Ex_08_01.java 解説

プログラムの概要

このプログラムは、指定されたテキストファイルからボートの出発（`O`）と到着（`I`）の時刻データを読み込み、各ボートの合計使用時間を計算して出力するものです。プログラムの流れは次の通りです：

1. テキストファイルを読み込む。
2. 読み込んだデータを時系列でソートする。
3. 各ボートの出発と到着をペアリングして使用時間を計算する。
4. 各ボートの使用時間を出力する。

各部分の解説

1. インポートとメインメソッドの設定

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

public class Ex_08_01 {
    public static void main(String[] args) {
```

プログラムは、ファイルを読み込むために`BufferedReader`と`FileReader`を使用し、データを格納・処理するために`Map`や`List`を使用しています。`main`メソッドはプログラムのエントリーポイントです。

2. 初期化

```
        String fileName = args[0];
        Map<Character, Integer> totalMinutesPerBoat = new H
ashMap<>();
        for (char c = 'A'; c <= 'J'; c++) {
            totalMinutesPerBoat.put(c, 0);
        }
```

`args[0]`からファイル名を取得し、`Map`を使用して各ボート（`A`から`J`まで）の合計時間を格納するための初期化を行っています。

3. ファイルからデータを読み込む

```
        List<String[]> records = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new Fil
eReader(fileName))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(" ");
                records.add(parts);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

`BufferedReader`を使用して、ファイルから1行ずつデータを読み込み、空白で分割して`List`に格納します。データが読み込まれた後、リスト`records`に保存されます。

4. 時系列でデータをソート

```
        records.sort((r1, r2) -> {
            int hour1 = Integer.parseInt(r1[2]);
            int minute1 = Integer.parseInt(r1[3]);
            int hour2 = Integer.parseInt(r2[2]);
            int minute2 = Integer.parseInt(r2[3]);
            return (hour1 * 60 + minute1) - (hour2 * 60 + m
inute2);
        });
```

データを時系列でソートしています。具体的には、時刻（時と分）を分単位に変換し、それを基にソートを行っています。

5. 出発と到着のペアリングと時間計算

```
        Map<Character, Integer> startTime = new HashMap<>
();
```

```
        for (String[] parts : records) {
            char boat = parts[0].charAt(0);
            char action = parts[1].charAt(0);
            int hour = Integer.parseInt(parts[2]);
            int minute = Integer.parseInt(parts[3]);

            int timeInMinutes = hour * 60 + minute;

            if (action == 'O') {
                startTime.put(boat, timeInMinutes);
            } else if (action == 'I' && startTime.containsK
ey(boat)) {
                int duration = timeInMinutes - startTime.ge
t(boat);
                totalMinutesPerBoat.put(boat, totalMinutesP
erBoat.get(boat) + duration);
                startTime.remove(boat);
            }
        }
```

各レコードについて、ボートの出発（`O`）と到着（`I`）を処理します。出発時刻が記録されたら`startTime`に保存し、到着時刻が記録された時に使用時間を計算し、それを合計時間に加えます。その後、対応する出発時刻を`startTime`から削除します。

6. ボートの使用時間をソートして出力

```
        List<Map.Entry<Character, Integer>> boatsList = new
ArrayList<>(totalMinutesPerBoat.entrySet());

        // バブルソートを用いて貸出時間の長い順にソート
        for (int i = 0; i < boatsList.size() - 1; i++) {
            for (int j = 0; j < boatsList.size() - 1 - i; j
++) {
                if (boatsList.get(j).getValue() < boatsLis
t.get(j + 1).getValue()) {
                    Collections.swap(boatsList, j, j + 1);
                }
```

```
            }
        }
```

ボートの使用時間を、貸出時間が長い順にソートします。この部分はバブルソートを使用しています。

7. 結果の出力

```
        int totalMinutes = 0;

        for (Map.Entry<Character, Integer> entry : boatsLis
t) {
            int minutes = entry.getValue();
            if (minutes > 0) {
                int hours = minutes / 60;
                int remainingMinutes = minutes % 60;
                System.out.printf("%c %d:%02d\\n", entry.ge
tKey(), hours, remainingMinutes);
            }
            totalMinutes += minutes;
        }

        int totalHours = totalMinutes / 60;
        int remainingMinutes = totalMinutes % 60;
        System.out.printf("%d:%02d\\n", totalHours, remaini
ngMinutes);
    }
}
```

各ボートの使用時間を「時:分」の形式で出力します。最後に、すべてのボートの合計時間も計算して出力しています。