MINI PROJECT

NAME : S.V. WEDISINGHE

REGISTRATION NO: 102779

INDEX NO : 21/ENG/122

SUBMISSION DATE: 25/09/2023

UML Diagram of the Project

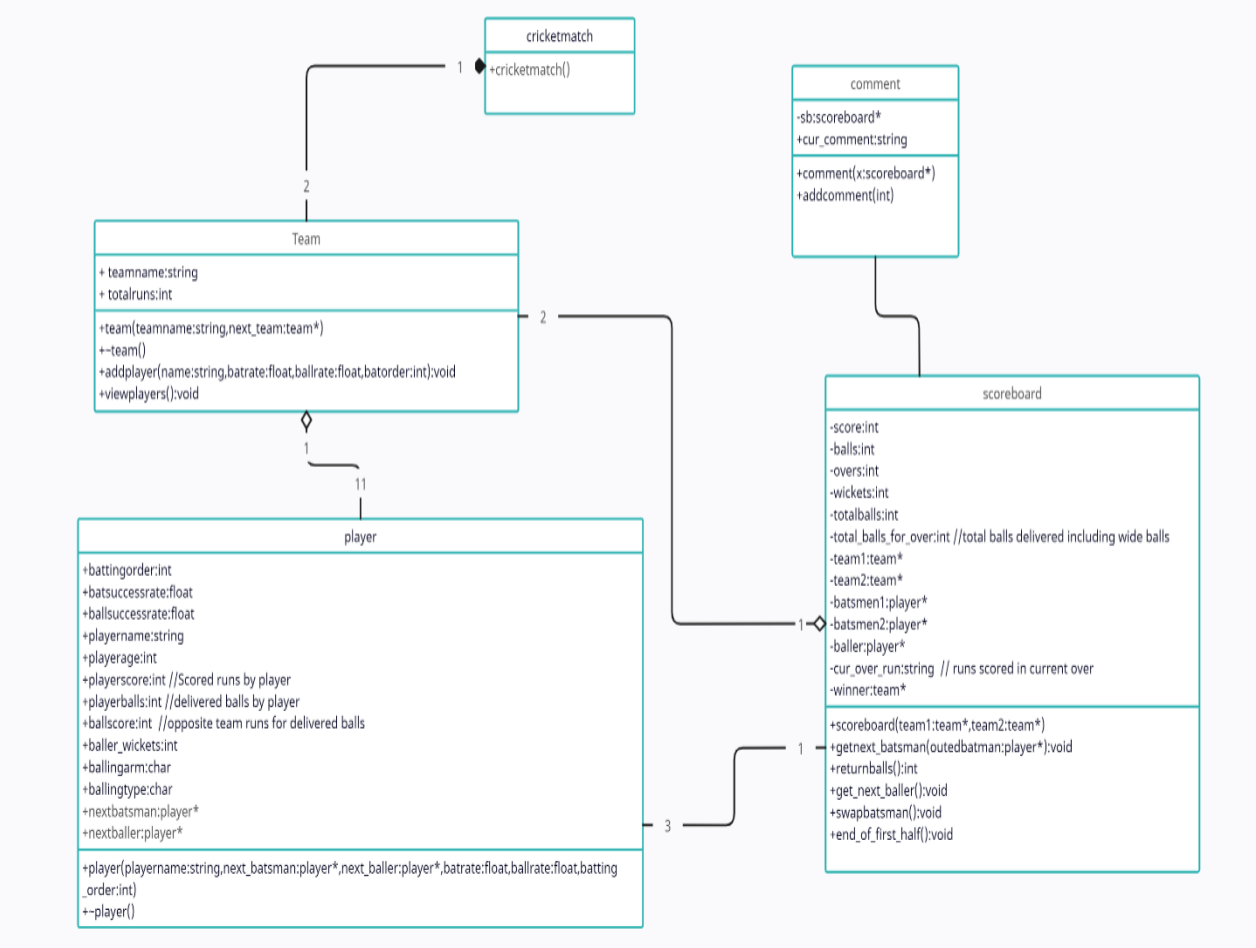

FIGURE 01: UML DIAGRAM OF THE PROJECT

Design of the Project

As shown in the UML diagram this project mainly consists five(5) classes.

I. Cricketmatch class
II. Team class
III. Player class
IV. Scoreboard class
V. Comment

Cricketmatch class

```
class cricketmatch {
public:
    team* team2;
    team* team1;

    cricketmatch() {
        team2 = new team("0", nullptr);
        team1 = new team("0", team2);
    }
};
```

Team class

```
class team {
public:
    string teamname;
    int totalruns;
    team* next;
    player* header;

    team(string teamname, team* next) {
        this->teamname = teamname;
        this->next = next;
        this->header = new player("sds", 0, nullptr, nullptr, 0, 0, 0);
    }
    ~team() {}

    void addplayer(string name, float batrate, float ballrate, int battingorder) {
        if (header->getplayername() == "sds") {
            player* newplayer = new player(name, 12, nullptr, nullptr, batrate, ballrate, battingorder);
            header = newplayer;
        }
        else if (header->nextbatsman == nullptr) {
            player* newplayer = new player(name, 12, nullptr, header, batrate, ballrate, battingorder);
            header->nextbatsman = newplayer;
        }
        else {
            player* temphead = header;
            while (temphead->nextbatsman != nullptr) {
                temphead = temphead->nextbatsman;
            }
            player* newplayer = new player(name, 12, nullptr, temphead, batrate, ballrate, battingorder);
            temphead->nextbatsman = newplayer;
        }
    }

    void viewplayers() {
        player* headplayer = header;
        while (headplayer != nullptr) {
            std::cout << headplayer->playername << endl;
            headplayer = headplayer->nextbatsman;
        }
    }
};
```

“int totalruns” in this class stores total score team achived during a half.

“team* next ” stores the pointer to other team.

“player* header” store the pointer to header player of the player linked list. Team class stores 11 players as a linked list.

player class

```cpp
class player {
public:
    int battingorder;
    float battingsuccessrate;
    float ballingsuccessrate;
    string playername;
    int playerage;
    int playerscore = 0;//player individual score
    int playerballs = 0;//Balls delivered as a bawler.DOn't include wide balls
    int facedballs = 0;//player faced balls as an batsman
    int ballscore = 0;// Score the bawler gave to oppsite team by balling and wide balls
    int ballerwickets = 0;//total wickets got by baller

    char ballingarm;// L for left arm R for right arm
    char ballingtype;// F for fast S for slow

    player* nextbatsman;
    player* nextballer;


    string getplayername() {
        return playername;
    }

    player(string Playername, int Playerage, player* next_batsman, player* next_baller, float battingsuccessrate, float ballingsuccessrate, int battingorder) {
        this->playername = Playername;
        this->playerage = Playerage;
        this->nextbatsman = next_batsman;
        this->nextballer = next_baller;
        this->ballingsuccessrate = ballingsuccessrate;
        this->battingsuccessrate = battingsuccessrate;
        this->battingorder = battingorder;
    }
    ~player() {

    }

};
```

Player class stores data about player(his bat rate, ball rate, name)

scoreboard class

```
class scoreboard {
private:
    int score = 0;
    int balls = 0;
    int overs = 0;
    int wickets = 0;
    int total_balls = 0;
    int total_balls_for_over = 0;//total balls sent including wide balls
    team* team1 = new team("", nullptr);
    team* team2;
    player* batsman1;
    player* batsman2;
    player* baller;
    string cur_over_runs = "                                                    ";
    team* winner;
public:
    scoreboard(team* team1, team* team2) {
        this->team1 = team1;
        this->team2 = team2;
        batsman1 = team1->header;
        batsman2 = team1->header->nextbatsman;
        player* temphead = team2->header;
        while (temphead->nextbatsman != nullptr) {
            temphead = temphead->nextbatsman;
        }
        baller = temphead;
    }
    ~scoreboard() {}

    int returnballs() {
        return balls;
    }

    void getnextbatsman(player* outedbatman) {

        if (outedbatman->battingorder > batsman2->battingorder) {
            batsman1 = outedbatman->nextbatsman;
        }
        else {
            batsman1 = batsman2->nextbatsman;
        }
```

scoreboard class contains all the attributes required to show in scoreboard.

It includes important metods like:

- "`void getnextbatsman(player* outedbatman)`" : use for get next batsman after a wicket
- "`void getnextballer()`" : used when a over is over and next bowler is balling
- " `void swapbatsmen()`" : Used when batsmen scored a run or over finish then batsmen are swapping
- "`void endoffirsthalf()`" : Used when end of a half reached. Batting team and balling team of the Scoreboard are changing. All scores & overs are resetted.

scoreboard class contains few friend functions :

```
friend void showscreen(scoreboard* sb, comment* commentar,int);
friend int OSsystem(scoreboard* score_board, comment* commentar);
friend class comment;
```

These friend functions allow outside functions like "showscreen()" , "OSsystem()" & classes like "comment" access to private attributes.

comment class

```
class comment {
    scoreboard* sb;
public:
    string cur_comment = " ";

    comment(scoreboard* x) {
        this->sb = x;
        cur_comment = " ";
    }

    void addcomment(int x) {
        switch (x)
        {
        case 1:
            // code for baller running
            if (sb->total_balls_for_over == 0) {

                cur_comment += "First ball of the over. Now balling "+ sb->baller->playername+" .brilliant ";
                if (sb->baller->ballingarm == 'L') {
                    if (sb->baller->ballingtype == 'F') {
                        cur_comment += "left arm fast baller. With a average success rate of " + to_string(sb->baller->ballingsuccessrate) + ".";
                    }
                    else {
                        cur_comment += "left arm spinner with a average success rate of " + to_string(sb->baller->ballingsuccessrate) + ".";
                    }
                }
                else {
                    if (sb->baller->ballingtype == 'F') {
                        cur_comment += "Right arm fast baller. With a average success rate of " + to_string(sb->baller->ballingsuccessrate) + ".";
                    }
                    else {
                        cur_comment += "Right arm spinner with a average success rate of " + to_string(sb->baller->ballingsuccessrate) + ".";
                    }
                }
            }
            else if (sb->returnballs() == 5) {
                cur_comment += "Bowler looking dedicated as he runs towards balling his final delivery!!.....";
            }
            randNum = randomnumber_generator() % 4 + 1;
            switch (randNum)
            {
            case 1:
                cur_comment += "A smooth rundown towards the popping crease by the bowler..";
                break;
            case 2:
```

Comment class stores the commentary lines of the cricket match.
“cur_comment” stores a commentary lines for different situations in the cricket match.

Main functions

This program consists few main functions:

a. “`int randomnumber_generator()`” : generate random number using <random> libraries
b. “`void showscreen(scoreboard* sb,comment* commentar,int turn)`”:showscreen function determines the output layout according to match situation. It provides four(4) differen output layers for **first half**, **first half time out match summary**, `second half`,`second half over match summary` .
c. “`void screenrefresh(scoreboard* SCB,comment*commentar,intx)`”: screenrefresh function refreshes output terminal and pause it for user given time period.
d. “`void textprep(string textline, cricketmatch& match)`”: textprep function preparing input file line to a suitable case and creat team and assign players to suitable team.
e. “`int OSsystem(scoreboard* score_board, comment* commentar)`” : This is the main function of the program. This function controls every ball and batsmen score and outs according to creator defined equations.

INPUT

In the beginning of the program user should provide text file include informations about team and players according to following commands

commands:

Match <number_of_overs_per_half> : "define overs per half"

create <teamname> : " creating a team"

add <teamname> <playername> <playerage> <playerbattingsuccsrate> <playerballsuccessrate> : "adding a player to a team"

<playerates> shoud be send as a percentage// eg: 47.5

- input text file should be named as "inputs.txt"

  example:

```
match 5
create Srilanka
add Srilanka Sahas 23 90 79.34 1
add Srilanka Dimuth 19 89 87.4 2
add Srilanka Lavindu 22 75 87.9 3
add Srilanka Kumara 27 76.7 93.5 4
add Srilanka Pamod 25 74.7 90.5 5
add Srilanka Kiriella 22 63.7 88.5 6
add Srilanka Nuwan 24 70.8 91.3 7
add Srilanka Kamal 28 76.7 93.5 8
add Srilanka Kularathna 19 70.6 92.5 9
add Srilanka Arjuna 20 66.7 95.5 10
add Srilanka Kavindu 20 56.7 93.5 11
create Austrailia
add Austrailia Peter 23 80 90.99 1
add Austrailia Glenn 23 96 78.56 2
add Austrailia Alex 26 74 89.99 3
add Austrailia Jay 19 76 69.4 4
add Austrailia Adam 20 86 80.4 5
add Austrailia Stoinis 25 88 90 6
add Austrailia Rocky 25 76 95 7
add Austrailia Starc 24 76 87.8 8
add Austrailia David 27 76 90.67 9
add Austrailia Kane 26 70 92.99 10
add Austrailia Smith 21 70 96.9 11
```

ASSUMPTIONS

- Assuming first created team using input file wins the toss and elect to bat first
- There is no need of no balls or a umpire or a third_ umpire. No classes have been made to store umpires even though some comments mention his actions.
- Only first eleven(11) players are included in input file.
- All the players in the team can bat and ball under any circumstance.
- Input file have batting success rate & balling success rate of the player. Recommending their balling success rate suppose to be higher than 80%.