Project: Cell Simulator
Anthony, Quincy & Reed

Classes:

1. Cell:
   - Description: Represents a single cell in the grid with states such as edge, empty, live tree, burning tree, and burnt-down tree. Manages cell state updates based on neighboring cells.
   - Instance Variables:
     - state: Current state of the cell (e.g., edge, empty, live tree, burning tree, burnt-down tree).
     - burnTimeCounter: Tracks cycles left for a cell in a burning state.
   - Public Methods:
     - setState(newState): Sets the state of the cell.
     - updateState(neighbors, burnTime, spreadProbability): Updates cell state based on neighboring cells and given parameters (e.g., spread probability).
     - isBurning(): Returns true if the cell is in a burning state.
     - resetBurnTime(): Resets the burn time counter for newly ignited cells.

2. Grid:
   - Description: Manages a 2D array of Cell objects, initializes the grid based on parameters, and updates cell states each simulation step.
   - Instance Variables:
     - cells[][]: 2D array holding all Cell objects.
     - numRows: Number of rows in the grid.
     - numCols: Number of columns in the grid.
   - Public Methods:
     - initializeGrid(density, initialBurningTrees): Populates the grid with cells, setting initial live tree, burning tree cells based on density, and initial burn parameters.
     - getNeighbors(cell): Returns a list of neighboring cells for the specified cell.
     - updateGrid(burnTime, spreadProbability): Updates each cell based on the current grid state, applying the simulation rules.
     - resetGrid(): Resets all cells to their initial states based on density and initial burning trees.

3. SimulationController:
   - Description: Acts as the controller, managing simulation parameters, controlling simulation steps, and communicating between the grid and the UI.
   - Instance Variables:
     - grid: The grid object containing the cells.
     - burnTime, spreadProbability, forestDensity, initialBurningTrees: Simulation parameters.
     - running: Boolean to track if the simulation is currently running.
   - Public Methods:
     - startSimulation(): Begins the simulation loop.
     - pauseSimulation(): Pauses the simulation.
     - stepSimulation(): Advances the simulation by one step, calling updateGrid.
     - resetSimulation(): Resets the simulation, reinitializing the grid and parameters.
     - setParameter(parameterName, value): Updates the specified parameter for the simulation.

4. WildFireView:
   - Description: Handles the graphical user interface, displaying the grid, controls for setting parameters, and buttons for interacting with the simulation.
   - Instance Variables:
     - gridDisplay: JavaFX node for displaying cells in the grid.
     - startButton, pauseButton, stepButton, resetButton: Buttons for controlling the simulation.
     - inputFields: UI fields for setting burn time, spread probability, forest density, and grid dimensions.
   - Public Methods:
     - initializeUI(): Sets up the GUI layout, adding buttons, input fields, and grid display.
     - updateDisplay(): Refreshes the display based on the current grid state.
     - addEventListeners(SimulationController): Links buttons to appropriate controller methods.

Flow of Control:

Button	Method(s) called	Description
Start	SimulationController.startSimulation()	Starts the simulation loop, updating the grid at set intervals.
Pause	SimulationController.pauseSimulation()	Pauses the simulation loop, halting automatic updates.
Step	SimulationController.stepSimulation()	Advances the simulation by one step, calling updateGrid and refreshing the view.
Reset	SimulationController.resetSimulation()	Resets all cells in the grid based on initial parameters, re-initializes with specified density, etc.
Set Parameters	SimulationController.setParameter(parameterName, value)	Updates the simulation parameter based on user input for burn time, spread probability, etc.

Member Responsibility:

Teammate	Assigned Classes	Responsibilities
Anthony	Cell, Grid	Implement Cell and Grid classes, define state update logic based on neighbor rules and parameters.
Quincy	SimulationController	Handle simulation logic, manage states, parameters, and connect grid updates with the view.
Reed	WildFireView	Design and implement GUI, connect user actions to the controller, ensure dynamic display of cell states.

Timeline:

- **Nov 7**: Complete project planning document, defining classes, methods, flow control, and responsibilities.
- **Nov 8–9**: Member 1 begins implementing Cell and Grid with methods for state updates and initialization.
- **Nov 10**: Member 2 completes initial implementation of SimulationController, including methods to handle simulation steps and parameters.
- **Nov 11**: Member 3 finalizes WildFireView setup and connects GUI elements to SimulationController.
- **Nov 12**: Integrate all components, ensuring correct interaction between view, model, and controller. Begin testing.
- **Nov 13**: Complete final testing and debugging, verify edge cases, and prepare for the project demo.