George‘s Module Documentation

MountainAgent Class:

- **Overview**:
  - For the given number of mountains, the agents finds an acceptable start point on land. It then raises that point and points around it to create a cone/mountains-like shape then moves to the neighboring point in the given direction. It turns randomly and also creates foothills, which are just like mountains but cannot create foothills of their own, randomly based on the given parameters. All altered points have their biome labelled as ‘mountain’.
- **Methods**:
  - **generate(map)**:
    - **Purpose**: runs the mountain agent on the given map
  - **pickRandomStart(map):**
    - **Purpose:** picks a new point on the map to start the agent
  - **checkPoint(map, x, y):**
    - **Purpose:** checks if a point is valid on the given map
    - **Unit Tests:**
      - **Equivalence Classes:**
        - X and y a in the bounds of the map object
        - X and y are not within the bounds
      - **Test Case 1:** X and y a in the bounds of the map object
        - Expected Output: returns the point object at (x, y)
      - **Test Case 1:** X and y are not within the bounds
        - Expected Output: returns ‘null’
  - **checkStart(map):**
    - **Purpose:** checks if the area around the agent is big enough to start
    - **Unit Tests:**
      - **Equivalence Classes:**
        - The area is sufficient to start
        - Agent is too close to the ocean or the edge to start
      - **Test Case 1:** The area is sufficient to start
        - Expected Output: returns false
      - **Test Case 1:** Agent is too close to the ocean or the edge to start
        - Expected Output: returns true
  - **elevateCircle(map):**
    - **Purpose:** raises the points around the agent to form a mountain
    - **Unit Tests:**
      - **Equivalence Classes:**
        - Agent is within range of an ocean
        - Agent not within range of an ocean
      - **Test Case 1:** Agent is within range of an ocean
        - Expected Output: agent turns away from the ocean

- **Test Case 1:** Agent not within range of an ocean
    - Expected Output: agent continues in given direction

- **elevateFoothillCircle(map, x, y, height, width):**
    - **Purpose:** raises the points around the agent to form a foothill
    - **Unit Tests:**
        - **Equivalence Classes:**
            - Foothill is within range of an ocean
            - Foothill not within range of an ocean
        - **Test Case 1:** Foothill is within range of an ocean
            - Expected Output: foothill turns away from ocean
        - **Test Case 1:** Foothill not within range of an ocean
            - Expected Output: foothill continues in given direction
- **newHeightWithDropoff(centerX, centerY, x, y, height):**
    - **Purpose:** calculates the new height for a point based on its distance from the ridgeline
- **newSetHeight(map, x, y, height):**
    - **Purpose:** sets a point to its new height based on newHeightWithDropoff()
    - **Unit Tests:**
        - **Equivalence Classes:**
            - New height is higher than old
            - New height is lower or equal to old
        - **Test Case 1:** New height is higher than old
            - Expected Output: height of point is set to the new height
        - **Test Case 1:** New height is lower or equal to old
            - Expected Output: height of point is not changed
- **makeFoothills(map):**
    - **Purpose:** creates foothill agents and runs them at the agent's current position
- **runFoothillAgent(map, height, dir, length):**
    - **Purpose:** runs the foothill agent with given parameters
- **turnFoothill(map, direction, x, y, turn):**
    - **Purpose:** changes the foothill agents direction to a new one based on the value of turn
    - **Unit Tests:**
        - **Equivalence Classes:**
            - 0 <= Direction + turn <= 360
            - Direction + turn < 0
            - Direction + turn > 360
        - **Test Case 1:** 0 <= Direction + turn <= 360
            - Expected Output: returns direction + turn
        - **Test Case 2:** Direction + turn < 0
            - Expected Output: new direction bounded by 0, 360
        - **Test Case 3:** Direction + turn > 360

            - Expected Output: new direction bounded by 0, 360
    - **newTurnLeft(hard):**
        - **Purpose:** turns the agent left a random amount between min and max values
        - **Unit Tests:**
            - **Equivalence Classes:**
                - Hard is true
                - Hard is false
            - **Test Case 1:** hard is true
                - Expected Output: direction is set to a new value on the upper half of the range between min and max turn
            - **Test Case 1:** hard is false
                - Expected Output: direction is set to a new value between min and max turn
    - **newTurnRight(hard):**
        - **Purpose:** turns the agent right a random amount between min and max values
        - **Unit Tests:**
            - **Equivalence Classes:**
                - Hard is true
                - Hard is false
            - **Test Case 1:** hard is true
                - Expected Output: direction is set to a new value on the upper half of the range between min and max turn
            - **Test Case 1:** hard is false
                - Expected Output: direction is set to a new value between min and max turn
    - **newRotateAgent(amount):**
        - **Purpose:** changes the agents direction to a new one based on the amount
    - **newGetTurn():**
        - **Purpose:** returns a positive or negative value between min and max turn to use to turn the agent left or right
    - **newMoveAgent():**
        - **Purpose:** moves the agent 1 unit in the given direction
    - **distance(x1, y1, x2, y2):**
        - **Purpose:** returns the distance between two points

Main Class:

- **Overview**:
    - Main function that is called to create and display the map. It recieves the user parameters and extracts all necessary information from them. Then it creates all

the agents and runs them sequentially. Then it colors the heightmap based on elevation and biomes and displays the image to the user.

- **Methods**:
  - **determineColor(map, raw, col, x, y)**:
    - **Purpose**: determines what color the point should be based on its biome-raw and its surrounding points
    - **Unit Tests:**
      - **Equivalence Classes:**
        - Raw is ocean, lake, river, beach, shore or tall shore
        - Raw is coast, mountain, or ridge
      - **Test Case 1:** Raw is ocean, lake, river, beach, shore or tall shore
        - Expected Output: preset color is returned for each given biome
      - **Test Case 1:** Raw is coast, mountain, or ridge
        - Expected Output: color is returned based on elevation and shaded to show depth
    -
  - **scorePoint(map, x, y):**
    - **Purpose**: returns a score for a point based on a convolution matrix to used for shading
    - **Unit Tests:**
      - **Equivalence Classes:**
        - All surrounding points are in bounds of the map
        - Not all surrounding points are in bounds of the map
      - **Test Case 1:** All surrounding points are in bounds of the map
        - Expected Output: score based on the convolution matrix
      - **Test Case 1:** Not all surrounding points are in bounds of the map
        - Expected Output: score based on convolution matrix omitting the points that are outside of the map
  - **smoothPoint(map, x, y):**
    - **Purpose**: smooths the height of the given point
    - **Unit Tests:**
      - **Equivalence Classes:**
        - All surrounding points are in bounds of the map
        - Not all surrounding points are in bounds of the map
      - **Test Case 1:** All surrounding points are in bounds of the map
        - Expected Output: new height based on all surrounding points
      - **Test Case 1:** Not all surrounding points are in bounds of the map
        - Expected Output: new height based on valid surrounding points