HW13

I have chosen to refactor home work 7 assignment that is cross country. I have written a method which handles many operations in one method that is added below:

*@*PostMapping("/calculate")

public Results calculateRaceResults(*@*RequestBody List<String> teamNames) {

teamsMap.clear();

for (String teamName : teamNames) {

teamsMap.putIfAbsent(teamName, new Team(teamName));

}

List<Result> resultsList = addingValues(teamNames);

results.setTeams(resultsList);

String tea = sortTeamScores(resultsList);

if (tea != null) {

results.setMessage("team " + tea + " wins");

} else {

tiebreak(resultsList);

}

return results;

}

v I have fixed the above code smell, pasting the fixed code below:

// Endpoint to calculate race results

*@PostMapping*("/calculate")

public Results calculateRaceResults(*@RequestBody* List<String> teamNames) {

initializeTeams(teamNames);

List<Result> resultsList = calculateResults(teamNames);

setResultsAndDetermineWinner(resultsList);

return results;

}

// Initializes teams based on provided team names

private void initializeTeams(List<String> teamNames) {

***teamsMap***.clear();

for (String teamName : teamNames) {

***teamsMap***.putIfAbsent(teamName, new Team(teamName));

}

}

// Calculates results for the teams

private List<Result> calculateResults(List<String> teamNames) {

List<Result> resultsList = addingValues(teamNames);

results.setTeams(resultsList);

return resultsList;

}

// Sets the results and determines the winning team or tiebreak

private void setResultsAndDetermineWinner(List<Result> resultsList) {

String winningTeam = sortTeamScores(resultsList);

if (winningTeam != null) {

results.setMessage("team " + winningTeam + " wins");

} else {

tiebreak(resultsList);

}

}

c By segregating responsibilities into separate methods (initializeTeams, calculateResults, setResultsAndDetermineWinner), the code becomes more modular, readable, and maintainable. Each method now handles a specific task, enhancing code readability and making it easier to understand the purpose of each function.

Moreover, by decoupling functionalities, I have likely improved testability. Unit tests can now be written for each individual method, allowing for more granular and targeted testing. These tests will validate the functionality of each method in isolation, ensuring that after refactoring, the initial tests remain functional and that each piece of the codebase performs as expected.