Integration Tests

Test Performed	Type	Date
Statistics system integration test 1	individual	24 Oct 2013

This integration phase uses the bottom-up strategy, because most crucial classes are the low-level ones that read and write to the file. Integration involves back-end CSV file reader and writer, back-end statistics data logic, and front-end GUI for displaying statistics in a table format.

Tested flow of data:
1. Passed: Unit tests write data for mock players info and mock games info is passed to stats logic
2. Passed: Stats data logic takes mock info and returns statistics between mock players
3. Passed: Data is stored in pair_records.csv and player_records.csv
4. Passed: Data is retrieved from pair_records.csv and player_records.csv
5. Passed: Data is displayed as a table of statistics in GUI panel

User Management (Login) system integration test 1	individual	24 Oct 2013

This integration phase uses bottom-up Test Driven Development strategy, because most crucial classes are the low-level ones that read and write to the file. Integration involves back-end CSV file reader and writer, back-end user management logic, and front-end for prompting users to log-in/log-out/register.

Tested flow of data:
1. Passed: GUI panel for user management is loaded with 2 sets of fields and buttons
2. Passed: Mock non-existent user info (username) is used to attempt login and error message confirms user is not found in user_data.csv
3. Passed: Mock user info (username and password) is used to register a new user
4. Passed: Mock user info is written to user_data.csv
5. Passed: Mock user info is entered and used to log-in; message confirms successful login
6. Passed: Same mock username is used to attempt registration and error message confirms user already exists in user_data.csv
7. Passed: Steps 3-5 are repeated with a different mock user (username)
8. After both players are logged-in, a button in the GUI becomes active

Main Game system integration test 1	individual	25 Oct 2013

This integration phase uses the top-down strategy, because the overall architecture is determined first and then broken down into smaller sub-classes. Integration involves front-end main game panel, front-end side info panel, back-end game logic, and back-end game objects.

Tested flow of data:
1. Passed: Game is initialized and a blank grid is displayed in GUI panel for new round
2. Passed: Player presses specified key on keyboard and round begins

3. Passed: Player presses specified keys to control movement of racer inside GUI grid
4. Passed: Player actions are mapped onto back-end grid logic
5. Passed: Player hits a wall in GUI grid
6. Passed: Collision is detected in back-end game logic
7. Passed: Round is terminated and message appears confirming game over
8. Passed: Side info panel updates the number of rounds played
9. Passed: Steps 2-8 are repeated for a total of 3 rounds

Map Selection integration test 1	individual	Oct 25 2013

This integration phase uses the bottom-up strategy, because most crucial classes are the low-level ones that read from loaded custom map files. Integration involves back-end TXT file reader, back-end map (grid) creation logic, and front-end map selector GUI.

Tested flow of data:

1. Passed: GUI with 3 radio buttons for default map selection and 1 button for file browser are loaded
2. Passed: Visual previews of 3 default maps are pre-loaded from files (map1.txt, map2.txt, and map3.txt) in the background
3. Passed: User clicks on radio button to select a map
4. Passed: GUI displays preview grid for selected map
5. Passed: User clicks OK to confirm selection
6. Passed: Grid creation logic returns mapped grid corresponding to selected file
7. Passed: Steps 3-6 are repeated for each of the 3 default maps
8. Passed: Repeat steps 1 and 2
9. Passed: User clicks on file browser button and selects a supported TXT file from disk
10. Passed: Visual previews of custom map is loaded from TXT file and preview grid is displayed
11. Passed: Same as step 6

Main Game system integration test 2	individual	11 Nov 2013

This integration phase uses the top-down strategy, because the overall architecture is determined first and then broken down into smaller sub-classes. Integration involves front-end main game panel, front-end side info panel, back-end game logic, and back-end game objects.

Tested flow of data:

1. Passed: Game is initialized and a blank grid is displayed in GUI panel for new round
2. Passed: Either player presses specified key on keyboard and round begins
3. Passed: Both players press specified keys to control movement of racers inside GUI grid
4. Passed: Player actions are mapped onto back-end grid logic
5. Passed: Either player hits a wall in GUI grid
6. Passed: Collision is detected in back-end game logic
7. Passed: Back-end logic determines the winner and loser of the round (or a tie)

8. Passed: Round is terminated and message appears confirming game over and winner
9. Passed: Side info panel updates the number of rounds played and the ratio of wins/losses between players
10. Passed: Steps 2-8 are repeated for a total of 3 rounds (and for each combination of winner/loser/tie)

Statistics system integration test 2	individual	13 Nov 2013

This integration phase uses the bottom-up strategy, because most crucial classes are the low-level ones that read and write to the file. Integration involves back-end CSV file reader and writer, back-end statistics data logic, and front-end GUI for displaying statistics in a table format.

Tested flow of data:

1. Passed: Unit tests write data for mock players info and mock games info in the format returned by the main game logic is passed to stats logic
2. Passed: Stats data logic takes info and returns statistics between players
3. Passed: Data is stored in pair_records.csv and player_records.csv
4. Passed: Data is retrieved from pair_records.csv and player_records.csv
5. Passed: Data is displayed as a table of statistics and a pair score in GUI panel

User Management (Login) system integration test 2	individual	14 Nov 2013

This integration phase uses bottom-up Test Driven Development strategy, because most crucial classes are the low-level ones that read and write to the file. Integration involves back-end CSV file reader and writer, back-end user management logic, and front-end for prompting users to log-in/log-out/register.

Tested flow of data:

1. Passed: GUI panel for user management is loaded with 2 sets of fields and buttons
2. Passed: Non-existent user info (username) is used to attempt login and error message confirms user is not found in user_data.csv
3. Passed: User info with invalid username is entered to attempt registration and error message confirms username not accepted
4. Passed: User info with valid username and invalid password are entered to attempt registration and error message confirms password does not meet requirements
5. Passed: Valid user info (username and password) is used to register a new user
6. Passed: User info is written to user_data.csv
7. Passed: Same username is used to attempt registration and error message confirms user already exists in user_data.csv
8. Passed: User info is entered and used to log-in as Player1; message confirms successful login
9. Passed: Same user info is entered and used as an attempt to log-in as Player2 and error message confirms that the user is already logged-in
10. Passed: User is logged-out of Player1 account by pressing logout button
11. Passed: Step 7 is repeated, but for Player2

12. Passed: Steps 2-11 are repeated with a different user (username)
13. Passed: After both players are logged-in, a button in the GUI becomes active

Main Game and Statistics integration test	collaborative	15 Nov 2013

This integration phase links the Main Game and Statistics segments. They are linked through the back-end, whereas their front-end GUIs are independent.

Tested flow of data:

1. Passed: Data is reset in pair_records.csv and player_records.csv
2. Passed: Game is initialized and a blank grid is displayed in GUI panel for new round
3. Passed: Either player presses specified key on keyboard and round begins
4. Passed: Both players press specified keys to control movement of racers inside GUI grid
5. Passed: Player actions are mapped onto back-end grid logic
6. Passed: Either player hits a wall in GUI grid
7. Passed: Collision is detected in back-end game logic
8. Passed: Back-end logic determines the winner and loser of the round (or a tie)
9. Passed: Round is terminated and message appears confirming game over and winner
10. Passed: Side info panel updates the number of rounds played and the ratio of wins/losses between players
11. Passed: Steps 2-10 are repeated for a total of 3 rounds (and for each combination of winner/loser/tie)
12. Passed: Game logic passes players info and games info to stats logic
13. Passed: Stats data logic takes info and returns statistics between players
14. Passed: Data is stored in pair_records.csv and player_records.csv
15. Passed: Data is retrieved from pair_records.csv and player_records.csv
16. Passed: Data is displayed as a table of statistics and a pair score in GUI panel

Main Game, Statistics, and Login integration test	collaborative	Nov 16 2013

This integration phase links the Login, Main Game, and Statistics segments. They are linked through the back-end, whereas their front-end GUIs are independent.

1. Passed: GUI panel for user management is loaded with 2 sets of fields and buttons
2. Passed: Non-existent user info (username) is used to attempt login and error message confirms user is not found in user_data.csv
3. Passed: User info with invalid username is entered to attempt registration and error message confirms username not accepted
4. Passed: User info with valid username and invalid password are entered to attempt registration and error message confirms password does not meet requirements
5. Passed: Valid user info (username and password) is used to register a new user
6. Passed: User info is written to user_data.csv
7. Passed: Same username is used to attempt registration and error message confirms user already

exists in user_data.csv

8. Passed: User info is entered and used to log-in as Player1; message confirms successful login
9. Passed: Same user info is entered and used as an attempt to log-in as Player2 and error message confirms that the user is already logged-in
10. Passed: User is logged-out of Player1 account by pressing logout button
11. Passed: Step 7 is repeated, but for Player2
12. Passed: Steps 2-11 are repeated with a different user (username)
13. Passed: After both players are logged-in, a button in the GUI becomes active
14. Passed: By clicking active button, the game is initialized and a blank grid is displayed in GUI panel for new round
15. Passed: Either player presses specified key on keyboard and round begins
16. Passed: Both players press specified keys to control movement of racers inside GUI grid
17. Passed: Player actions are mapped onto back-end grid logic
18. Passed: Either player hits a wall in GUI grid
19. Passed: Collision is detected in back-end game logic
20. Passed: Back-end logic determines the winner and loser of the round (or a tie)
21. Passed: Round is terminated and message appears confirming game over and winner
22. Passed: Side info panel updates the number of rounds played and the ratio of wins/losses between players
23. Passed: Steps 15-22 are repeated for a total of 3 rounds (and for each combination of winner/loser/tie)
24. Passed: Game logic passes players info and games info to stats logic
25. Passed: Stats data logic takes info and returns statistics between players associated with accounts retrieved in user_data.csv
26. Passed: Data is stored in pair_records.csv and player_records.csv
27. Passed: Data is retrieved from pair_records.csv and player_records.csv
28. Passed: Data is displayed as a table of statistics and a pair score in GUI panel

Map Selection integration test 2	individual	Nov 20 2013

This integration phase uses the bottom-up strategy, because most crucial classes are the low-level ones that read from loaded custom map files. Integration involves back-end TXT file reader, back-end map (grid) creation logic, and front-end map selector GUI.

Tested flow of data:

1. Passed: Updated GUI with 3 radio buttons for default map selection and 1 button for file browser are loaded
2. Passed: Visual previews of 3 default maps are pre-loaded from files (map1.txt, map2.txt, and map3.txt) in the background
3. Passed: User does not select any map and exits map selection window (by either clicking Close or OK)
4. Passed: No grid is generated and passed on in the back-end by the logic
5. Passed: Repeat step 1 and 2
6. Passed: User clicks on radio button to select a map
7. Passed: GUI displays preview grid for selected map

8. Passed: User clicks OK to confirm selection
9. Passed: Grid creation logic returns mapped grid corresponding to selected file
10. Passed: Steps 5-9 are repeated for each of the 3 default maps
11. Passed: Repeat steps 1 and 2
12. Passed: User clicks on file browser button and selects a supported TXT file from disk
13. Passed: Visual previews of custom map is loaded from TXT file and preview grid is displayed
14. Passed: Same as step 6

Main Game, Statistics, Login, and Map Selection int. test	collaborative	Nov 23 2013

This integration phase links the Login, Map Selection, Main Game, and Statistics segments. They are linked through the back-end, whereas their front-end GUIs are independent.

1. Passed: GUI panel for user management is loaded with 2 sets of fields and buttons
2. Passed: Non-existent user info (username) is used to attempt login and error message confirms user is not found in user_data.csv
3. Passed: User info with invalid username is entered to attempt registration and error message confirms username not accepted
4. Passed: User info with valid username and invalid password are entered to attempt registration and error message confirms password does not meet requirements
5. Passed: Valid user info (username and password) is used to register a new user
6. Passed: User info is written to user_data.csv
7. Passed: Same username is used to attempt registration and error message confirms user already exists in user_data.csv
8. Passed: User info is entered and used to log-in as Player1; message confirms successful login
9. Passed: Same user info is entered and used as an attempt to log-in as Player2 and error message confirms that the user is already logged-in
10. Passed: User is logged-out of Player1 account by pressing logout button
11. Passed: Step 7 is repeated, but for Player2
12. Passed: Steps 2-11 are repeated with a different user (username)
13. Passed: After both players are logged-in, a button in the GUI becomes active
14. Passed: By clicking active button, map selection GUI is launched with 3 radio buttons for default map selection and 1 button for file browser are loaded
15. Passed: Visual previews of 3 default maps are pre-loaded from files (map1.txt, map2.txt, and map3.txt) in the background
16. Passed: User does not select any map and exits map selection window (by either clicking Close or OK)
17. Passed: No grid is generated and passed on in the back-end by the logic
18. Passed: Repeat step 14 and 15
19. Passed: User clicks on radio button to select a map
20. Passed: GUI displays preview grid for selected map
21. Passed: User clicks OK to confirm selection
22. Passed: Grid creation logic returns mapped grid corresponding to selected file
23. Passed: Steps 18-22 are repeated for each of the 3 default maps
24. Passed: Repeat steps 14 and 15

25. Passed: User clicks on file browser button and selects a supported TXT file from disk
26. Passed: Visual previews of custom map is loaded from TXT file and preview grid is displayed
27. Passed: Same as step 22
28. Passed: The grid is passed through the back-end, the game is initialized and the received grid is displayed in GUI panel for new round
29. Passed: Either player presses specified key on keyboard and round begins
30. Passed: Both players press specified keys to control movement of racers inside GUI grid
31. Passed: Player actions are mapped onto back-end grid logic
32. Passed: Either player hits a wall in GUI grid
33. Passed: Collision is detected in back-end game logic
34. Passed: Back-end logic determines the winner and loser of the round (or a tie)
35. Passed: Round is terminated and message appears confirming game over and winner
36. Passed: Side info panel updates the number of rounds played and the ratio of wins/losses between players
37. Passed: Steps 29-36 are repeated for a total of 3 rounds (and for each combination of winner/loser/tie)
38. Passed: Game logic passes players info and games info to stats logic
39. Passed: Stats data logic takes info and returns statistics between players associated with accounts retrieved in user_data.csv
40. Passed: Data is stored in pair_records.csv and player_records.csv
41. Passed: Data is retrieved from pair_records.csv and player_records.csv
42. Passed: Data is displayed as a table of statistics and a pair score in GUI panel

Complete integration test (with final updated GUI)	collaborative	Nov 24 2013

This integration phase links the Login, Map Selection, Main Game, and Statistics segments. They are linked through the back-end, whereas their front-end GUIs are independent, but loosely connected with a Main Menu paradigm.

1. Passed: GUI panel for user management is loaded with 2 sets of fields and buttons
2. Passed: Non-existent user info (username) is used to attempt login and error message confirms user is not found in user_data.csv
3. Passed: User info with invalid username is entered to attempt registration and error message confirms username not accepted
4. Passed: User info with valid username and invalid password are entered to attempt registration and error message confirms password does not meet requirements
5. Passed: Valid user info (username and password) is used to register a new user
6. Passed: User info is written to user_data.csv
7. Passed: Same username is used to attempt registration and error message confirms user already exists in user_data.csv
8. Passed: User info is entered and used to log-in as Player1; message confirms successful login
9. Passed: Same user info is entered and used as an attempt to log-in as Player2 and error message confirms that the user is already logged-in
10. Passed: User is logged-out of Player1 account by pressing logout button
11. Passed: Step 7 is repeated, but for Player2

12. Passed: Steps 2-11 are repeated with a different user (username)
13. Passed: After both players are logged-in, a button in the GUI becomes active
14. Passed: By clicking active button, main menu is launched.
15. Passed: The user can launch statistics from the main menu
16. Passed: Data is displayed as a table of statistics and a pair score in GUI panel
17. Passed: The user returns to the main menu
18. Passed: From main menu, map selection is launched
19. Passed: Map selection GUI is launched with 3 radio buttons for default map selection and 1 button for file browser are loaded
20. Passed: Visual previews of 3 default maps are pre-loaded from files (map1.txt, map2.txt, and map3.txt) in the background
21. Passed: User does not select any map and exits map selection window (by either clicking Close or OK)
22. Passed: No grid is generated and passed on in the back-end by the logic
23. Passed: Repeat step 19 and 22
24. Passed: User clicks on radio button to select a map
25. Passed: GUI displays preview grid for selected map
26. Passed: User clicks OK to confirm selection
27. Passed: Grid creation logic returns mapped grid corresponding to selected file
28. Passed: Steps 23-27 are repeated for each of the 3 default maps
29. Passed: Repeat steps 19 and 22
30. Passed: User clicks on file browser button and selects a supported TXT file from disk
31. Passed: Visual previews of custom map is loaded from TXT file and preview grid is displayed
32. Passed: Same as step 27
33. Passed: The grid is passed through the back-end
34. Passed: The user returns to the main menu
35. Passed: The game is initialized and the received grid is displayed in GUI panel for new round
36. Passed: Either player presses specified key on keyboard and round begins
37. Passed: Both players press specified keys to control movement of racers inside GUI grid
38. Passed: Player actions are mapped onto back-end grid logic
39. Passed: Either player hits a wall in GUI grid
40. Passed: Collision is detected in back-end game logic
41. Passed: Back-end logic determines the winner and loser of the round (or a tie)
42. Passed: Round is terminated and message appears confirming game over and winner
43. Passed: Side info panel updates the number of rounds played and the ratio of wins/losses between players
44. Passed: Steps 35-43 are repeated for a total of 3 rounds (and for each combination of winner/loser/tie)
45. Passed: Map selection can be launched in between games
46. Passed: Steps 19-44 are repeated 8 times to ensure robustness
47. Passed: Game logic passes players info and games info to stats logic
48. Passed: Stats data logic takes info and returns statistics between players associated with accounts retrieved in user_data.csv
49. Passed: Data is stored in pair_records.csv and player_records.csv
50. Passed: Data is retrieved from pair_records.csv and player_records.csv
51. Passed: Data is displayed as a table of statistics and a pair score in GUI panel