**User interface menu**

*Ask if they would like to automatically load from the default path src/tracker/Roster.txt*

Tracker for the “team name”

1. Load/save
   1. Load match
   2. Load roster
   3. Save roster
2. Manage
   1. Organize roster
   2. Organize the Matches
   3. Reset Statistics
   4. Trade Player
   5. Hire Coach
   6. Fire Coach
3. Print
   1. Particular Player / Coach
   2. The Roster
   3. All Matches

*Ask if they would like to automatically save to the default path src/tracker/Roster.txt*

**File format for input/output files:**

Matches.csv

* Oppoenet team name, your score, opponent score, “regular” or “playoff”
  + jersey number, points, assists, rebounds, field goal %, three print %.
  + jersey number, points, assists, rebounds, field goal %, three print %.
  + …
  + …
  + Do this for the rest of the players on the roster

Roster.csv

* Name, age, matches attended, playoff matches attended, “player” or “coach”
  + player: jersey number, points/g, assists/g, rebounds/g, playoff points/g, playoff assists/g, playoff rebounds/g, field goal %, three point %.
  + coach: win %, playoff win %

**File Structure (src folder):**

* tracker (package)
  + *Roster*
  + Player
  + Coach
  + Match
  + PlayoffMatch
  + Team
  + ExtraPlayers
    - Player1
    - Player2
    - …
  + ExtraCoaches
    - Coach1
    - Coahc2
    - …
  + Matches.txt
    - Match1
    - Match2
    - …
  + Roster.txt
* Main

The Roster class is designed to represent a person on the team's roster, which consists of four fields: the person's name, age, and the number of matches/playoff matches they have played. This abstract class is to be inherited from two other classes, the Player and Coach classes. It performs operations such as incrementing the number of matches/playoff matches played and resetting statistics. It prepares operations such as saving, listing, and displaying statistics.

| ***Roster*** |
| --- |
| #name: String  #age: int  #matchesAttended: int  #playoffMatchesAttended: int |
| +Roster(name: String, age: int, matchesAttended: int, playoffMatchesAttended: int)  +getName(): String  +round(d: double): double  +addMatch: void  +addPlayoffMatch: void  +resetStatistics(): void  +toString():String  *+saveStatistics(stats: String[]): void*  *+listOutStats(): String*  *+display(): void* |

The Player class is inherited from the Roster class and is designed to represent a player on the team's roster, which consists of three unique fields: the player's jersey number, a list of their overall stats, and a list of their playoff stats. This class inherits all methods from the Roster class. It performs operations such as modifying statistics. It also implements the abstract methods in the Roster class.

| **Player** |
| --- |
| -jerseyNumber: int  -statistics: double[]  -playoffStatistics: double[] |
| +Player(name: String, age: int, jerseyNumber: String, matchesAttended: int, playoffMatchesAttended: int)  +modifyStatistic(stats: String[], isPlayoff: boolean): void  +resetStatistics(): void  +toString(): String  +compareTo(other: Player): int  *+saveStatistics(stats: String[]): void*  *+listOutStats(): String*  *+display(): void* |

The Coach class is inherited from the Roster class and is designed to represent a coach on the team's roster, which consists of two unique fields: the coach's overall win percentage and their playoff win percentage. This class inherits all methods from the Roster class. It performs operations such as modifying the win percentage. It also implements the abstract methods in the Roster class.

| **Coach** |
| --- |
| -winPercentage: double  -playoffWinPercentage: double |
| +Coach(name: String, age: int, matchesAttended: int, playoffMatchesAttended: int)  +modifyWinPercentage(matchWon: boolean, isPlayoff: boolean): void  +resetStatistics(): void  +toString(): String  +compareTo(other: Coach): int  *+saveStatistics(stats: String[]): void*  *+listOutStats(): String*  *+display(): void* |

The Match class is designed to represent a match played by the team consisting of three fields: the name of the opponent team, the number of points your team has scored, and the number of points the opponent team has scored. Each match looks at three stats for every player who has played: their points per game, assists per game, and rebounds per game.

| **Match** |
| --- |
| -opponentTeam: String  -yourScore: int  -opponentScore: int |
| +Match(opponentTeam: String, yourScore: int, opponentScore: int)  +toString(): String  +compareTo(other: Match): int |

The PlayoffMatch class extends the Match class is designed to represent a match - specifically a playoff match - played by the team consisting of four fields: three from the Match class and the bracket stage. Each playoff match looks at five stats for every player who has played: the three from the match class as well as field goal percentage and three point percentage.

| **PlayoffMatch** |
| --- |
| bracketStage: String |
| +PlayoffMatch(opponentTeam: String, yourScore: int, opponentScore: int, bracketStage: String)  +toString(): String |

The Team class is designed to represent a basketball team consisting of one singular field: the name of the team. This class allows you to execute a variety of operations such as loading all information from a match and the current roster, saving your current roster, sorting your roster and matches, resetting your team's statistics, a variety of displaying, trading players, and firing/hiring coaches.

| **Team** |
| --- |
| -teamName: String  -roster: ArrayList<Roster>  -matches: ArrayList<Match>  -playoffMatches: ArrayList<PlayoffMatch> |
| +Team(teamName: String)  +getTeamName(): String  +loadMatchStatistics(matchFilePath: String, matchFileName: String): boolean  +loadRosterStatistics(rosterFilePath: String): boolean  +saveRosterStatistics(rosterFilePath: String): boolean  +organizeRoster(): void  +organizeMatches(): void  +resetStatistics(): void  +findRoster(name: String): int  +displayRoster(name: String): void  +displayWholeRoster(): void  +displayAllMatches(): void  +tradePlayer(obtainPlayerPath: String, obtainPlayerName: String, losePlayerName: String): void  +hireCoach(coachFilePath: String, coachName: String): void  +fireCoach(coachName: String): void  +moveToTrash(filePath:String, fileName: String): void |