**Department of Computing**

**Bachelor of Information and Communication Technologies Graduate Diploma of Information and Communication Technologies**

**BCSE 101 Software Engineering**

**Assignment Two**

# Programming Assignment 1

Semester Two 2018

Due date: Fri 21 September 2018

Time: 5.00pm

**Section Marks**

A 10

B 20

C 50

D 20

**Total 100**

Student Name/ID ................................................................................................................



Ara and its faculty members reserve the right to use electronic means to detect and help prevent plagiarism. Students agree that when submitting this assignment, it may be subject to submission for textual similarity review to Turnitin.com.

Submissions received late will be subject to a penalty of 10% of the student’s mark per working day.

This assignment is worth 10% of the total marks for BCSE101.

This paper has eight (8) pages including the cover sheet.

### Assignment Description

The purpose of this assignment is to build part of a system that could be used to track results in the 2018 Mitre 10 Cup. The goal of this assessment is to create a basic system that enters data for teams and reports on what matches are scheduled for the first 9 weeks of play.

NOTE: this is a partial system. At the end of this assignment, once you have handed in your work, you will be given a model answer. That will be the basis for the next assignment which will add more details to what you have to do for this first assignment. (Entering of results and points, calculation of semi-finalists, and scheduling of subsequent rounds of play) Think of this assessment as being the beginning ‘iteration 1’ which will eventually lead to the building of a larger system.

The following code should be used to load your test data for JavaScript.

# controller.js

### class Controller {

### static getJSON () {

### // the code for this method WILL Be provided

### }

### static setup () {

### let the2018Competition = new Competition('Mitre 10 Cup', 2018)

### let competitionData = Controller.getJSON()

### 

### let allTeams = competitionData.teams

### for (let aTeam of allTeams) {

### the2018Competition.addTeam(aTeam[0], aTeam[1], aTeam[2])

### }

### 

### let allGames = competitionData.games

### for (let aGame of allGames) {

### the2018Competition.addGame(aGame[0], aGame[1], aGame[2], aGame[3])

### }

### 

### return the2018Competition

### }

### }

**index.html**

Run your JavaScript program using the following html and JavaScript code:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>BCSE101 18S2 Programming Assignment 1</title>

<!-- include source files here... -->

<script src="src/View.js"></script>

<script src="src/Team.js"></script>

<script src="src/Game.js"></script>

<script src="src/Competition.js"></script>

<script src="src/Controller.js"></script>

<div id="divDisplay"></div>

</head>

<body>

<script>

var the2018Competition // GLOBAL FOR DEBUGGING

var go = function () {

the2018Competition = Controller.setup()

let theDisplay = document.getElementById('divDisplay')

document.body.style.fontFamily = 'Courier New'

theDisplay.innerHTML = the2018Competition.getAll()

console.log(the2018Competition)

}

window.onload = go

</script>

</body>

</html>

### controller.py

The following code should be used to load your test data for Python.

The json.txt file will be provided.

**from** competition **import** Competition  
**import** json  
  
  
**def** setup():  
 the\_2018\_competition = Competition(**'Mitre 10 Cup'**, 2018)  
 competition\_data = json.loads(open(**'json.txt'**).read())  
 all\_teams = competition\_data[**'teams'**]  
 **for** team **in** all\_teams:  
 the\_2018\_competition.addTeam(team[**'rank'**], team[**'name'**], team[**'venue'**], team[**'city'**])  
 all\_games = competition\_data[**'games'**]  
 **for** game **in** all\_games:  
 the\_2018\_competition.addGame(game[**'week'**], game[**'homeTeamRank'**], game[**'awayTeamRank'**], game[**'dateTime'**])  
 **return** the\_2018\_competition  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 the\_2018\_competition = setup()  
 print(the\_2018\_competition)

### SECTION A TOTAL: 10 MARKS

**JavaScript: Create the directories, files and constructors.**

1. Create directories and build the necessary classes.

Look at the provided index.html code to see the required directory and file structure.

Plan this by drawing a UML package diagram. <http://www.agilemodeling.com/artifacts/packageDiagram.htm>

(2 marks)

1. Write constructor functions for the classes.

Model your code on the Toybox-Toy example code.

(3 marks)

**Python: Create the .py modules, \_\_init\_\_ functions and import statements.**

1. Create directories and build the necessary classes.

Look at the provided if \_\_name\_\_ == '\_\_main\_\_': code on the previous page to see what is required.

Plan this by drawing a UML package diagram. <http://www.agilemodeling.com/artifacts/packageDiagram.htm>

(2 marks)

1. Write \_\_init\_\_ functions for the classes.

Model your code on the Toybox-Toy example code.

(3 marks)

### SECTION B TOTAL: 20 MARKS

**addTeam**

1. JavaScript: Write an **addTeam** method for the Competition class

Plan this by drawing a class diagram and a collaboration diagram and a sequence diagram. These diagrams will be introduced in lectures and discussed in labs and workshops.

(5 marks)

1. JavaScript: Write a **getTeams** method in the Competition class. Call it from the **getAll** method in the Competition class.

Expected output is:

TEAMS  
Premiership Division  
1 Canterbury   
2 Tasman   
3 Taranaki   
4 North Harbour   
5 Counties Manukau   
6 Auckland   
7 Wellington   
Championship Division  
8 Waikato   
9 Bay of Plenty   
10 Otago   
11 Northland   
12 Manawatu   
13 Hawkes Bay   
14 Southland

(5 marks)

1. Python: Write an **add\_team** method for the Competition class

Plan this by drawing a class diagram and a collaboration diagram and a sequence diagram. These diagrams will be introduced in lectures and discussed in labs and workshops.

(5 marks)

1. Python: Write a **getTeams** method in the Competition class. Call it from the **getAll** method in the Competition class.

(5 marks)

### SECTION C TOTAL: 50 MARKS

**Add Game**

1. JavaScript: Write an **addGame** method for the Competition class

Plan this by drawing a class diagram and a collaboration diagram and a sequence diagram. These diagrams will be introduced in lectures and discussed in labs and workshops.

(10 marks)

1. JavaScript: Write a **getGames** method in the Competition class. Call it from the **getAll** method in the Competition class.

Output should be similar to that displayed at <http://files.allblacks.com/2018_m10_Draw.pdf>

(15 marks)

1. Python: Write an **add\_game** method for the Competition class

Plan this by drawing a class diagram and a collaboration diagram and a sequence diagram. These diagrams will be introduced in lectures and discussed in labs and workshops.

(10 marks)

1. Python: Write a **get\_games** method in the Competition class. Call it from the **get\_all** method in the Competition class.

Output should be similar to that displayed at <http://files.allblacks.com/2018_m10_Draw.pdf>

(15 marks)

### SECTION D TOTAL: 20 MARKS

### Writing get Methods

1. Write a JavaScript **getCanterburyGames** method that displays just the games in which Canterbury is playing.

(5 marks)

1. Write JavaScript a **getCrossOvergames** method that display just the cross over matches.

(5 marks)

1. Write a Python **getCanterburyGames** method that displays just the games in which Canterbury is playing

(5 marks)

1. Write Python **getCrossOvergames** method that display just the cross over matches.

(5 marks)

**How to Submit your Assignment:**

1. Submit all your \src directory files
2. Submit via the PROG1 assignment DROPBOX on BCSE101 Moodle