HW01 美國大聯盟季後賽排程-防禦性程式設計說明

Log 設計說明

讀檔偵錯

若讀取 csv 檔時找不到檔案、發生不明錯誤、勝場數或敗場數資料格式不正確，log 會回傳 severe 訊息並終止程式

```
    } catch (FileNotFoundException e) {
        logger.severe( msg: "File not found");
        System.exit( status: 1);
    } catch (IOException e) {
        logger.severe( msg: "Failed to read file");
        System.exit( status: 1);
    } catch (NumberFormatException e) {
        logger.severe( msg: "Invalid number format");
        System.exit( status: 1);
    }
```

資料欄位錯誤警示

若讀取 csv 檔時有資料少於 4 個欄位，log 會回傳 warning 訊息提示哪行資料缺少欄位；若勝場數與敗場數合計不等於 162，log 會回傳 warning 訊息提示哪個隊伍場數不合理

```
if (tokens.length >= 4) {
    String teamName = tokens[0].trim();
    String division = tokens[1].trim();
    int wins = Integer.parseInt(tokens[2].trim());
    int losses = Integer.parseInt(tokens[3].trim());
    if (wins + losses != 162) {
        logger.warning( msg: "Team " + teamName + " total games unequal to 162: " +
    }
    teams.add(new Team(teamName, division, wins, losses));
} else {
    logger.warning( msg: "Missing partial data: " + line);
}
```

讀入資料警示與提示

讀取 csv 檔將隊伍加入 allTeams 列表後，若列表為空，log 會回傳 warning 訊息若列表有資料，log 會回傳 info 訊息提示目前共有多少球隊

```
if (allTeams.isEmpty()) {
    logger.warning( msg: "Team list is empty");
} else {
    logger.info( msg: "Team list includes " + allTeams.size() + " teams");
}
```

球隊所屬分區錯誤警示

球隊歸類到所屬聯盟列表時，若有讀取到球隊分區不含 AL 或 NL 字串，log 會回傳 warning 訊息

```
for (Team t : allTeams) {
    if (t.getDivision().startsWith("AL")) {
        leagueMap.get("AL").add(t);
    } else if (t.getDivision().startsWith("NL")) {
        leagueMap.get("NL").add(t);
    } else {
        logger.warning( msg: "Team " + t.getTeam() + " division unclear: " + t.getDivision());
    }
}
```

球隊數偵錯與提示

所有球隊歸類到所屬聯盟列表後，若有列表少於 6 支球隊，log 會回傳 severe 訊息提示哪個聯盟缺少幾支球隊，並直接終止程式；若列表都有 6 支以上的球隊，log 會回傳 info 訊息提示兩聯盟分別有多少球隊

```
int ALTeams = leagueMap.get("AL").size(), NLTeams = leagueMap.get("NL").size();
if (ALTeams < 6) {
    logger.severe( msg: "AL lack of " + (6 - ALTeams) + " teams");
    System.exit( status: 1);
} else if (NLTeams < 6) {
    logger.severe( msg: "NL lack of " + (6 - NLTeams) + " teams");
    System.exit( status: 1);
} else {
    logger.info( msg: "AL contains " + leagueMap.get("AL").size() + " teams, " + "NL contains " +
}
```

季後賽種子球隊提示

存完所有種子球隊後，log 會回傳 info 訊息提示兩聯盟分別有哪些種子球隊

```
for (String league : leagueMap.keySet()) {
    List<Team> leagueTeams = leagueMap.get(league);
    List<Team> seeds = getSeeds(leagueTeams);
    leagueSeedsMap.put(league, seeds);
    logger.info( msg: league + " selected seeds: " + seeds);
}
```

程式結構說明

Main 類別：

建立 logger 物件以記錄各層級的日誌訊息

```
private static final Logger logger = Logger.getLogger(Main.class.getName());
```

放置輸入的 csv 檔案路徑，並檢查是否有從命令列傳入參數

```
// csv file location
String csvFile = "mlb_standings.csv";
if (args.length > 0) {
    csvFile = args[0];
}
```

呼叫 readCSV 方法將 csv 檔讀取成包含 Team 物件的列表 allTeams

```
// build team list
List<Team> allTeams = readCSV(csvFile);
if (allTeams.isEmpty()) {
    logger.warning( msg: "Team list is empty");
} else {
    logger.info( msg: "Team list includes " + allTeams.size() + " teams");
}
```

分別建立兩聯盟的 HashMap

遍歷讀取到的球隊並以分區字串判斷所屬列表

```
// divide league
Map<String, List<Team>> leagueMap = new HashMap<>();
leagueMap.put("AL", new ArrayList<>());
leagueMap.put("NL", new ArrayList<>());

for (Team t : allTeams) {
    if (t.getDivision().startsWith("AL")) {
        leagueMap.get("AL").add(t);
    } else if (t.getDivision().startsWith("NL")) {
        leagueMap.get("NL").add(t);
    } else {
        logger.warning( msg: "Team " + t.getTeam() + " division unclear: " + t.getDivision());
    }
}
```

建立 HashMap 存放兩聯盟的種子球隊

透過 leagueMap 的 key 取得兩聯盟的球隊列表

呼叫 getSeeds 方法從兩列表各選出 6 支種子球隊存入 HashMap

```
// choose league seeds
Map<String, List<Team>> leagueSeedsMap = new HashMap<>();
for (String league : leagueMap.keySet()) {
    List<Team> leagueTeams = leagueMap.get(league);
    List<Team> seeds = getSeeds(leagueTeams);
    leagueSeedsMap.put(league, seeds);
    logger.info( msg: league + " selected seeds: " + seeds);
}
```

呼叫 printSchedule 方法傳入聯盟名稱字串與對應聯盟的種子球隊列表，印出完整的季後賽程表

```
// print schedule
printSchedule( leagueName: "AMERICAN", leagueSeedsMap.get("AL"),  isAL: true);
printSchedule( leagueName: "NATIONAL", leagueSeedsMap.get("NL"),  isAL: false);
```

readCSV 方法：讀取 csv 檔

使用 BufferedReader 讀取 csv 檔，並利用 br.readLine() 忽略標題行
每讀取一行資料就以逗號將字串分割成各個欄位，依序解析球隊名稱、分區、勝場數和敗場數

```
    List<Team> teams = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {
        String line;
        br.readLine(); //skip title line
        while ((line = br.readLine()) != null) {
            String[] tokens = line.split( regex: ",");
            if (tokens.length >= 4) {
                String teamName = tokens[0].trim();
                String division = tokens[1].trim();
                int wins = Integer.parseInt(tokens[2].trim());
                int losses = Integer.parseInt(tokens[3].trim());
                if (wins + losses != 162) {
                    logger.warning( msg: "Team " + teamName + " total games unequal to 162: " +
                }
                teams.add(new Team(teamName, division, wins, losses));
            } else {
                logger.warning( msg: "Missing partial data: " + line);
            }
        }
```

getSeeds 方法：篩選出種子球隊

遍歷所有球隊，依據分區名稱記錄勝場最多的球隊，成為該分區冠軍
將所有分區冠軍存入 divisionWinners 列表並依勝場數排序

```
// find division champion
Map<String, Team> divisionChamps = new HashMap<>();
for (Team t : leagueTeams) {
    String division = t.getDivision();
    if (!divisionChamps.containsKey(division) || t.getWins() > divisionChamps.get(division).ge
        divisionChamps.put(division, t);
    }
}
List<Team> divisionWinners = new ArrayList<>(divisionChamps.values());
divisionWinners.sort(( Team a,  Team b) -> b.getWins() - a.getWins());
```

創建集合存放分區冠軍後遍歷球隊列表，選出不在集合的球隊作為外卡球隊候補存入 wildcards 列表，並以勝場數排序

```
// build wildcard list
Set<Team> champSet = new HashSet<>(divisionWinners);
List<Team> wildcards = new ArrayList<>();
for (Team t : leagueTeams) {
    if (!champSet.contains(t)) {
        wildcards.add(t);
    }
}
wildcards.sort((Team a, Team b) -> b.getWins() - a.getWins());
```

保留勝場數最高的 3 支球隊作為外卡球隊

```
// find top three of wildcard
if (wildcards.size() > 3) {
    wildcards = wildcards.subList(0, 3);
}
```

創建 allSeeds 列表，先加入分區冠軍再加入外卡球隊

```
// build seed list in order
List<Team> allSeeds = new ArrayList<>();
allSeeds.addAll(divisionWinners); //seed 1~3
allSeeds.addAll(wildcards); //seed 4~6
```

依序設定每支種子球隊的種子編號，返回完整種子球隊列表

```
// set seed number
for (int i = 0; i < allSeeds.size(); i++) {
    allSeeds.get(i).setSeed(i + 1);
}
return allSeeds;
```

printSchedule 方法：印出完整季後賽程表

依照所屬聯盟取得所有種子球隊

```
Team s1 = seeds.get(0);
Team s2 = seeds.get(1);
Team s3 = seeds.get(2);
Team s4 = seeds.get(3);
Team s5 = seeds.get(4);
Team s6 = seeds.get(5);
```

兩聯盟分別格式化輸出季後賽程表

利用 getSeedString 方法填入對應聯盟的種子隊伍名稱與編號

```
if (isAL) {
    System.out.println("(" + leagueName + " LEAGUE)");
    System.out.println("| WILDCARD | ALDS | ALCS | WORLD SERIES |");
    System.out.printf("%6s", getSeedString(s6));
    System.out.println(" ---");
    System.out.printf("%6s", getSeedString(s3));
    System.out.println(" --- ? ----");
    System.out.printf("%12s", getSeedString(s2));
    System.out.println(" ---- ? ----");
    System.out.printf("%6s", getSeedString(s5));
    System.out.println(" ---");
    System.out.printf("%6s", getSeedString(s4));
    System.out.println(" --- ? ----");
    System.out.printf("%12s", getSeedString(s1));
    System.out.println(" ---- ? ---- ? ------------");
    System.out.println("                                        ?");
} else {
    System.out.printf("%6s", getSeedString(s6));
    System.out.println(" --- ? ---- ? ---- ? ------------");
    System.out.printf("%6s", getSeedString(s3));
    System.out.println(" ---");
    System.out.printf("%12s", getSeedString(s2));
    System.out.println(" ----");
    System.out.printf("%6s", getSeedString(s5));
    System.out.println(" --- ? ---- ? ----");
    System.out.printf("%6s", getSeedString(s4));
    System.out.println(" ---");
    System.out.printf("%12s", getSeedString(s1));
    System.out.println(" ----");
    System.out.println("| WILDCARD | NLDS | NLCS | WORLD SERIES |");
    System.out.println("(" + leagueName + " LEAGUE)");
}
```

getSeedString 方法：取得種子隊伍名稱與編號

傳入 Team 物件的資料，返回其球隊名稱與種子編號

```
// return team name and seed number
private static String getSeedString(Team t) {
    return t.getTeam() + " " + t.getSeed();
}
```

Team 類別：

包含球隊名稱、所屬分區、勝場數、敗場數、季後賽球隊的種子編號等屬性

```
class Team {  28 usages  lan
    private String team;  6 usages
    private String division;  2 usages
    private int wins;  2 usages
    private int losses;  2 usages
    private int seed;  3 usages
```

Team 建構子，不包含種子編號屬性

```
public Team(String team, String division, int wins, int losses) {
    this.team = team;
    this.division = division;
    this.wins = wins;
    this.losses = losses;
}
```

存取各屬性的 Getter 跟修改種子編號屬性的 Setter

```
public String getTeam() {  2 usages  lan
    return team;
}
public String getDivision() {  4 usages  lan
    return division;
}
public int getWins() {  6 usages  lan
    return wins;
}
public int getLosses() {  no usages  lan
    return losses;
}
public int getSeed() {  1 usage  lan
    return seed;
}
public void setSeed(int seed) {  1 usage  lan
    this.seed = seed;
}
```

用於 log 訊息提示，回傳種子隊伍的隊伍名稱與種子編號

```
@Override  lan
public String toString() {
    return team + " seed:" + seed;
}
```

以球隊名稱的雜湊值作為唯一標識

```
// avoid team name repeat
@Override  lan
public int hashCode() {
    return team.hashCode();
}
```

根據球隊名稱判斷兩物件是否相同，避免重複加入相同球隊到集合中

```
@Override  lan
public boolean equals(Object obj) {
    if (this == obj) return true;
    if (obj == null || getClass() != obj.getClass()) return false;
    Team other = (Team) obj;
    return team.equals(other.team);
}
```