Project 5 Deliverables

Project Summary

- **Project Name:** Love Island Game Show Simulator
- **Team:** Alexis Davidson, Tiara Gibson, and Khushi Jani
- What is the high-level overview of your semester project? What are you trying to accomplish?
    - This simulation is a reality game show where contestants ("islanders") interact, participate in challenges, and earn scores based on their performance and behaviors.
- What will your system do when you are done?
    - The simulation aims to showcase various design patterns by simulating real life game scenarios, updating scores, tracking behaviors, and displaying results.

Project Requirements

Functional Requirements:

- User interfaces – viewing scores, navigating between screens, and imitating game actions
- Challenges – for islanders to compete in; different types of challenges and outcomes
- Behavioral strategies – influence how islanders participate in challenges
- Scoring and Tracking – scores are updated based on islander performance
- End-of-Game Reports – display final scores and a summary of islander actions

Non-Functional Requirements:

- Performance – simulation should operate without noticeable delays, using efficient data storage
- Usability – interface should be intuitive, with easy navigation
- Extensibility – design should allow for the addition of new challenges and islander roles

Users and Tasks: Use Cases

User Types

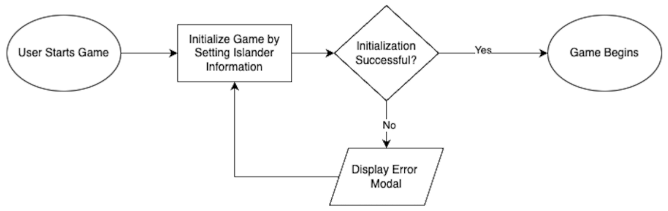
The system will have a single user type that will have tasks to accomplish, which is the “Game Manager”. The Game Manager is responsible for setting up the game and the islanders, initiating challenges, viewing scores, and ending the game.

User Tasks

There are 3 main user tasks as a part of the system: initializing the game, assigning challenges, and displaying the score report.

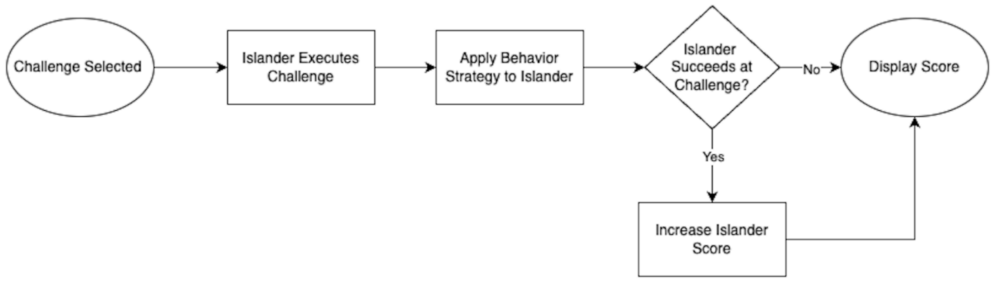
Initialize Game

The GameManager (user) starts the game by setting up the islanders, which involves selecting each islander’s roles and strategy type. If the GameManager attempts to start the game and an error occurs (i.e. there is invalid or missing data for the islanders), then an error message is displayed.


User Starts Game
Initialize Game by Setting Islander Information
Initialization Successful?
Yes
Game Begins
No
Display Error Modal

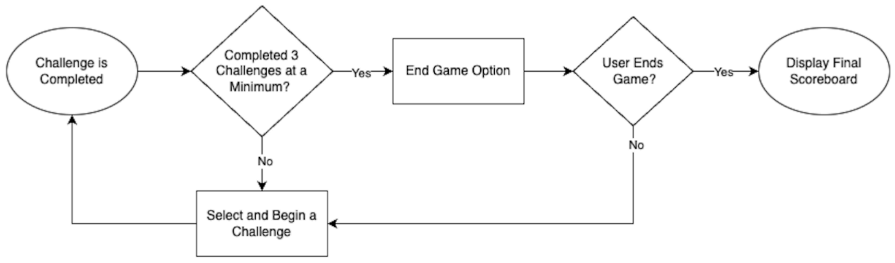
Assign Challenge

The GameManager (user) assigns a challenge to all islanders as a part of the game simulation, and this triggers different behaviors for each islander based on their individual strategies. An islander’s probability of successfully completing a challenge is based on their roles and strategies that are set at initialization. If an islander successfully completes a challenge, then that islander’s score is increased.

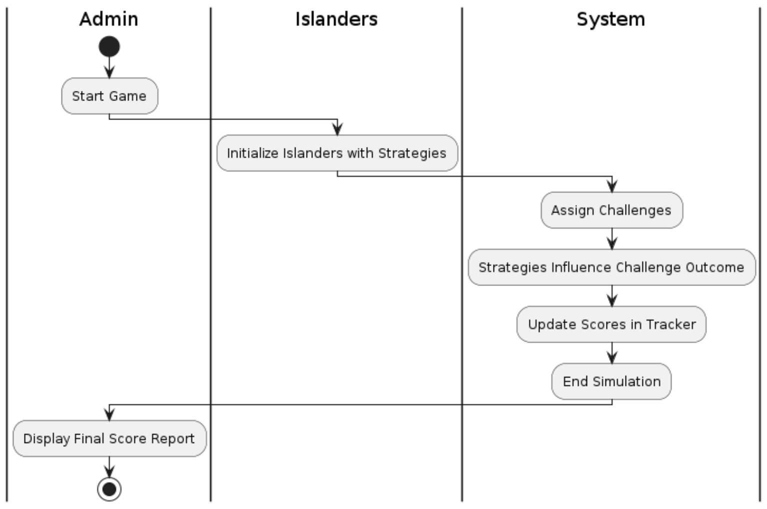

Challenge Selected
Islander Executes Challenge
Apply Behavior Strategy to Islander
Islander Succeeds at Challenge?
No
Display Score
Yes
Increase Islander Score

Display Score Report

When at least 3 challenges have been completed, the Game Manager will have the opportunity to end the game. When the game ends, then the score report is displayed for all islanders.



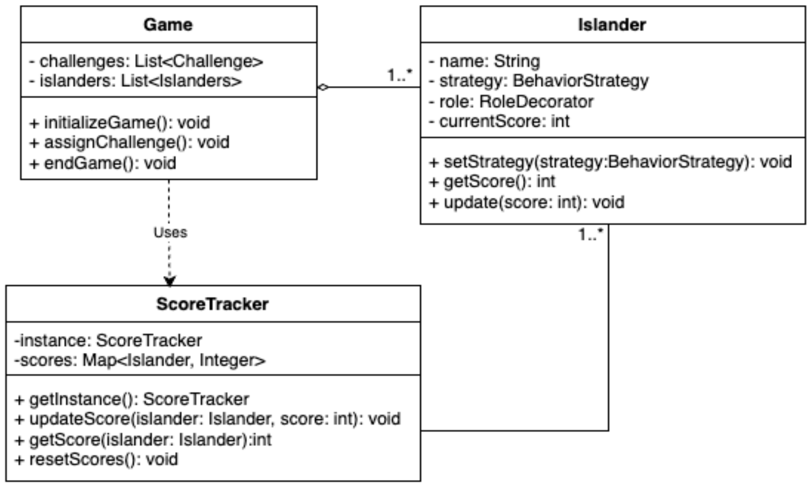
UML Activity Diagram


Admin
Islanders
System
Start Game
Initialize Islanders with Strategies
Assign Challenges
Strategies Influence Challenge Outcome
Update Scores in Tracker
End Simulation
Display Final Score Report

Data Storage

The application does not require persistence storage because of the requirement of a single simulation occurring at a time. While a game is in progress, the Islander information, including names, strategies, roles, and current scores will be maintained temporarily in memory using Java’s in-memory data structures.

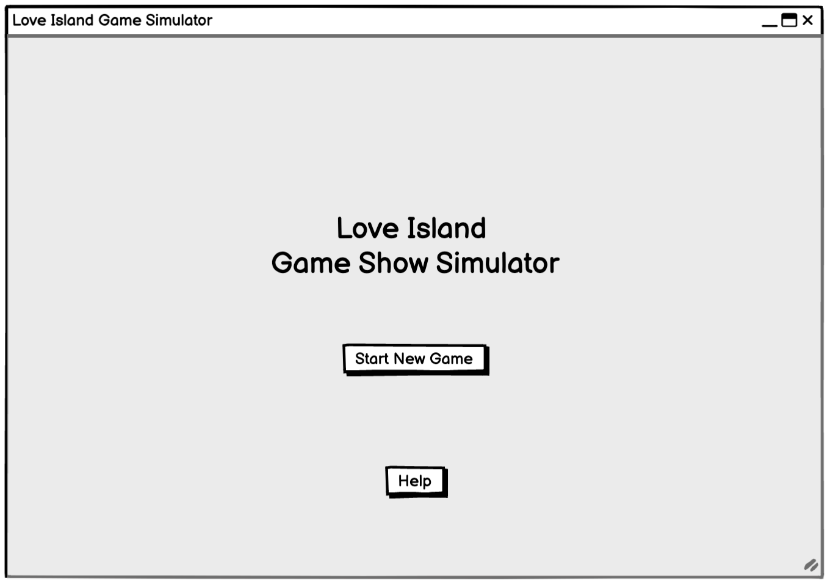
ER Diagram for In-Memory Storage


Game
- challenges: List<Challenge>
- islanders: List<Islanders>
+ initializeGame(): void
+ assignChallenge(): void
+ endGame(): void
1..*
Islander
- name: String
- strategy: BehaviorStrategy
- role: RoleDecorator
- currentScore: int
+ setStrategy(strategy:BehaviorStrategy): void
+ getScore(): int
+ update(score: int): void
Uses
1..*
ScoreTracker
-instance: ScoreTracker
-scores: Map<Islander, Integer>
+ getInstance(): ScoreTracker
+ updateScore(islander: Islander, score: int): void
+ getScore(islander: Islander):int
+ resetScores(): void

UI Mockups/Sketches

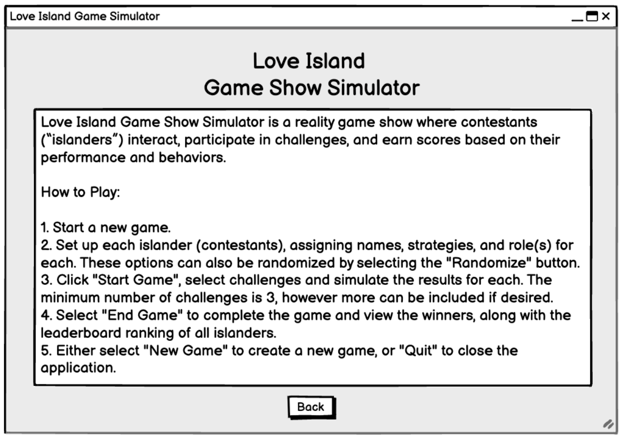
Game Initialization

The application gives the user (GameManager) the ability to start a new game, or to view instructions via the “Help” button.



User Selected “Help”

When a user clicks the “Help” button, a description of the game as well as a tutorial of how to play the game is displayed.



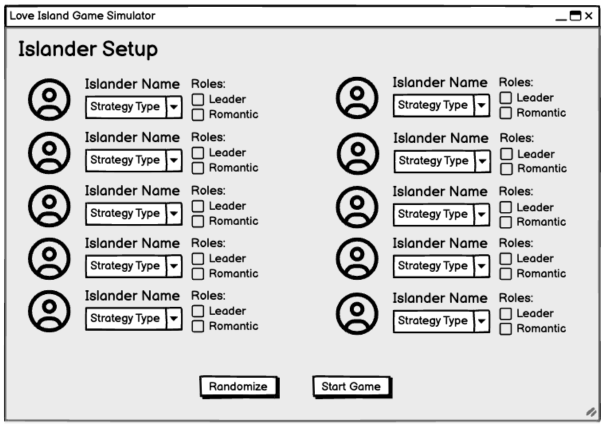
Love Island Game Show Simulator is a reality game show where contestants ("islanders") interact, participate in challenges, and earn scores based on their performance and behaviors.

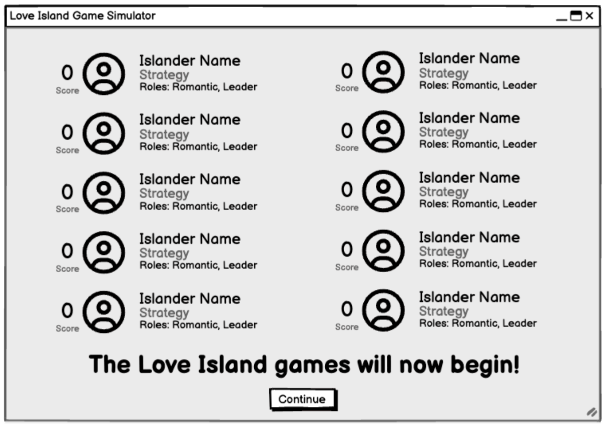
How to Play:

1. Start a new game.
2. Set up each islander (contestants), assigning names, strategies, and role(s) for each. These options can also be randomized by selecting the "Randomize" button.
3. Click "Start Game", select challenges and simulate the results for each. The minimum number of challenges is 3, however more can be included if desired.
4. Select "End Game" to complete the game and view the winners, along with the leaderboard ranking of all islanders.
5. Either select "New Game" to create a new game, or "Quit" to close the application.

User Selects "Start New Game"

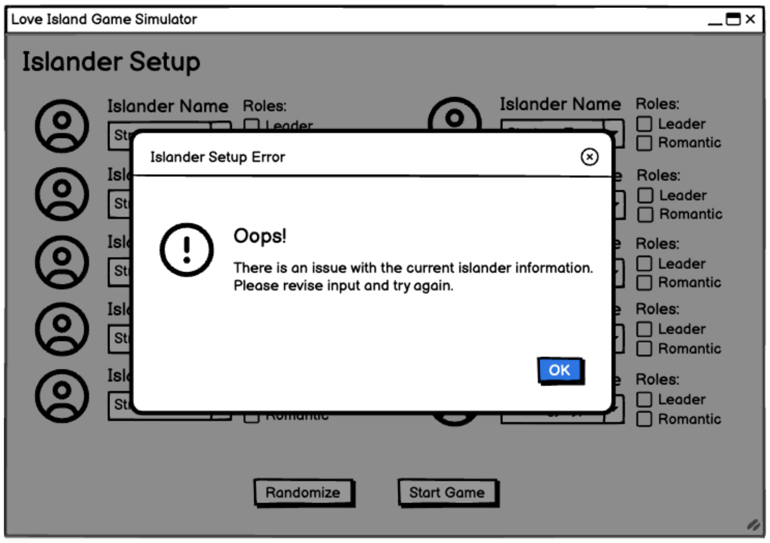
When a new game is started, the user is met with a screen that allows them to customize all 10 islanders (contestants) that will be participants in the game. Each islander will have a unique name, a strategy type selected from predetermined options, and optional roles to assign. When "Start Game" is selected with valid data for all islanders, the user is notified of a successful submission and the game begins.




Love Island Game Simulator
0
Score
Islander Name
Strategy
Roles: Romantic, Leader
The Love Island games will now begin!
Continue

Game Initialization - Alternate Scenario (Error)

If there is an issue with the data entered during the islander setup, a modal containing an alert informing the user of an error is displayed.

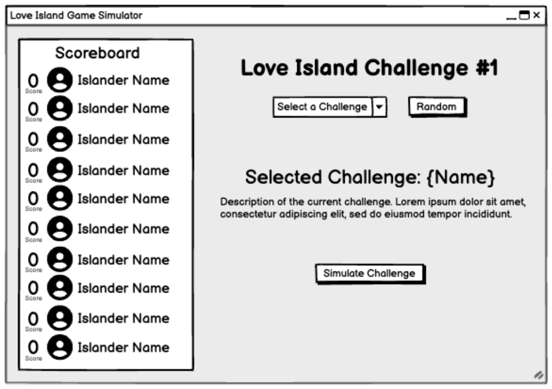


Oops!

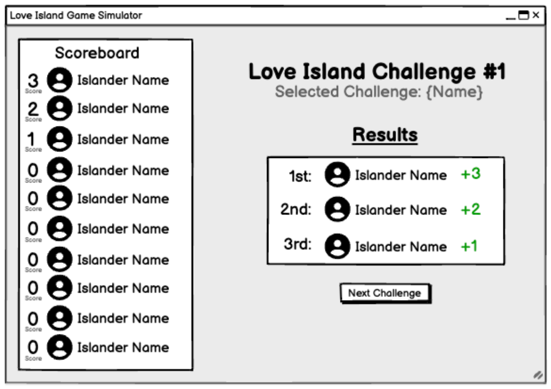
There is an issue with the current islander information. Please revise input and try again.

Challenge Assignment and Execution

During the simulation, the user will be able to either select the challenge they would like to have the islanders participate in or choose a random one. The selected challenge name and description is displayed, and the user can confirm the challenge to simulate. When the challenge is simulated, the results are displayed containing the top 3 islanders for that challenge, and the scoreboard is updated to reflect the cumulative scores for all islanders.

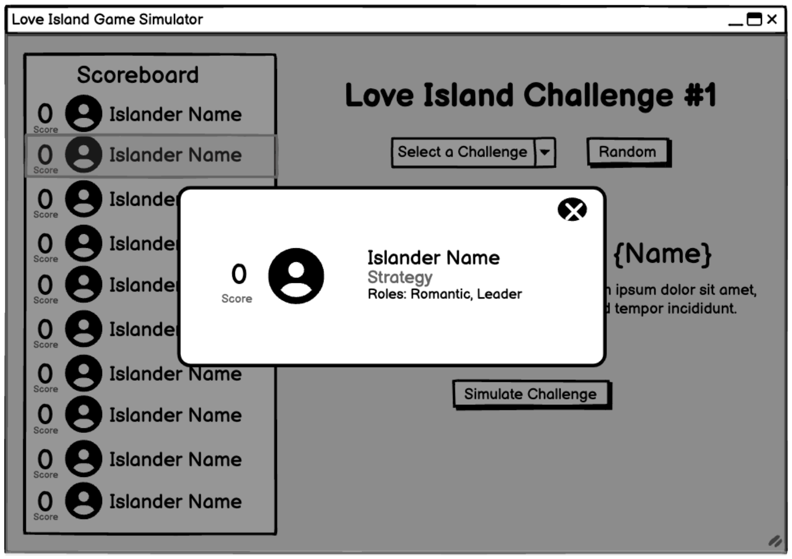


Description of the current challenge. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt.



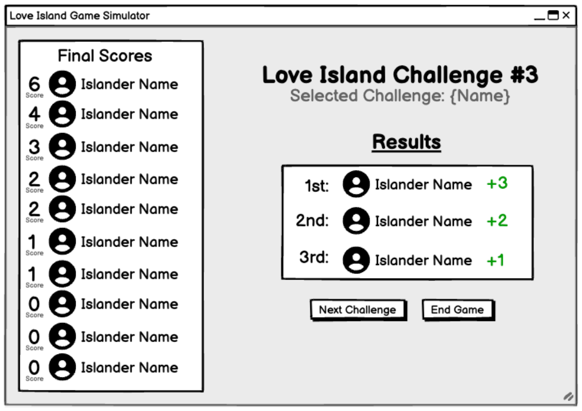
Viewing Islander Info During Simulation

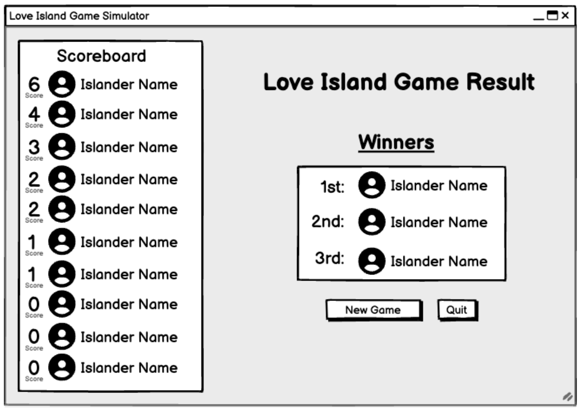
When a user clicks an islander during the simulation, a modal containing all relevant islander information will display.



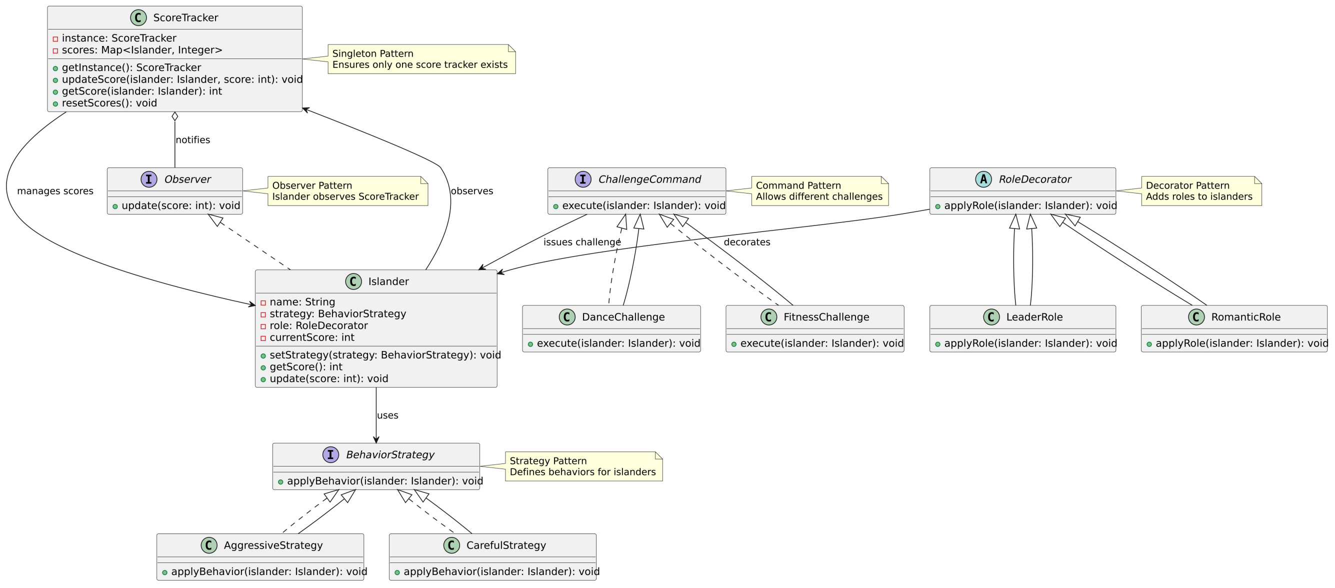
Final Challenge Completion and Game Results

Once 3 challenges are simulated, the user can either assign another challenge to islanders or end the game. If the game is ended, then the top 3 islanders are displayed with their total scores, and the scoreboard of all participants and their cumulative scores are visible.





UML Class Diagram & Pattern Use


ScoreTracker
instance: ScoreTracker
scores: Map<Islander, Integer>
getInstance(): ScoreTracker
updateScore(islander: Islander, score: int): void
getScore(islander: Islander): int
resetScores(): void
Singleton Pattern
Ensures only one score tracker exists
notifies
manages scores
observes
Observer
update(score: int): void
Observer Pattern
Islander observes ScoreTracker
ChallengeCommand
execute(islander: Islander): void
Command Pattern
Allows different challenges
RoleDecorator
applyRole(islander: Islander): void
Decorator Pattern
Adds roles to islanders
issues challenge
decorates
Islander
name: String
strategy: BehaviorStrategy
role: RoleDecorator
currentScore: int
setStrategy(strategy: BehaviorStrategy): void
getScore(): int
update(score: int): void
DanceChallenge
execute(islander: Islander): void
FitnessChallenge
execute(islander: Islander): void
LeaderRole
applyRole(islander: Islander): void
RomanticRole
applyRole(islander: Islander): void
uses
BehaviorStrategy
applyBehavior(islander: Islander): void
Strategy Pattern
Defines behaviors for islanders
AggressiveStrategy
applyBehavior(islander: Islander): void
CarefulStrategy
applyBehavior(islander: Islander): void

User Interactions/UML Sequence Diagrams

Primary User Interactions

The three primary user interactions in the application are the following:

1. Initializing the game
2. Assigning challenges to islanders
3. Updating and displaying the score tracker and final scores.

Initializing the Game

1. When the program is executed, the GameManager will start a new game and create 10 islanders that are initialized with input names and will set the current score to 0.
2. The GameManager will get input from the player to assign a selection of BehaviorStrategy for each islander.
3. GameManager will get input from the player to allow them to optionally assign role(s) to islanders.
4. Once the setup to instantiate and initialize islanders with player input is completed by the GameManager, it will display a message to display all the islanders' information: name, current score, strategy, and role(s), and announce that "The Love Island games will now begin."
5. ScoresTracker is updated to contain the initial scores in Map<Islander, Integer>.

Assigning Challenges to Islanders

1. GameManager will call assignChallenge() on a ChallengeCommand.
2. ChallengeCommand will iterate each islander and call executeChallenge().
3. Each Islander applies the strategy and roles assigned to them to complete the challenge, which impacts their overall performance in the challenge and their scores.
4. ScoreTracker updates each islander's score via updateScore() based on their challenge performance.

Updating & Displaying the Score Tracker & Final Scores.

1. After each challenge, GameManger requests the score report by calling displayScoreReport() on ScoreTracker.
2. ScoreTracker displays the formatted scores for each islander.

After the 3 challenges are completed, the player can choose to issue more challenges or terminate the game and get the final results. When the player chooses to terminate the game, the program will display the top 3 scoring islanders, along with a list of the remaining islanders and their scores.