**1. Introduction**

This report summarizes the testing, analysis, and recommendations for the Bowling Game Assessment. The purpose is to evaluate the functionality, reliability, and maintainability of the BowlingGame class through comprehensive unit testing and code review.

**2. Testing Approach**

The testing methodology followed a systematic and structured approach:

**2.1 Unit Testing**

* Implemented using the **pytest** framework.
* Tested all core features including:
  + All gutter rolls (0 pins)
  + All single-pin rolls (1 pin per roll)
  + Spares and strikes in regular frames
  + 10th-frame spares and strikes with bonus rolls
  + Perfect game scenario (12 strikes)

**2.2 Helper Functions**

* Used roll\_many(), roll\_spare(), and roll\_strike() for reusable and readable tests.

**2.3 Validation Checks**

* Ensured errors are raised for invalid inputs (pins < 0 or > 10).
* Prevented rolls after the game is finished.

**2.4 Code Coverage**

* All methods and scenarios were tested, ensuring complete coverage of the game logic.

**3. Findings**

* **All basic game mechanics function correctly:** The game accurately calculates scores for all standard frames.
* **Edge cases handled:** 10th-frame spares and strikes are properly processed, allowing for bonus rolls.
* **Error handling is robust:** Invalid pin counts and attempts to roll after game completion trigger appropriate exceptions.
* **Code readability improved:** Refactored methods are easier to follow, with clear variable names and documentation.

**Test Results Summary**

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Status** |
| test\_all\_gutters | All rolls are 0 pins. The game score should be 0. | PASSED |
| test\_all\_ones | All rolls knock down 1 pin. Total score should be 20. | PASSED |
| test\_one\_spare | A spare is scored in one frame and followed by a regular roll to check bonus calculation. | PASSED |
| test\_one\_strike | A strike is scored in one frame, followed by two normal rolls to validate strike bonus calculation. | PASSED |
| test\_perfect\_game | All rolls are strikes. The final score should be 300. | PASSED |
| test\_10th\_frame\_spare\_bonus | A spare in the 10th frame grants one bonus roll. Validates correct scoring of the final frame. | PASSED |
| test\_10th\_frame\_strike\_bonus | A strike in the 10th frame grants two bonus rolls. Validates correct scoring for last frame strikes. | PASSED |
| test\_no\_roll\_after\_finished\_game | Ensures the game cannot accept rolls after it has finished. | PASSED |
| test\_invalid\_roll\_negative | Attempts a roll with negative pins. Should raise ValueError. | PASSED |
| test\_invalid\_roll\_too\_many\_pins | Attempts a roll with more than 10 pins. Should raise ValueError. | PASSED |

**4. Risk Assessment**

* **Incorrect scoring logic:** Low risk due to comprehensive tests.
* **Incomplete edge case handling:** Mitigated by testing all possible frame scenarios.
* **Future maintainability issues:** Addressed through refactoring and documentation.

**5. Recommendations**

1. Maintain test suite as code evolves to ensure consistent scoring accuracy.
2. Use helper functions for any new frame-specific test cases to reduce redundancy.
3. Document new features with clear docstrings and update pdoc documentation regularly.
4. Consider performance optimization if the game scales (e.g., multiplayer or extended scoring logic).
5. Conduct regular code reviews to maintain readability and adherence to best practices.
6. **Conclusion**

The BowlingGame class is fully functional, reliable, and maintainable. The comprehensive testing approach confirms correct scoring, robust error handling, and proper management of all edge cases. With the recommended practices in place, the project is well-prepared for future extensions and real-world usage.