



CRICKET SCORE BOARD

IN J A V A S C R I P T



FATIMA JINNAH WOMEN UNIVERSITY
Department of Software Engineering

Software Construction and Development

Open Ended Lab Report

Submitted To: Engr. Sir Shahzad

Submitted By: Laiba Imran

Registration No.: 2021-BSE-072

Date: 1st January 2024

Cricket Java

1. Overview

The `Cricket` class in Java is a Swing-based application designed to provide live cricket match details through web scraping. It features a graphical user interface (GUI) with a JComboBox for selecting live matches, a "Check Score" button, and a display area for presenting match details.

2. Instance Variables

- `rootWindow (JFrame)`: The main JFrame window for the application.
- `cb (JComboBox<String>)`: A JComboBox for selecting live matches.
- `frame1 (JPanel)`: A JPanel used to display detailed match information.

2.1. `Cricket()``

- Constructor for the Cricket class.
- Initializes the GUI components, sets up the main JFrame window, and adds event listeners.

2.2. `getMatchDetails()``

- Retrieves live cricket match details by scraping the "<https://www.cricbuzz.com/>" website.
- Returns an array of live match descriptions.

2.3. `showMatchDetails()``

- Displays detailed information about the selected match.
- Utilizes a layered pane to show team names, score details, and match summary.

2.4. `getCompleteMatchDetails(String match)``

- Retrieves complete match details for the selected match.
- Uses web scraping to fetch data from "<https://www.cricbuzz.com/>" and extracts information such as match header, score card, and summary.

2.5. `main(String[] args)``

- The main method to launch the application.
- Invokes the Cricket constructor using the SwingUtilities.invokeLater() method.

2.6. `JFrame`

- The main window of the application.

2.7. `JLabels`

- "Live Matches": A label indicating the purpose of the JComboBox.
- "Check Score": A label for the button to check the match score.

- Match details labels: Display information such as team names, score details, and match summary.

2.8. `JComboBox`

- Allows the user to select live cricket matches.

2.9. `JButton`

- "Check Score": Triggers the action to display detailed match information.

2.10. `JLayeredPane`

- Provides a layered architecture for the GUI, positioning components at different levels.

2.11. Web Scraping (Jsoup)

- Utilizes the Jsoup library for web scraping live cricket match details from "https://www.cricbuzz.com/".

3. Usage

- Execute the `main` method to launch the application.
- Select a live match from the dropdown.
- Click the "Check Score" button to display detailed match information.

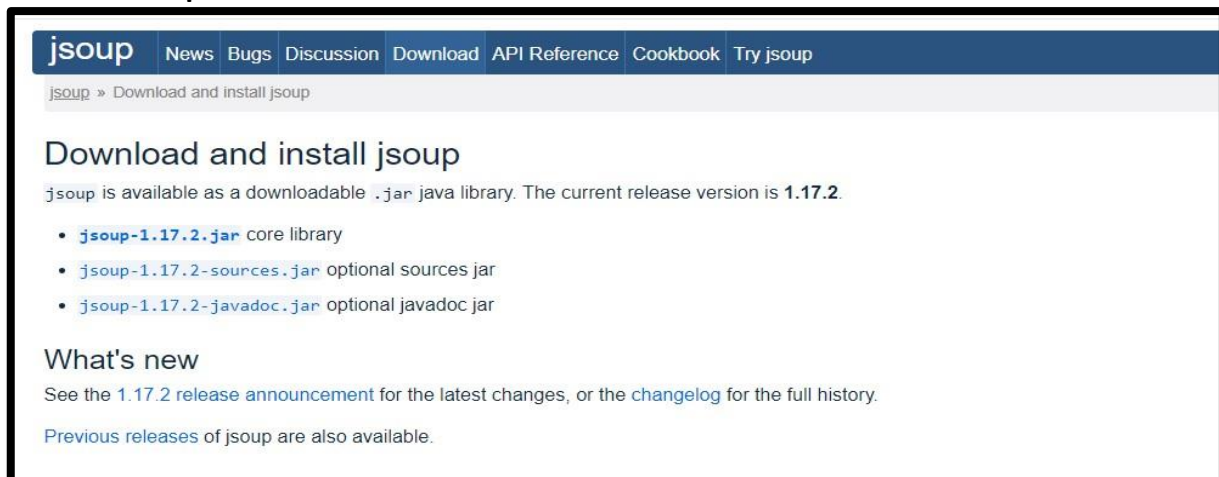
4. Dependencies

- Jsoup: Used for web scraping.

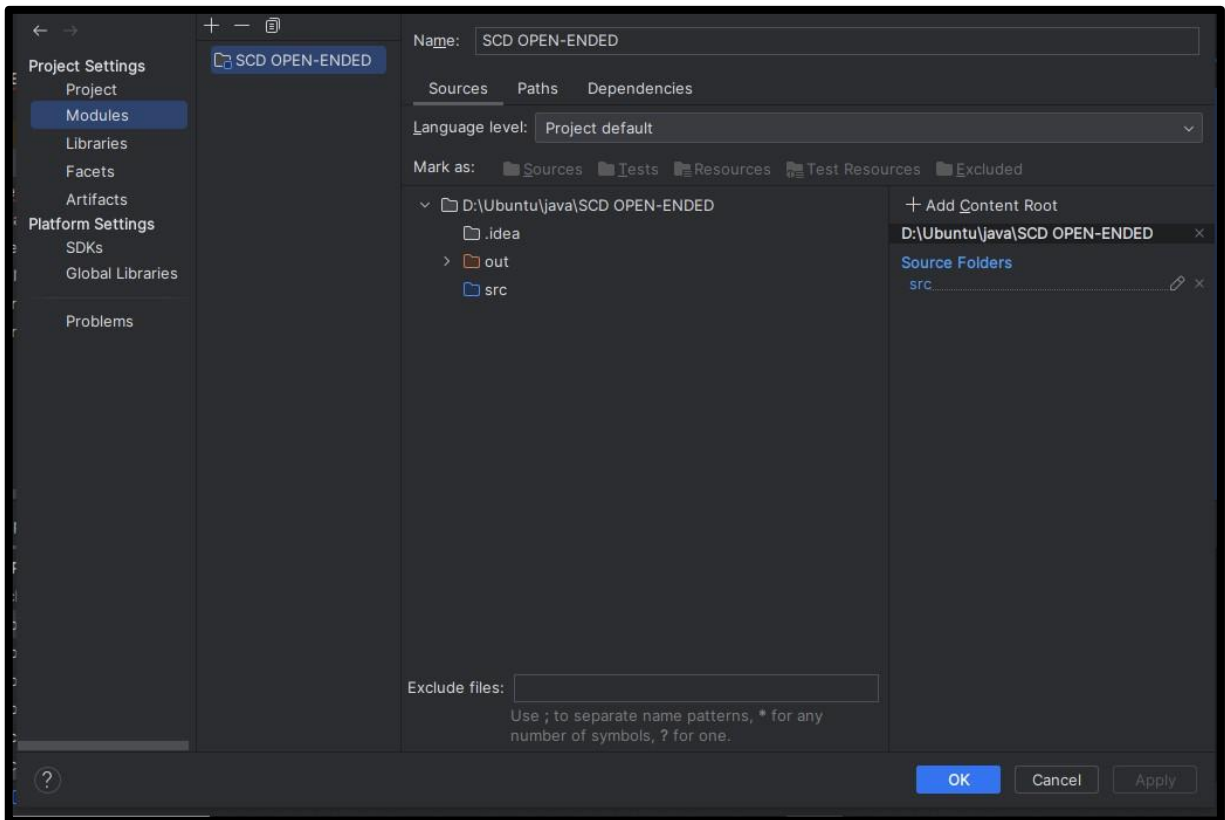
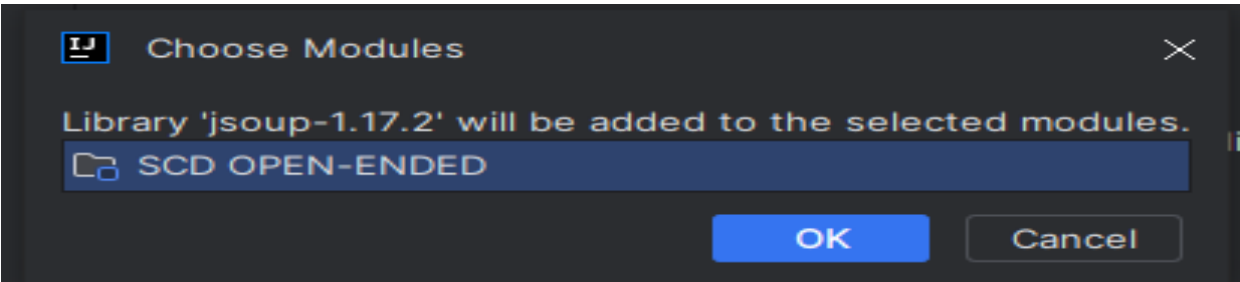
5. Known Issues

- The application relies on the website's structure being scraped, and changes to the website may break functionality.

6. Prerequisites



The screenshot displays the Jsoup project's website. At the top, there is a navigation bar with links for 'jsoup', 'News', 'Bugs', 'Discussion', 'Download', 'API Reference', 'Cookbook', and 'Try jsoup'. Below the navigation bar, the page title is 'Download and install jsoup'. The main content area states that Jsoup is available as a downloadable .jar java library and mentions the current release version is 1.17.2. A bulleted list provides links to download the core library (jsoup-1.17.2.jar), optional sources jar (jsoup-1.17.2-sources.jar), and optional javadoc jar (jsoup-1.17.2-javadoc.jar). Below this, there is a section titled 'What's new' which directs users to the 1.17.2 release announcement for the latest changes or the changelog for the full history. It also notes that previous releases of jsoup are available.



7. Code

```
import org.jsoup.Jsoup;
import
org.jsoup.nodes.Document;
import
org.jsoup.nodes.Element;
import
org.jsoup.select.Elements;
import
javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Cricket {

    private JFrame rootWindow;
    private JComboBox<String>
cb; private JPanel frame1;

    public Cricket() {
        // Creating JFrame window
        rootWindow = new JFrame("LIVE CRICKET SCORE");

        rootWindow.setSize(800, 500);

        rootWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Adding "Live Matches" label
        JLabel label = new JLabel("Live Matches");
        label.setFont(new Font("Times New Roman", Font.PLAIN, 35));

        label.setForeground(Color.white);
```

```

// Adding ComboBox for live
matches String[] matches =
getMatchDetails(); cb = new
JComboBox<>(matches);

// Adding "Check Score" button
JButton checkScoreButton = new JButton("Check Score");

checkScoreButton.setFont(new Font("Times New Roman",
Font.PLAIN, 15)); checkScoreButton.addActionListener(new
ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {

        showMatchDetails();

    }

});

// Creating a layered pane
JLayeredPane layeredPane = new JLayeredPane();

layeredPane.setLayout(null);

// Adding background image to the layered pane
ImageIcon originalBgIcon = new ImageIcon("C:\\Users\\Mansoor Hayat
Khan\\Desktop\\download.jfif");
Image originalImage = originalBgIcon.getImage();
Image scaledImage = originalImage.getScaledInstance(800, 500, Image.SCALE_SMOOTH);

ImageIcon bgIcon = new ImageIcon(scaledImage);

JLabel bgLabel = new
JLabel(bgIcon);

bgLabel.setBounds(0, 0, 800,
500);

layeredPane.add(bgLabel, JLayeredPane.DEFAULT_LAYER);

// Adding components to the layered pane with absolute positioning
label.setBounds(300, 80, 200, 30);

```

```

cb.setBounds(230, 190, 300, 25);
checkScoreButton.setBounds(25, 330, 120, 30);

layeredPane.add(label, JLayeredPane.PALETTE_LAYER);
layeredPane.add(cb, JLayeredPane.PALETTE_LAYER);
layeredPane.add(checkScoreButton,
JLayeredPane.PALETTE_LAYER);

// Setting up the root window
rootWindow.setContentPane(layeredPane);
rootWindow.setVisible(true);
}

private String[] getMatchDetails()
{ try {
    // Web scraping using Jsoup
    String url = "https://www.cricbuzz.com/";

    Document document =
    Jsoup.connect(url).get();

    Elements matchCards = document.select(".cb-match-card");
    java.util.List<String> matchesList = new java.util.ArrayList<>();

    for (int i = 0; i < matchCards.size(); i++) {
        String team1 = matchCards.get(i).select(".cb-hmscg-bat-
txt").text(); String team2 = matchCards.get(i).select(".cb-
hmscg-bwl-txt").text(); String teams = team1 + " vs " + team2;

        // Check if the match has started
        String matchStatus = matchCards.get(i).select(".cb-mtch-crd-
status").text().toLowerCase(); if (!matchStatus.contains("not started")) {
            matchesList.add(teams);
        }
    }
}

```



```

    }

    return matchesList.toArray(new String[0]);
} catch (IOException e) {

    e.printStackTrace();

    return new String[0];
}
}

private void
showMatchDetails() { if
(frame1 != null) {

    rootWindow.getLayeredPane().remove(frame1);
}

// Building match details frame

frame1 = new JPanel();

frame1.setBackground(new Color(173, 216, 230));
frame1.setBounds(180, 280, 600, 200);

frame1.setLayout(null);

// Fetching details of the selected match
String selectedMatch = (String) cb.getSelectedItemAt();
Map<String, String> matchDetails = getCompleteMatchDetails(selectedMatch);

// Displaying team names
JLabel teamLabel = new JLabel(selectedMatch + " - " +
matchDetails.get("match_header")); teamLabel.setFont(new Font("Times New Roman",
Font.BOLD, 15)); teamLabel.setForeground(Color.RED);

teamLabel.setBounds(150, 15, 300, 20);

frame1.add(teamLabel);

```

```
// Displaying score details
JLabel scoreLabel = new JLabel("Score Details: ");

scoreLabel.setFont(new Font("Times New Roman", Font.BOLD,
10)); scoreLabel.setForeground(Color.BLACK);

scoreLabel.setBounds(10, 40, 100, 20);

frame1.add(scoreLabel);


JLabel scoreDetailsLabel = new
JLabel(matchDetails.get("score_card"));

scoreDetailsLabel.setFont(new Font("Times New Roman",
Font.PLAIN, 10)); scoreDetailsLabel.setForeground(Color.BLACK);

scoreDetailsLabel.setBounds(20, 60, 300, 20);

frame1.add(scoreDetailsLabel);


// Displaying match summary
JLabel summaryLabel = new JLabel("Summary: ");

summaryLabel.setFont(new Font("Times New Roman",
Font.BOLD, 10)); summaryLabel.setForeground(Color.BLACK);

summaryLabel.setBounds(10, 100, 100, 20);

frame1.add(summaryLabel);


JLabel summaryDetailsLabel = new JLabel(matchDetails.get("summary"));

summaryDetailsLabel.setFont(new Font("Times New Roman", Font.PLAIN,
10)); summaryDetailsLabel.setForeground(Color.BLACK);

summaryDetailsLabel.setBounds(20, 120, 300, 20);

frame1.add(summaryDetailsLabel);


// Adding the frame to the layered pane
rootWindow.getLayeredPane().add(frame1,
JLayeredPane.PALETTE_LAYER);


// Refreshing the window to reflect changes
rootWindow.revalidate();
```

```
    rootWindow.repaint();  
}
```

```
private Map<String, String> getCompleteMatchDetails(String match) {
```

```
    Map<String, String> matchDetails = new HashMap<>();
```

```
    try {
```

```
        // Web scraping using Jsoup
```

```
        String url = "https://www.cricbuzz.com/";
```

```
        Document document =
```

```
        Jsoup.connect(url).get();
```

```
        Elements matchCards = document.select(".cb-match-  
card"); for (Element matchCard : matchCards) {
```

```
            String team1 = matchCard.select(".cb-hmscg-bat-  
txt").text(); String team2 = matchCard.select(".cb-  
hmscg-bwl-txt").text(); String teams = team1 + " vs " +  
            team2;
```

```
            if (teams.contains(match)) {  
                // Complete details for the selected match  
                String matchHeader = matchCard.select(".cb-mtch-crd-  
hdr").text(); String scoreCard = team1 + " :: " + team2;
```

```
                String summary = matchCard.select(".cb-mtch-crd-state").text();
```

```
                matchDetails.put("match_header",  
                matchHeader);
```

```
                matchDetails.put("score_card", scoreCard);
```

```
                matchDetails.put("summary", summary);
```

```
                break; // Exit loop after finding the match
```

```
            }
```

```
        }
```

```
    } catch (IOException e) {
```

```
        e.printStackTrace(); }
```

```

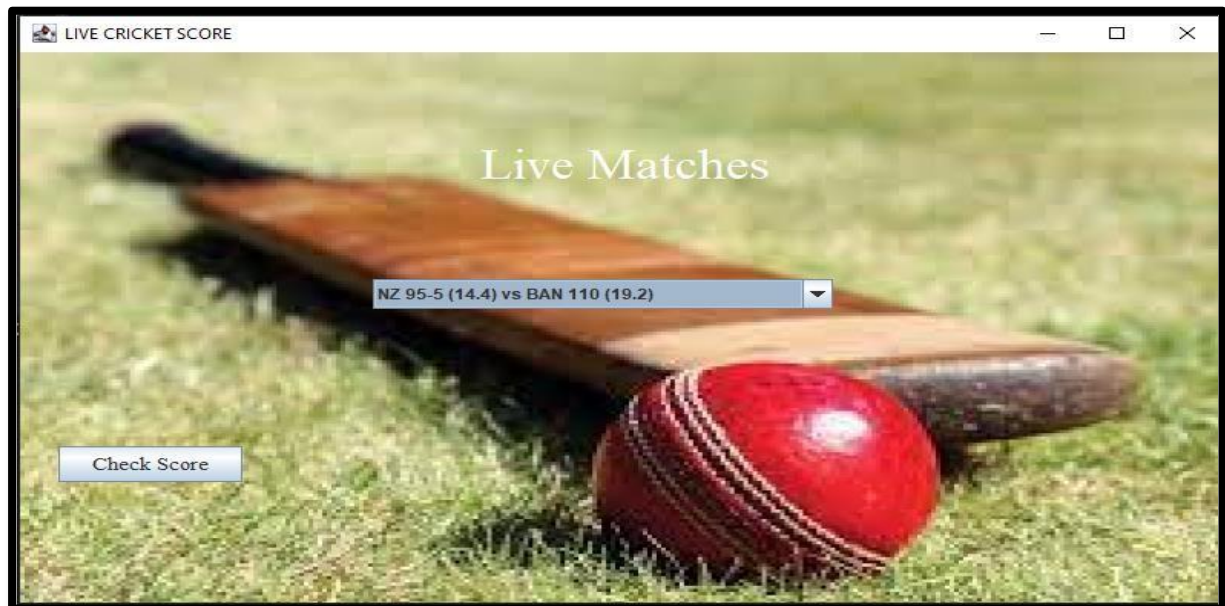
    return matchDetails; }
    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> new Cricket());

    }
}

```

8. Output:





9. Cross Validating

