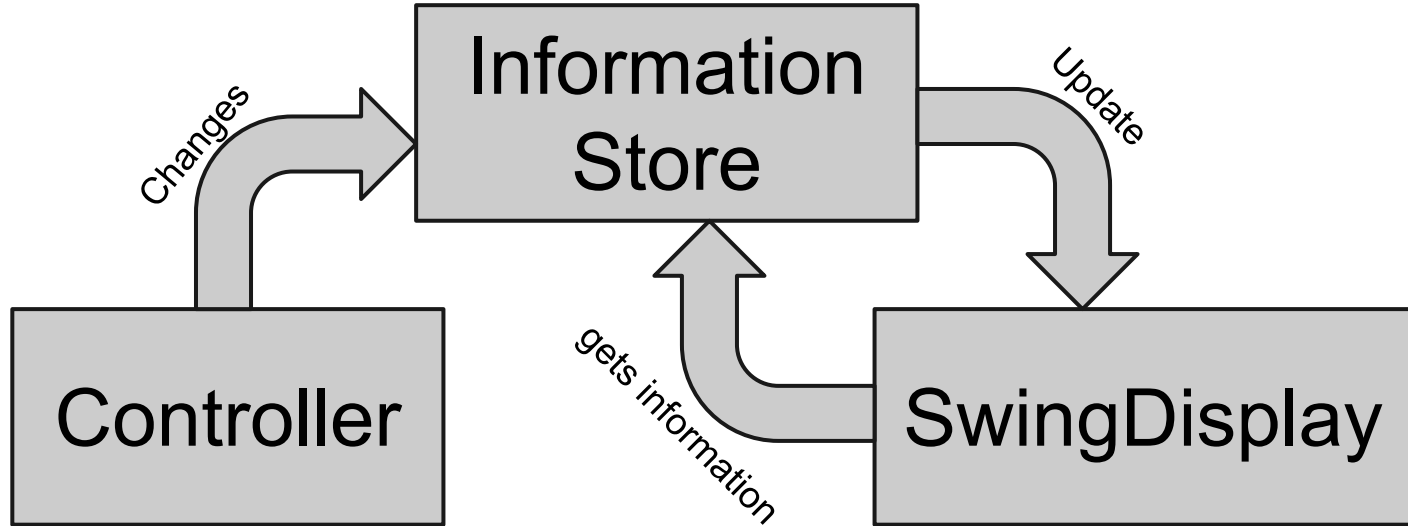
# Finite State Machine

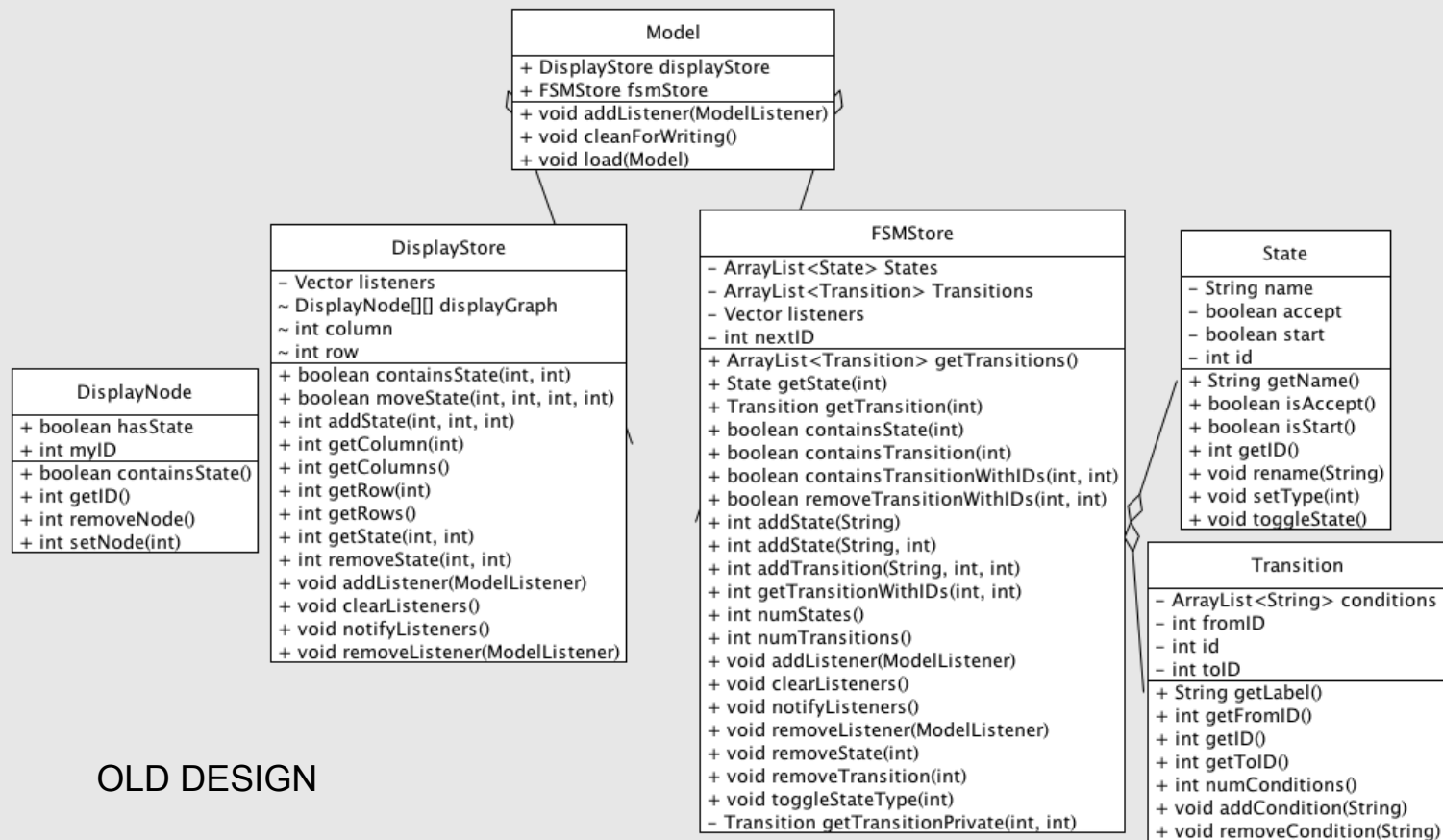
Frank, Rudy & Nate

# Updates:

What have we done?

# Model-View-Control Layout





OLD DESIGN

# Storage Changes: Facade Pattern

(pulling the functionality into the model itself)

```

<<interface>>
InformationStore

+ ArrayList<Transition> getTransitions()
+ LinkedList<Integer> getNextStates(String, int)
+ State getState(int)
+ State getState(int, int)
+ boolean containsState(int, int)
+ int getColumn(int)
+ int getColumns()
+ int getRow(int)
+ int getRows()
+ int getStart()
+ void addListener(InformationStoreListener)
+ void addState(String, int, int)
+ void addTransition(String, int, int, int)
+ void clearHighlights()
+ void clearListeners()
+ void highlight(int)
+ void load(InformationStore)
+ void moveState(int, int, int, int)
+ void notifyListeners()
+ void removeListener(InformationStoreListener)
+ void removeState(int, int)
+ void removeTransition(int, int, int, int)
+ void toggleStateType(int, int)
+ void unhighlight(int)

```

```

<<interface>>
InformationStoreListener

+ void update()

```

```

StoreException

```

```

InformationStoreImpl

- DisplayStore displayStore
- FSMStore fsmStore
- Vector listeners
- static final int DIMENSION

+ ArrayList<Transition> getTransitions()
+ LinkedList<Integer> getNextStates(String, int)
+ State getState(int)
+ State getState(int, int)
+ boolean containsState(int, int)
+ int getColumn(int)
+ int getColumns()
+ int getRow(int)
+ int getRows()
+ int getStart()
+ void addListener(InformationStoreListener)
+ void addState(String, int, int)
+ void addTransition(String, int, int, int)
+ void clearHighlights()
+ void clearListeners()
+ void highlight(int)
+ void load(InformationStore)
+ void moveState(int, int, int, int)
+ void notifyListeners()
+ void removeListener(InformationStoreListener)
+ void removeState(int, int)
+ void removeTransition(int, int, int, int)
+ void toggleStateType(int, int)
+ void unhighlight(int)

```

```

DisplayStore

~ DisplayNode[][] displayGraph
~ int column
~ int row

+ boolean containsState(int, int)
+ boolean moveState(int, int, int, int)
+ int addState(int, int, int)
+ int getColumn(int)
+ int getColumns()
+ int getRow(int)
+ int getRows()
+ int getState(int, int)
+ int removeState(int, int)

```

```

FSMStore

- ArrayList<State> States
- ArrayList<Transition> Transitions
- int nextID

# ArrayList<State> getStates()
+ ArrayList<Transition> getTransitions()
+ State getState(int)
+ Transition getTransition(int)
+ boolean containsState(int)
+ boolean containsTransition(int)
+ boolean containsTransitionWithIDs(int, int)
+ boolean removeTransitionWithIDs(int, int)
+ int addState(String)
+ int addState(String, int)
+ int addTransition(String, int, int)
+ int getTransitionWithIDs(int, int)
+ int numStates()
+ int numTransitions()
+ void removeState(int)
+ void removeTransition(int)
+ void toggleStateType(int)
- Transition getTransitionPrivate(int, int)

```

```

DisplayNode

+ boolean hasState
+ int myID
+ boolean containsState()
+ int getID()
+ int removeNode()
+ int setNode(int)

```

```

State

# static final int ACCEPT
# static final int ALL
# static final int NEITHER
# static final int START
- String name
- boolean highlighted
- int id
- int type
# void highlight()
# void rename(String)
# void setType(int)
# void toggleState()
# void unhighlight()
+ String getName()
+ boolean isAccept()
+ boolean isHighlighted()
+ boolean isStart()
+ int getID()

```

```

Transition

- ArrayList<String> conditions
- int fromID
- int id
- int toID
# boolean containsCondition(String)
# void addCondition(String)
# void removeCondition(String)
+ String getLabel()
+ int getFromID()
+ int getID()
+ int getToID()
+ int numConditions()

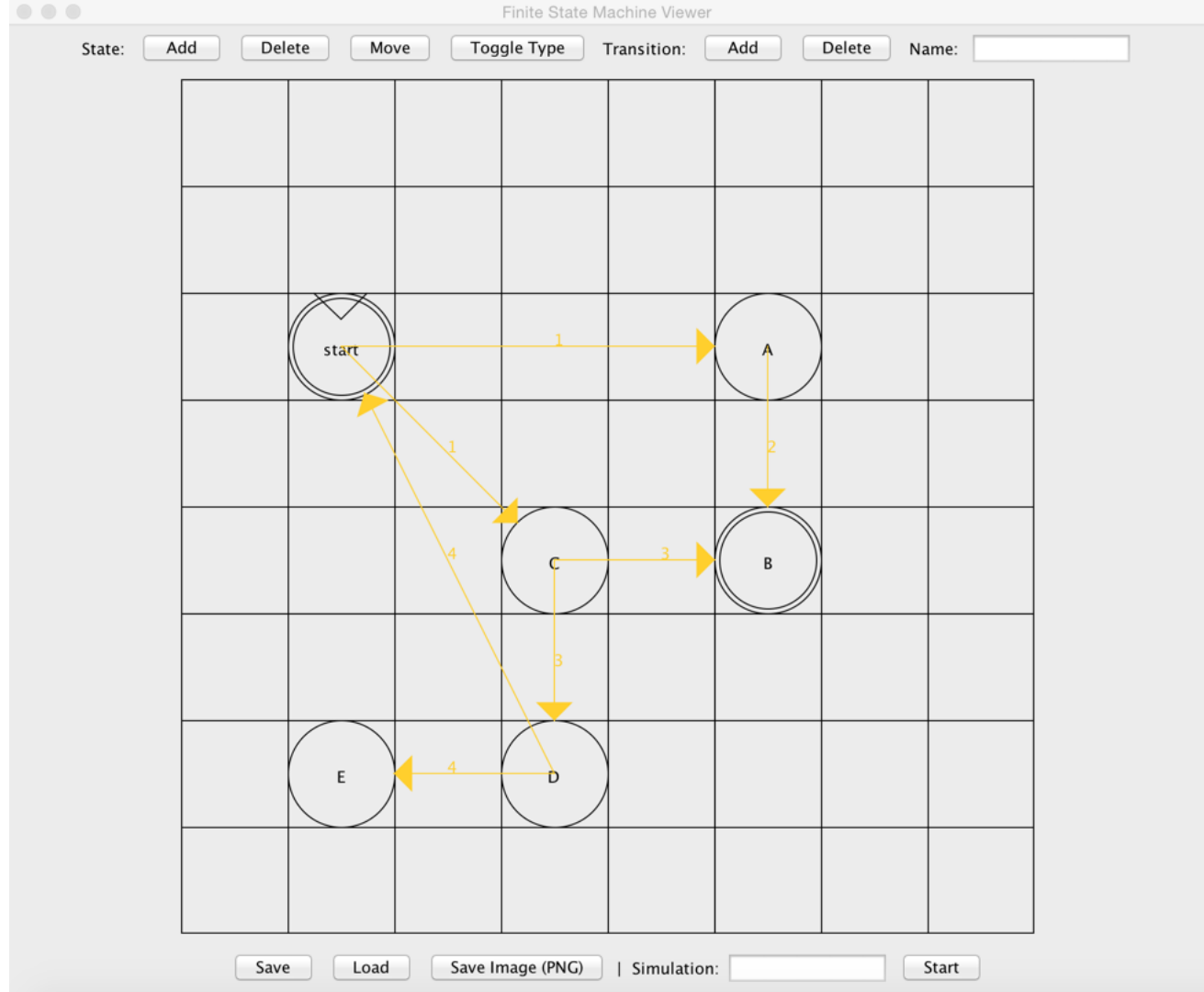
```

NEW DESIGN

# Storage Changes: Facade Pattern

(pulling the functionality into the model itself)

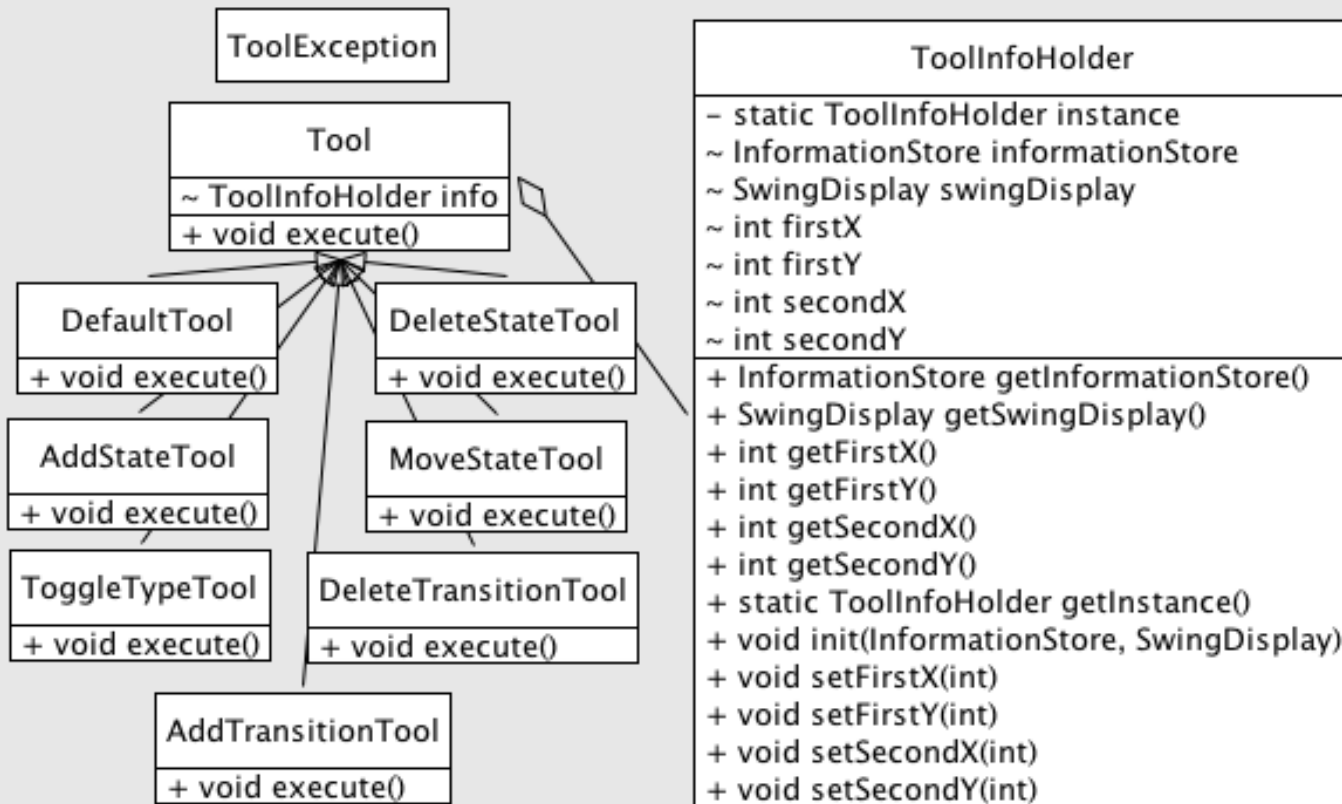
# User Interface Diagram



# Controller Changes:

## Tool Pattern

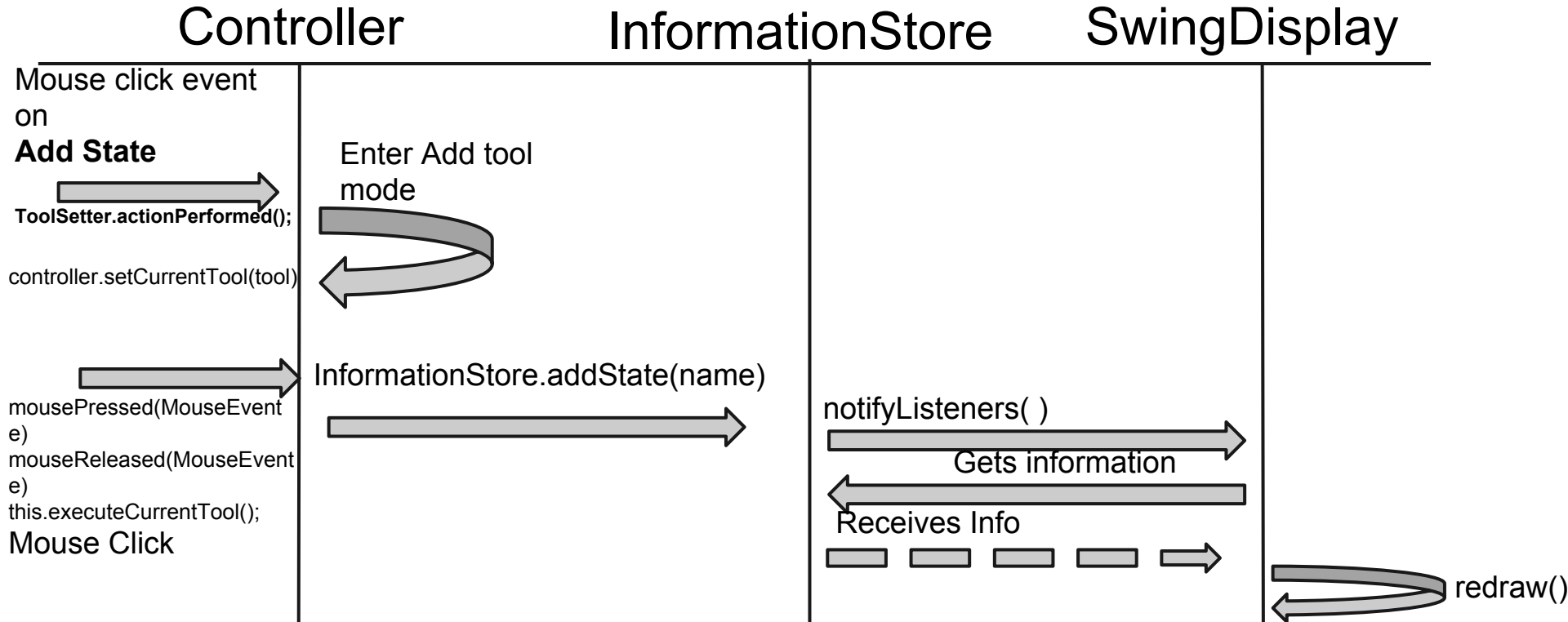
- Pulling out Tools to their own package
- ToolInfoHolder to pass along information to tools
  - now a singleton
    - cuts down on construction of tools vastly
- Tool: interface -> abstract class
  - tools only need the super constructor
    - ToolInfoHolder.getInstance() called in Tool class



**Tool Changes: New Package (edu.union.fsm.tool)**



# Click on Empty Location with add State Tool - Sequence Diagram



# Executing with Exception Handling

```
/**  
 * Executes the current Tool.  
 */  
private void executeCurrentTool() {  
    try {  
        currentTool.execute();  
    } catch (ToolException ex) {  
        debugger.stackDebug(ex);  
        String toPrint = ex.getMessage();  
        swingDisplay.displayErrorMessage(toPrint);  
    }  
}
```

# ActionListener Package

LoadBinButtonListener
<ul style="list-style-type: none"><li>- InformationStore informationStore</li><li>- SwingDisplay swingDisplay</li><li>~ SimulateButtonListener simulator</li></ul>
+ void actionPerformed(ActionEvent)

SaveBinButtonListener
<ul style="list-style-type: none"><li>- InformationStore informationStore</li><li>- SwingDisplay swingDisplay</li></ul>
+ void actionPerformed(ActionEvent)

SavePNGButtonListener
<ul style="list-style-type: none"><li>- SwingDisplay swingDisplay</li></ul>
+ void actionPerformed(ActionEvent)

SimulateButtonListener
<ul style="list-style-type: none"><li>- Debugger debugger</li><li>- InformationStore informationStore</li><li>- LinkedList&lt;Integer&gt; highlighted</li><li>- LinkedList&lt;String&gt; queue</li><li>- SwingDisplay swingDisplay</li><li>- boolean mode</li><li>- final boolean NEXT</li><li>- final boolean START</li></ul>
+ void actionPerformed(ActionEvent)
+ void clearSimulation()
- LinkedList<String> generateQueue()

# Saving and Loading

- Serializable: for saving and loading, forces users to save to .bin files.
- Image: saves png representation of the FSM
- Independent of FSM Program (reusable)
- allows for saving and loading anywhere using a file chooser.

# Saving and Loading (cont)

## SaveBin

- Object toSave
- JFrame toPrompt

+ void saveFile

## loadBin

- Object toLoad
- JFrame toPrompt

+ Object loadFile

## SavePNG

- JComponent toSave
- JFrame toPrompt

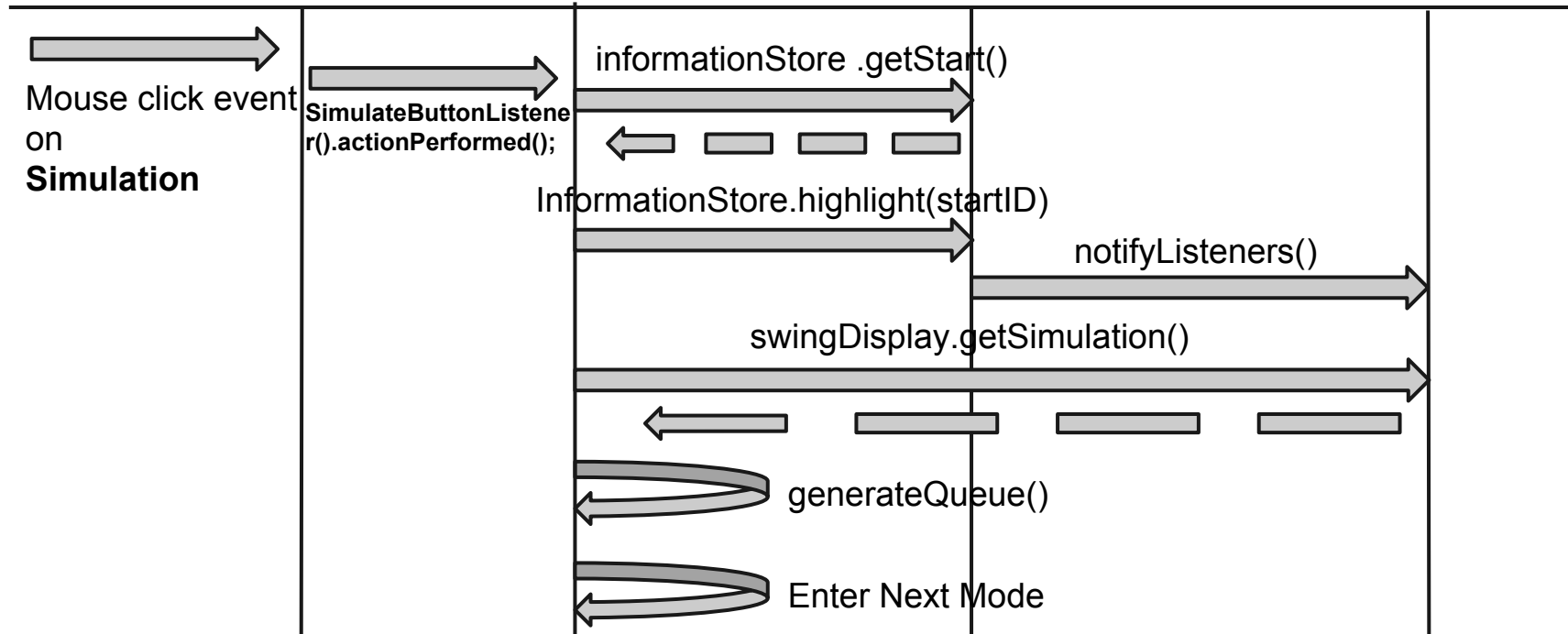
+ void saveFile

# Simulation

- ActionListener
  - on Start, parses the input into a queue, begins by highlighting the start state, then goes into NEXT mode
  - in next mode, it grabs the next transition to from the queue and gets the next states and highlights them
  - prompts using dialog boxes if any invalid traversal occurs or the traversal ends.

# Clicking Start Simulation - Sequence Diagram

Controller    SimulateButtonListener    InformationStore    SwingDisplay



# Debugging using a singleton

Debugger
<ul style="list-style-type: none"><li>- boolean debugEnabled</li><li>- static Debugger instance</li></ul>
<ul style="list-style-type: none"><li>+ static Debugger getInstance()</li><li>+ void setEnable(boolean)</li><li>+ void stackDebug(Exception)</li></ul>



```
/**  
 * Executes the current Tool.  
 */  
private void executeCurrentTool() {  
    try {  
        currentTool.execute();  
    } catch (ToolException ex) {  
        debugger.stackDebug(ex);  
        String toPrint = ex.getMessage();  
        swingDisplay.displayErrorMessage(toPrint);  
    }  
}
```

# Demo

YAY!

# Any Questions?