# Java Technical Case Study

## Exercise: “Deloitte Digital Away Day"

Deloitte Digital is rewarding their employees in recognition for their hard work for range of successful projects by organising a "Deloitte Digital Away Day". Due to high demand across the firm a number of activities were proposed and a selection of them have been approved. Given there are time constraints, you have been asked to help event organisers by writing a program to accommodate various activities for the day.

Organisers have provided you with the following useful information:

1. The employees will be divided into various teams and each team will be performing various activities
2. Activities start from 9am until lunch is served at 12 noon
3. Activities resume from 1pm and must finish in time for a presentation to be delivered by external speaker on “Staff Motivation"
4. The presentation can start no earlier than 4:00 and no later than 5:00
5. All activity lengths are either in minutes (not hours) or sprint (15 minutes)
6. Due to high spirit of the team there needs to be no gap between each activity

### Provided Input

Here is the provided text file that includes list of activities carefully selected by the organisers.

**Test Data (activities.txt)**

```
Duck Herding 60min
Archery 45min
Learning Magic Tricks 40min
Laser Clay Shooting 60min
Human Table Football 30min
Buggy Driving 30min
Salsa & Pickles sprint
2-wheeled Segways 45min
Viking Axe Throwing 60min
Giant Puzzle Dinosaurs 30min
Giant Digital Graffiti 60min
Cricket 2020 60min
Wine Tasting sprint
Arduino Bonanza 30min
Digital Tresure Hunt 60min
Enigma Challenge 45min
Monti Carlo or Bust 60min
New Zealand Haka 30min
Time Tracker sprint
Indiano Drizzle 45min
```

### Expected Output

The solution you need to produce is expected to create a schedule that includes different teams and their schedule of activities with use of above provided test data.

Please Note:

- Depending on how you choose implement the solution, there might be a different ordering or combination of activities for the teams. This is acceptable; you don’t need to exactly duplicate the sample output given here.
- The expected output can be either in form of console output, or if you wish you can also export the output in the form of a plain text file

**Output Data**

```
Team 1:
09:00 am : Duck Herding 60min
10:00 am : Archery 45min
10:45 am : Learning Magic Tricks 40min
11:25 am : Lunch Break 60min
12:25 pm : Laser Clay Shooting 50min
01:15 pm : Human Table Football 30min
01:45 pm : Buggy Driving 30min
02:15 pm : Salsa & Pickles sprint
02:30 pm : 2-wheeled Segways 45min
03:15 pm : Viking Axe Throwing 60min
04:15 pm : Giant Puzzle Dinosaurs 30min
04:45 pm : Wine Tasting sprint
05:00 pm : Staff Motivation Presentation

Team 2:
09:00 am : Giant Digital Graffiti 45min
09:45 am : Cricket 2020 60min
10:45 am : Arduino Bonanza 30min
11:15 am : Lunch Break 60min
12:15 pm : Digital Tresure Hunt 60min
01:15 pm : Enigma Challenge 45min
02:00 pm : Monti Carlo or Bust 60min
03:00 pm : New Zealand Haka 30min
03:30 pm : Indiano Drizzle 45min
04:15 pm : Time Tracker sprint
05:00 pm : Staff Motivation Presentation
```

## Instructions

1. Implement the solution in Java
2. For the produced solution, there must be a way to supply the application with the input data via a text file (provided as part of this exercise)
3. You can make use of external libraries to solve the problem, for building and testing purposes. As an example, you can use utility, unit testing libraries and build tools for your chosen language (Apache Commons IO, JUnit, Jasmine, Ant, Gradle etc.).
4. You should provide sufficient evidence that your solution is complete by demonstrating that it works correctly against the supplied test data
5. We will assess a number of things including your understanding of the problem, the design aspect of your solution and your object oriented programming skills
6. We expect you to submit what you believe is production-quality code; code that you'd be able to run, maintain, and evolve
7. You must submit a readme file alongside your code. The readme file should include a brief explanation of your design and assumptions, as well as detailed instructions to run your application
8. Please submit the final solution as a archived (.zip) folder or as a link to a Bitbucket or GitHub repository. Note we need to see the source code, not just an executable application