

1. Initial Scoreboard, Inserting and Removing Entries:

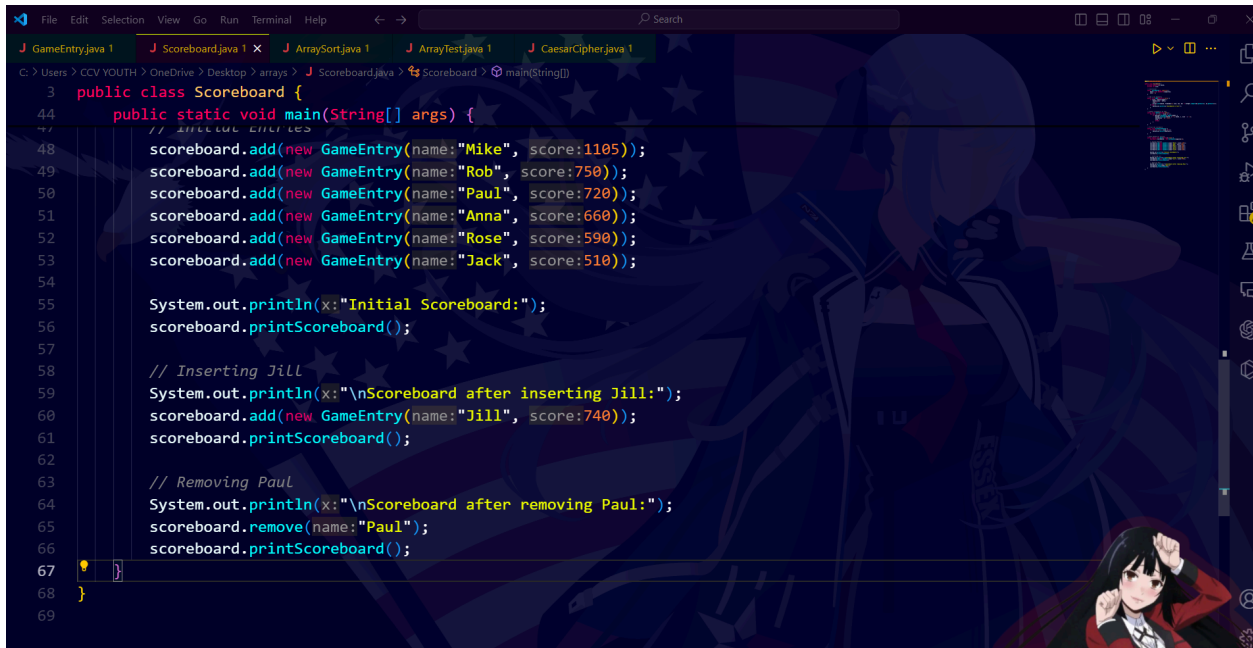
We'll need to maintain a scoreboard and perform the following:

- Print the initial scoreboard.
- Insert a new entry.
- Remove an entry.

Here's the code for **Scoreboard.java**:

```
File Edit Selection View Go Run Terminal Help
J GameEntry.java 1 J Scoreboard.java 1 X J ArraySort.java 1 J ArrayTest.java 1 J CaesarCipher.java 1
C:\Users> CCV YOUTH > OneDrive > Desktop > arrays > J Scoreboard.java > Scoreboard > main(String[])

1 import java.util.Arrays;
2
3 public class Scoreboard {
4     private GameEntry[] board;
5     private int size;
6
7     // Constructor
8     public Scoreboard(int capacity) {
9         board = new GameEntry[capacity];
10        size = 0;
11    }
12
13    // Add a new GameEntry
14    public void add(GameEntry entry) {
15        if (size < board.length) {
16            board[size] = entry;
17            size++;
18            Arrays.sort(board, fromIndex:0, size, (e1, e2) -> Integer.compare(e2.getScore(), e1.getScore()));
19        } else {
20            System.out.println(x:"Scoreboard is full!");
21        }
22    }
23
24    // Remove a GameEntry by name
25    public void remove(String name) {
26
27
28
29
30
31
32
33
34
35
36    // Print the scoreboard
37    public void printScoreboard() {
38        for (int i = 0; i < size; i++) {
39            System.out.println(board[i]);
40        }
41    }
42
43    // Main method for testing
44    public static void main(String[] args) {
45        Scoreboard scoreboard = new Scoreboard(capacity:6);
46    }
```



```
3 public class Scoreboard {
44 public static void main(String[] args) {
47 // Initial Entries
48 scoreboard.add(new GameEntry(name:"Mike", score:1105));
49 scoreboard.add(new GameEntry(name:"Rob", score:750));
50 scoreboard.add(new GameEntry(name:"Paul", score:720));
51 scoreboard.add(new GameEntry(name:"Anna", score:660));
52 scoreboard.add(new GameEntry(name:"Rose", score:590));
53 scoreboard.add(new GameEntry(name:"Jack", score:510));
54
55 System.out.println("Initial Scoreboard:");
56 scoreboard.printScoreboard();
57
58 // Inserting Jill
59 System.out.println(x:"\nScoreboard after inserting Jill:");
60 scoreboard.add(new GameEntry(name:"Jill", score:740));
61 scoreboard.printScoreboard();
62
63 // Removing Paul
64 System.out.println(x:"\nScoreboard after removing Paul:");
65 scoreboard.remove(name:"Paul");
66 scoreboard.printScoreboard();
67 }
68 }
69 }
```

The **Scoreboard** class manages a scoreboard of game entries. It allows the addition and removal of game entries, and sorts the array of entries by score.

- **add(GameEntry entry)**: Adds a new **GameEntry** to the scoreboard and sorts it by score.
- **remove(String name)**: Removes a **GameEntry** by name and shifts remaining entries.
- **sort()**: Sorts the scoreboard in descending order of score.
- **printScoreboard()**: Prints the current scoreboard.

GameEntry.java

The **GameEntry** class represents a single entry in the scoreboard, consisting of a player's name and their score. It includes:

- **Constructor**: Initializes a new game entry with a player's name and score.
- **Getters/Setters**: Provides methods to access and modify the name and score.
- **toString() Method**: Returns a string representation of the game entry, e.g., "Mike: 1105".

```
J GameEntry.java 1 x J Scoreboard.java 1 J ArraySort.java 1 J ArrayTest.java 1 J CaesarCipher.java 1
C:\Users\CCV YOUTH> OneDrive\ Desktop> arrays> J GameEntry.java> ...
1 public class GameEntry {
2     private String name;
3     private int score;
4
5     // Constructor
6     public GameEntry(String name, int score) {
7         this.name = name;
8         this.score = score;
9     }
10
11     // Getter methods
12     public String getName() {
13         return name;
14     }
15
16     public int getScore() {
17         return score;
18     }
19
20     @Override
21     public String toString() {
22         return name + ": " + score;
23     }
24 }
```

ArraySort.java

The `ArraySort` class demonstrates how to sort a character array using insertion sort. During sorting, the array is printed at each step to show the intermediate steps of the sorting process.

- **Main Method:** Initializes an unsorted array, sorts it, and prints the array during each step of sorting.

```
J GameEntry.java 1 x J Scoreboard.java 1 J ArraySort.java 1 x J ArrayTest.java 1 J CaesarCipher.java 1
C:\Users\CCV YOUTH> OneDrive\ Desktop> arrays> J ArraySort.java> ...
1 public class ArraySort {
2     public static void main(String[] args) {
3         char[] array = {'B', 'C', 'D', 'A', 'E', 'H', 'G', 'F'};
4
5         // Print the unsorted array
6         System.out.println("Unsorted array: " + Arrays.toString(array));
7
8         // Sorting the array using a simple sort (insertion sort-like)
9         for (int i = 1; i < array.length; i++) {
10             char key = array[i];
11             int j = i - 1;
12             // Shift larger elements to the right
13             while (j >= 0 && array[j] > key) {
14                 array[j + 1] = array[j];
15                 j = j - 1;
16             }
17             array[j + 1] = key;
18
19             // Print current step
20             System.out.println("cur: " + key + " " + Arrays.toString(array));
21         }
22
23         // Print the sorted array
24         System.out.println("\nSorted array: " + Arrays.toString(array));
25     }
}
```

CaesarCipher.java

The `CaesarCipher` class provides a simple Caesar cipher encryption and decryption method. It shifts each character of a message by a specified number of positions in the alphabet.

- **encrypt(String text, int shift):** Encrypts the given text by shifting each letter by the specified shift value.
- **Main Method:** Demonstrates the encryption of a message and its subsequent decryption.

A screenshot of a code editor window showing the implementation of the CaesarCipher class. The code is written in Java and includes a static method 'encrypt' and a 'main' method for testing. The background of the editor has a dark theme with a faint American flag pattern and a character illustration in the bottom right corner.

```
1 public class CaesarCipher {
2     // Encrypt the text using a Caesar cipher
3     public static String encrypt(String text, int shift) {
4         StringBuilder encrypted = new StringBuilder();
5         for (int i = 0; i < text.length(); i++) {
6             char c = text.charAt(i);
7             if (Character.isLetter(c)) {
8                 char base = (Character.isLowerCase(c)) ? 'a' : 'A';
9                 c = (char) ((c - base + shift) % 26 + base);
10            }
11            encrypted.append(c);
12        }
13        return encrypted.toString();
14    }
15
16    public static void main(String[] args) {
17        String message = "THE EAGLE IS IN PLAY; MEET AT JOE'S.";
18        int shift = 3;
19
20        // Encrypt the message
21        String encryptedMessage = encrypt(message, shift);
22        System.out.println("Message: " + message);
23        System.out.println("Secret: " + encryptedMessage);
24    }
25 }
```

THE OUTPUT ARE

```
PS C:\Users\CCV YOUTH> ^C
PS C:\Users\CCV YOUTH>
PS C:\Users\CCV YOUTH> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.2.13-
scodesws_6a6a4\jdt_ws\jdt.ls-java-project\bin' 'ArrayTest'
arrays equal before sort: true
orig = [55, 21, 71, 43, 93, 93, 3, 13, 93, 30]
data = [55, 21, 71, 43, 93, 93, 3, 13, 93, 30]

arrays equal after sort: false
orig = [55, 21, 71, 43, 93, 93, 3, 13, 93, 30]
data = [3, 13, 21, 30, 43, 55, 71, 93, 93, 93]
PS C:\Users\CCV YOUTH>
```

```
scodesws_6a6a4\jdt_ws\jdt.ls-java-project\bin' Ca
Message: THE EAGLE IS IN PLAY; MEET AT JOE'S.
Secret: WKH HDJOH LV LQ SODB; PHHW DW MRH'V.
PS C:\Users\CCV YOUTH>
```

```
Initial Scoreboard:
Mike: 1105
Rob: 750
Paul: 720
Anna: 660
Rose: 590
Jack: 510

Scoreboard after inserting Jill:
Scoreboard is full!
Mike: 1105
Rob: 750
Paul: 720
Anna: 660
Rose: 590
Jack: 510

Scoreboard after removing Paul:
Mike: 1105
Rob: 750
Anna: 660
Rose: 590
Jack: 510
PS C:\Users\CCV YOUTH>
```