CODE

# Cricket-Scorer-java

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

class Player {

String name;

int score;

int ballsPlayed;

int fours;

int sixes;

Player(String name) {

this.name = name;

this.score = 0;

this.ballsPlayed = 0;

this.fours = 0;

this.sixes = 0;

}

void addScore(int runs, int balls, int fours, int sixes) {

this.score += runs;

this.ballsPlayed += balls;

this.fours += fours;

this.sixes += sixes;

}

double getStrikeRate() {

if (ballsPlayed == 0) {

return 0.0;

}

return (score \* 100.0) / ballsPlayed;

}

}

class Bowler {

String name;

int runsGiven;

int wicketsTaken;

int oversBowled;

Bowler(String name, int runsGiven, int wicketsTaken, int oversBowled) {

this.name = name;

this.runsGiven = runsGiven;

this.wicketsTaken = wicketsTaken;

this.oversBowled = oversBowled;

}

double getEconomy() {

if (oversBowled == 0) {

return 0.0;

}

return runsGiven / (oversBowled \* 1.0);

}

}

class TeamTotal {

int totalRuns;

int totalWickets;

int totalOvers;

int highestScore;

int highestWickets;

String highestScorer;

String highestWicketTaker;

TeamTotal() {

this.totalRuns = 0;

this.totalWickets = 0;

this.totalOvers = 0;

this.highestScore = -1;

this.highestWickets = -1;

this.highestScorer = "";

this.highestWicketTaker = "";

}

}

class Summary {

List<TeamTotal> teamTotals;

List<String> teamNames;

Summary(List<TeamTotal> teamTotals, List<String> teamNames) {

this.teamTotals = teamTotals;

this.teamNames = teamNames;

}

JTable generateSummaryTable() {

String[] columnNames = {"Team", "Total Runs", "Total Wickets", "Highest Scorer", "Highest Score", "Highest Wicket-Taker", "Highest Wickets"};

Object[][] data = new Object[teamTotals.size()][columnNames.length];

for (int i = 0; i < teamTotals.size(); i++) {

TeamTotal teamTotal = teamTotals.get(i);

data[i][0] = teamNames.get(i);

data[i][1] = teamTotal.totalRuns;

data[i][2] = teamTotal.totalWickets;

data[i][3] = teamTotal.highestScorer;

data[i][4] = teamTotal.highestScore;

data[i][5] = teamTotal.highestWicketTaker;

data[i][6] = teamTotal.highestWickets;

}

return new JTable(new DefaultTableModel(data, columnNames));

}

String determineWinningTeam() {

int maxRuns = -1;

int winningTeam = -1;

for (int i = 0; i < teamTotals.size(); i++) {

if (teamTotals.get(i).totalRuns > maxRuns) {

maxRuns = teamTotals.get(i).totalRuns;

winningTeam = i;

} else if (teamTotals.get(i).totalRuns == maxRuns) {

// If the runs are the same, check for fewer wickets taken (assuming lower wickets are better)

if (teamTotals.get(i).totalWickets < teamTotals.get(winningTeam).totalWickets) {

winningTeam = i;

}

}

}

return (winningTeam != -1) ? teamNames.get(winningTeam) : "Draw";

}

}

class Team {

String name;

List<Player> players;

List<Bowler> bowlers;

Team(String name, List<Player> players) {

this.name = name;

this.players = players;

this.bowlers = new ArrayList<>();

}

}

class ScoreboardApp {

private JFrame frame;

private JPanel panel;

private List<JTable> battingTables;

private List<JTable> bowlingTables;

private List<DefaultTableModel> battingModels;

private List<DefaultTableModel> bowlingModels;

private List<Team> teams;

private List<TeamTotal> teamTotals;

private List<String> teamNames;

public ScoreboardApp() {

initialize();

}

private void initialize() {

// ... (unchanged)

frame = new JFrame("Scoreboard App");

frame.setBounds(100, 100, 1200, 600);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

panel = new JPanel();

frame.getContentPane().add(panel, BorderLayout.CENTER);

panel.setLayout(new GridLayout(4, 1));

teams = new ArrayList<>();

battingTables = new ArrayList<>();

bowlingTables = new ArrayList<>();

battingModels = new ArrayList<>();

bowlingModels = new ArrayList<>();

teamTotals = new ArrayList<>();

teamNames = new ArrayList<>();

for (int i = 0; i < 2; i++) {

int numberOfPlayers = Integer.parseInt(JOptionPane.showInputDialog("Enter the number of players for Team " + (i + 1) + ":"));

List<Player> players = new ArrayList<>();

for (int j = 0; j < numberOfPlayers; j++) {

players.add(new Player(JOptionPane.showInputDialog("Enter name of Player " + (j + 1) + " for Team " + (i + 1) + ":")));

}

Team team = new Team(JOptionPane.showInputDialog("Enter name of Team " + (i + 1) + ":"), players);

teams.add(team);

teamNames.add(team.name);

// Initialize batting table model with column names

String[] battingColumnNames = {"Player", "Runs", "Balls", "4s", "6s", "Strike Rate"};

DefaultTableModel battingModel = new DefaultTableModel(battingColumnNames, 0);

battingModels.add(battingModel);

// Initialize bowling table model with column names

String[] bowlingColumnNames = {"Bowler", "Overs", "Runs Given", "Wickets", "Economy"};

DefaultTableModel bowlingModel = new DefaultTableModel(bowlingColumnNames, 0);

bowlingModels.add(bowlingModel);

JTable battingTable = new JTable(battingModel);

JTable bowlingTable = new JTable(bowlingModel);

battingTables.add(battingTable);

bowlingTables.add(bowlingTable);

JScrollPane battingScrollPane = new JScrollPane(battingTable);

JScrollPane bowlingScrollPane = new JScrollPane(bowlingTable);

panel.add(createTablePanel(battingScrollPane, team.name + " - Batting"));

panel.add(createTablePanel(bowlingScrollPane, team.name + " - Bowling"));

}

for (int i = 0; i < 2; i++) {

teamTotals.add(new TeamTotal());

}

JButton buttonAddScore = new JButton("Add Score");

panel.add(buttonAddScore);

buttonAddScore.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addScore();

updateScoreboard();

}

});

}

// ... (unchanged)

private JPanel createTablePanel(JScrollPane scrollPane, String title) {

JPanel tablePanel = new JPanel(new BorderLayout());

JLabel titleLabel = new JLabel(title, SwingConstants.CENTER);

tablePanel.add(titleLabel, BorderLayout.NORTH);

tablePanel.add(scrollPane, BorderLayout.CENTER);

return tablePanel;

}

private void addScore() {

for (int i = 0; i < teams.size(); i++) {

Team team = teams.get(i);

// Batting

for (Player player : team.players) {

String runsInput = JOptionPane.showInputDialog("Enter runs scored by " + player.name + ":");

String ballsInput = JOptionPane.showInputDialog("Enter balls played by " + player.name + ":");

String foursInput = JOptionPane.showInputDialog("Enter number of 4s by " + player.name + ":");

String sixesInput = JOptionPane.showInputDialog("Enter number of 6s by " + player.name + ":");

try {

int runs = Integer.parseInt(runsInput);

int balls = Integer.parseInt(ballsInput);

int fours = Integer.parseInt(foursInput);

int sixes = Integer.parseInt(sixesInput);

player.addScore(runs, balls, fours, sixes);

// Update team totals

teamTotals.get(i).totalRuns += runs;

teamTotals.get(i).totalOvers += balls / 6; // Assuming 6 balls per over

// Update highest scorer

if (runs > teamTotals.get(i).highestScore) {

teamTotals.get(i).highestScore = runs;

teamTotals.get(i).highestScorer = player.name;

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Invalid input. Please enter valid integers.");

}

}

// Bowling

for (Player selectedPlayer : team.players) {

String oversBowledInput = JOptionPane.showInputDialog("Enter overs bowled by " + selectedPlayer.name + ":");

String runsGivenInput = JOptionPane.showInputDialog("Enter runs given by " + selectedPlayer.name + ":");

String wicketsTakenInput = JOptionPane.showInputDialog("Enter wickets taken by " + selectedPlayer.name + ":");

try {

int oversBowled = Integer.parseInt(oversBowledInput);

int runsGiven = Integer.parseInt(runsGivenInput);

int wicketsTaken = Integer.parseInt(wicketsTakenInput);

team.bowlers.add(new Bowler(selectedPlayer.name, runsGiven, wicketsTaken, oversBowled));

// Update team totals

teamTotals.get(i).totalWickets += wicketsTaken;

// Update highest wicket-taker

if (wicketsTaken > teamTotals.get(i).highestWickets) {

teamTotals.get(i).highestWickets = wicketsTaken;

teamTotals.get(i).highestWicketTaker = selectedPlayer.name;

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Invalid input. Please enter valid integers.");

}

}

}

}

private void updateScoreboard() {

for (int i = 0; i < teams.size(); i++) {

Team team = teams.get(i);

// Update Batting Table

DefaultTableModel battingModel = battingModels.get(i);

battingModel.setRowCount(0);

for (Player player : team.players) {

Object[] battingRowData = {

player.name,

player.score,

player.ballsPlayed,

player.fours,

player.sixes,

player.getStrikeRate()

};

battingModel.addRow(battingRowData);

}

}

for (int i = 0; i < teams.size(); i++) {

Team team = teams.get((i + 1) % teams.size());

// Update Bowling Table

DefaultTableModel bowlingModel = bowlingModels.get(i);

bowlingModel.setRowCount(0);

for (Bowler bowler : team.bowlers) {

Object[] bowlingRowData = {

bowler.name,

bowler.oversBowled,

bowler.runsGiven,

bowler.wicketsTaken,

bowler.getEconomy()

};

bowlingModel.addRow(bowlingRowData);

}

}

// Display the summary after updating the scoreboard

displaySummary();

}

private void displaySummary() {

Summary summary = new Summary(teamTotals, teamNames);

JTable summaryTable = summary.generateSummaryTable();

JScrollPane summaryScrollPane = new JScrollPane(summaryTable);

String winningTeam = summary.determineWinningTeam();

String summaryText = "Game Summary\n";

summaryText += "Winning Team: " + winningTeam + "\n";

summaryText += "-----------------------------------------\n";

summaryText += "Team Totals:\n";

for (int i = 0; i < teams.size(); i++) {

TeamTotal teamTotal = teamTotals.get(i);

summaryText += "Team " + (i + 1) + ": " + teamTotal.totalRuns + " runs, " + teamTotal.totalWickets + " wickets\n";

}

summaryText += "-----------------------------------------\n";

summaryText += "Highest Scorers:\n";

for (int i = 0; i < teams.size(); i++) {

TeamTotal teamTotal = teamTotals.get(i);

summaryText += "Team " + (i + 1) + ": " + teamTotal.highestScorer + " (" + teamTotal.highestScore + " runs)\n";

}

summaryText += "-----------------------------------------\n";

summaryText += "Highest Wicket-Takers:\n";

for (int i = 0; i < teams.size(); i++) {

TeamTotal teamTotal = teamTotals.get(i);

summaryText += "Team " + (i + 1) + ": " + teamTotal.highestWicketTaker + " (" + teamTotal.highestWickets + " wickets)\n";

}

JOptionPane.showMessageDialog(frame, summaryText, "Game Summary", JOptionPane.INFORMATION\_MESSAGE);

}

public void setVisible(boolean visible) {

frame.setVisible(visible);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

try {

ScoreboardApp window = new ScoreboardApp();

window.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

}