1. Reasons for Choosing Strategy Design Pattern:
   1. Flexibility: The Strategy pattern allows me to swap algorithms (scoring and tie-breaking) at runtime, making the system flexible and adaptable to different race rule sets without altering the existing code significantly.
   2. Encapsulation: Each strategy encapsulates a specific algorithm, promoting clean and modular code by separating concerns. This helps with easy maintenance and extension of the codebase.

By employing the Strategy pattern, I can better manage the variations in scoring and tie-breaking methods that might occur in different cross-country races while keeping the code organized and maintainable.

1. Implementation: In this scenario, the calculation of a team's score involves different strategies:
   1. Scoring Strategy: Different races might have different scoring methods (e.g., some might consider only the top 5 runners, while others may have different tie-breaking rules).
   2. Tie-Breaker Strategy: The tie-breaking logic might vary based on race rules (e.g., considering the 6th runner's position or other criteria).
2. Implementation Details:
   1. I created an abstract ScoringStrategy interface or base class that defines a method like calculateScore() and TieBreakerStrategy interface with methods like getTiebreakPlace().
   2. Then, implemented concrete classes for different scoring strategies and tie-breaker strategies. For example:
      1. Top5ScoringStrategy: Implemented the scoring logic based on the top 5 runners.
      2. SixthRunnerTieBreakerStrategy: Implemented the tie-breaking logic using the 6th runner's position.
      3. Finally, modified the Team class to accept instances of these strategies and use them in the calculate\_score() and get\_tiebreak\_place() methods accordingly.
3. A, A, B, B, B, A, A, B, B, A

A = 26

B = 29

Team A wins

A screenshot of a computer

Description automatically generated

1. A, A, A, A, A, A, A, B, B, B, B, B

A = 15

B = 50

Team A wins

A screen shot of a computer

Description automatically generated

1. A, B, B, A, B, A, B, A, A, A, B, B, A, B

A = 28

B = 28

Team A wins by tie breaker (their 6th runner finished 10th and Team Bs 6th runner finished 11th)

A screen shot of a computer

Description automatically generated

Unit Test: total 9 test cases

A screenshot of a computer program

Description automatically generated