```java
import components.map.Map1L;

/**
 * Kernel interface for managing college basketball team
statistics.
 */
public interface CollegeBasketballTeamKernel
        extends Standard<CollegeBasketballTeam> {

    public enum StatCategory {
        FIRST_HALF_POINTS_PER_GAME, SECOND_HALF_POINTS_PER_GAME,
ASSIST_TO_TURNOVER_RATIO, ASSISTS_PER_FIELD_GOAL_MADE,
ASSISTS_PER_GAME, ASSISTS_PER_POSSESSION,
AVERAGE_FIRST_HALF_MARGIN, AVERAGE_SECOND_HALF_MARGIN,
AVERAGE_OVERTIME_MARGIN, AVERAGE_SCORING_MARGIN,
BLOCK_PERCENTAGE, BLOCKS_PER_GAME, DEFENSIVE_EFFICIENCY,
DEFENSIVE_REBOUNDING_PERCENTAGE, DEFENSIVE_REBOUNDS_PER_GAME,
EFFECTIVE_FIELD_GOAL_PERCENTAGE, EFFECTIVE_POSSESSION_RATIO,
EXTRA_CHANCES_PER_GAME, FIELD_GOALS_ATTEMPTED_PER_GAME,
FIELD_GOALS_MADE_PER_GAME, FLOOR_PERCENTAGE,
FREE_THROW_PERCENTAGE, FREE_THROW_RATE,
FREE_THROWS_ATTEMPTED_PER_GAME, FREE_THROWS_MADE_PER_GAME,
FTA_PER_FGA, FTM_PER_100_POSSESSIONS,
NON_BLOCKED_TWO_POINT_PERCENTAGE, OFFENSIVE_EFFICIENCY,
OFFENSIVE_REBOUNDING_PERCENTAGE, OFFENSIVE_REBOUNDS_PER_GAME,
OPPONENT_FIRST_HALF_POINTS_PER_GAME,
OPPONENT_SECOND_HALF_POINTS_PER_GAME,
OPPONENT_ASSIST_TO_TURNOVER_RATIO,
OPPONENT_ASSISTS_PER_FIELD_GOAL_MADE, OPPONENT_ASSISTS_PER_GAME,
OPPONENT_ASSISTS_PER_POSSESSION,
OPPONENT_AVERAGE_SCORING_MARGIN, OPPONENT_BLOCK_PERCENTAGE,
OPPONENT_BLOCKS_PER_GAME,
OPPONENT_DEFENSIVE_REBOUNDING_PERCENTAGE,
OPPONENT_DEFENSIVE_REBOUNDS_PER_GAME,
OPPONENT_EFFECTIVE_FIELD_GOAL_PERCENTAGE,
OPPONENT_EFFECTIVE_POSSESSION_RATIO,
OPPONENT_FIELD_GOALS_ATTEMPTED_PER_GAME,
OPPONENT_FIELD_GOALS_MADE_PER_GAME, OPPONENT_FLOOR_PERCENTAGE,
OPPONENT_FREE_THROW_PERCENTAGE, OPPONENT_FREE_THROW_RATE,
OPPONENT_FREE_THROWS_ATTEMPTED_PER_GAME,
OPPONENT_FREE_THROWS_MADE_PER_GAME, OPPONENT_FTA_PER_FGA,
OPPONENT_FTM_PER_100_POSSESSIONS,
OPPONENT_NON_BLOCKED_TWO_POINT_PERCENTAGE,
OPPONENT_OFFENSIVE_REBOUNDING_PERCENTAGE,
OPPONENT_OFFENSIVE_REBOUNDS_PER_GAME,
OPPONENT_OVERTIME_POINTS_PER_GAME,
OPPONENT_PERCENT_OF_POINTS_FROM_TWO_POINTERS,
OPPONENT_PERCENT_OF_POINTS_FROM_THREE_POINTERS,
OPPONENT_PERCENT_OF_POINTS_FROM_FREE_THROWS,
OPPONENT_PERSONAL_FOUL_PERCENTAGE,
OPPONENT_PERSONAL_FOULS_PER_GAME,
OPPONENT_PERSONAL_FOULS_PER_POSSESSION,
OPPONENT_POINTS_FROM_TWO_POINTERS,
OPPONENT_POINTS_FROM_THREE_POINTERS, OPPONENT_POINTS_PER_GAME,
OPPONENT_SHOOTING_PERCENTAGE, OPPONENT_STEAL_PERCENTAGE,
OPPONENT_STEALS_PER_GAME, OPPONENT_STEALS_PER_POSSESSION,
OPPONENT_TEAM_REBOUNDS_PER_GAME,
OPPONENT_THREE_POINTERS_ATTEMPTED_PER_GAME,
OPPONENT_THREE_POINTERS_MADE_PER_GAME,
OPPONENT_THREE_POINT_PERCENTAGE, OPPONENT_THREE_POINT_RATE,
OPPONENT_TOTAL_REBOUNDS_PER_GAME,
OPPONENT_TRUE_SHOOTING_PERCENTAGE, OPPONENT_TURNOVER_PERCENTAGE,
OPPONENT_TURNOVERS_PER_GAME, OPPONENT_TURNOVERS_PER_POSSESSION,
OPPONENT_TWO_POINT_PERCENTAGE, OPPONENT_TWO_POINT_RATE,
OVERTIME_POINTS_PER_GAME, PERCENT_OF_POINTS_FROM_TWO_POINTERS,
PERCENT_OF_POINTS_FROM_THREE_POINTERS,
PERCENT_OF_POINTS_FROM_FREE_THROWS, PERSONAL_FOUL_PERCENTAGE,
PERSONAL_FOULS_PER_GAME, PERSONAL_FOULS_PER_POSSESSION,
POINTS_FROM_TWO_POINTERS, POINTS_FROM_THREE_POINTERS,
POINTS_PER_GAME, POSSESSIONS_PER_GAME, SHOOTING_PERCENTAGE,
STEAL_PERCENTAGE, STEALS_PER_GAME, STEALS_PER_POSSESSION,
TEAM_REBOUNDS_PER_GAME, THREE_POINTERS_ATTEMPTED_PER_GAME,
THREE_POINTERS_MADE_PER_GAME, THREE_POINT_PERCENTAGE,
THREE_POINT_RATE, TOTAL_REBOUNDING_PERCENTAGE,
TOTAL_REBOUNDS_PER_GAME, TRUE_SHOOTING_PERCENTAGE,
TURNOVER_PERCENTAGE, TURNOVERS_PER_GAME,
TURNOVERS_PER_POSSESSION, TWO_POINT_PERCENTAGE, TWO_POINT_RATE,
AWAY_BY_OTHER, CONSISTENCY_BY_OTHER, HOME_ADV_BY_OTHER,
HOME_BY_OTHER, LUCK_BY_OTHER, NEUTRAL_BY_OTHER,
PREDICTIVE_BY_OTHER, FIRST_HALF_BY_OTHER, FUTURE_SOS_BY_OTHER,
IN_CONFERENCE_BY_OTHER, IN_CONFERENCE_SOS_BY_OTHER,
LAST_10_GAMES_BY_OTHER, LAST_5_GAMES_BY_OTHER,
NON_CONFERENCE_BY_OTHER, NON_CONFERENCE_SOS_BY_OTHER,
SCHEDULE_STRENGTH_BY_OTHER, SEASON_SOS_BY_OTHER,
SECOND_HALF_BY_OTHER, SOS_BASIC_BY_OTHER, VS_101_200_BY_OTHER,
VS_1_25_BY_OTHER, VS_201_AND_UP_BY_OTHER, VS_26_50_BY_OTHER,
VS_51_100_BY_OTHER
    }
```

```java

    /**
     * Adds a custom statistic to the team's data.
     *
     * @param category
     *            the category of the statistic
     * @param rank
     *            the rank of the statistic
     * @param value
     *            the value of the statistic
     * @requires category != null && value != null
     * @ensures the custom statistic is added to the team's data
     */
    void addCustomStatistic(StatCategory category, int rank,
double value);

    /**
     * Removes a statistic category from the team's data.
     *
     * @param category
     *            the category of the statistic to remove
     * @requires category != null
     * @ensures the specified statistic category is removed from
the team's data
     */
    void removeStatistic(StatCategory category);

    /**
     * Retrieves statistics by category for the team.
     *
     * @param category
     *            the category of statistics to retrieve
     * @return a map containing the statistics for the specified
category
     * @requires category != null
     * @ensures returns a map containing the statistics for the
specified
     *          category
     */
    Map1L<Integer, Double> getStatisticsByCategory(StatCategory
category);

    /**
     * Retrieves all categories of statistics for the team.
     *
     * @return a sequence containing all statistic categories
     * @ensures returns a sequence containing all statistic categories
     */
    Sequence1L<String> getAllCategories();

    /**
     * Identifies the top 5 and bottom 5 statistics for the team.
     *
     * @ensures the top 5 and bottom 5 statistics for the team are identified
     */
    void bestAndWorstStatistics();
}

```