MLB季後賽

preventive programming

例外處理

```
try {
    // code
} catch (Exception e) {
    e.printStackTrace();
}
```

讀取檔案

- 如果讀出來的檔案不存在會報錯

```
try{
  File jsonFile=new File("src/main/java/org/example/team2023battle.json");
    if(!jsonFile.exists()){
        throw new IOException("File not found: "+jsonFile.getAbsolutePath());
    }//因為new File不會偵測錯誤 所以要在這邊先檢查檔案存不存在 並且列出絕對路徑方便除錯
    Team=objectMapper.readValue(jsonFile,SeasonData.class);//讀取檔案
}catch(Exception e){
    e.printStackTrace();
};
```

資料完整性

- 讀完資料後檢查每支隊伍有沒有打滿比賽，並用log提出警告。
- 接著檢查聯盟是否有輸入錯誤，錯誤的話就丟錯誤出來。

```
for(int i=0;i<Team.getTeamRecord().size();i++){
    TeamRecord obj =Team.getTeamRecord().get(i);
    if((obj.getWins()+obj.getLosses())!=162){
        logger.warning(obj.getTeam()+"沒有打滿或超過162場比賽");
    }//沒打滿代表常規賽可能還沒打完 易出錯
    if(obj.getLeague().equals("America League"))obj.setLeague("AL");
    if(obj.getLeague().equals("Natinonal League"))obj.setLeague("NL");
    //轉換America League 成AL
```

```
    if(!(obj.getLeague().equals("AL") || obj.getLeague().equals("NL"))){
        throw new Exception(obj.getLeague()+" is not AL or NL");
    }
}
```

- 排序後檢查隊伍資料有沒有少給

```
for(int i=0;i<Team.getTeamRecord().size()-3;i++){
    if(!Team.getTeamRecord().get(i).getDivision().equals(divisionCmp)){
        for (int j=0;j<3;j++){
            if(!Team.getTeamRecord().get(j).getDivision().equals(Team.getTeam
Record().get(j+1).getDivision()))
                throw new Exception("資料給不足或是排序錯誤");
            if(!Team.getTeamRecord().get(j).getLeague().equals(Team.getTeamRe
cord().get(j+1).getLeague()))
                throw new Exception("資料給不足或是排序錯誤");
        }//確保給的資料是完整的沒有缺東缺西
    }
    divisionCmp=Team.getTeamRecord().get(i).getDivision();
}
```

- 檢查是否有重複的資料

```
public static int checkReapeat(String[] obj){
    List<String> list = Arrays.asList(obj);
    for(int i=0;i<obj.length;i++){
        if(obj[i]!=null && Collections.frequency(list, obj[i]) > 1){
            return i;
        }
    }
    return -1;
}

int tmpcheck=checkReapeat(Post);
if(tmpcheck!=-1){
    throw new Exception(Post[tmpcheck]+" is repeated team data");
}//檢查如果外卡跟季後賽有任何人重複的話就報錯
```

- 使用 `Optional` 來避免 NullPointerException

```
public static Optional<PlayoffTeam> findTeamByName(List<PlayoffTeam> playoffT
eams, String teamName) {
    return playoffTeams.stream()
            .filter(team -> team.getTeam().equals(teamName))
            .findFirst();
}
```

心得

這段程式碼使用到防禦性programming，讓我深刻體會到一個好的程式，要考慮可能發生的各種意外情況，會變的很複雜。透過檢查檔案存在、資料一致性和完整性和使用 `Optional` 等步驟，才終於寫完，真是相當費力。