Institute of Systems Science
National University of Singapore

# NICF – Essential Practices for Agile Teams

# Workshop

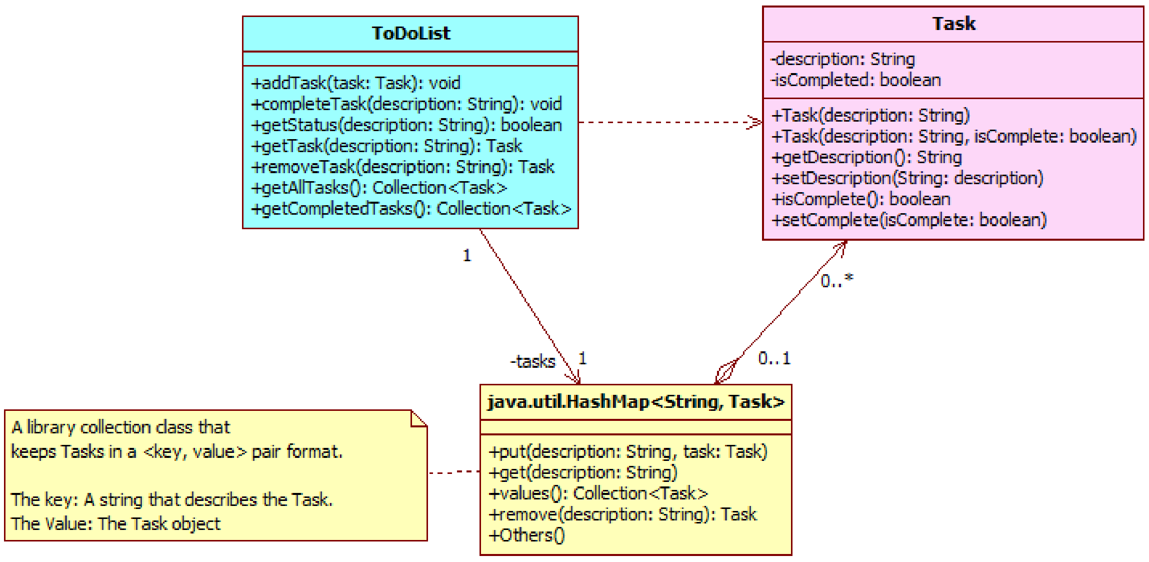
# Test Driven Development

**This page is intentionally left blank**

# Test Driven Development Exercise

## Introduction

1. You are strongly encouraged to try **pair programming** for this exercise. Choose a partner and take turns writing test cases and writing code. If we have odd number of people in the class, you can form a 3-person-group programming. A suggested design is provided below:


ToDoList
+addTask(task: Task): void
+completeTask(description: String): void
+getStatus(description: String): boolean
+getTask(description: String): Task
+removeTask(description: String): Task
+getAllTasks(): Collection<Task>
+getCompletedTasks(): Collection<Task>
Task
-description: String
-isCompleted: boolean
+Task(description: String)
+Task(description: String, isComplete: boolean)
+getDescription(): String
+setDescription(String: description)
+isComplete(): boolean
+setComplete(isComplete: boolean)
1
-tasks
1
0..*
0..1
java.util.HashMap<String, Task>
+put(description: String, task: Task)
+get(description: String)
+values(): Collection<Task>
+remove(description: String): Task
+Others()
A library collection class that keeps Tasks in a <key, value> pair format.
The key: A string that describes the Task.
The Value: The Task object

Task class: This class keeps all the information regarding a task; viz task description and status of the task (whether it is completed).
ToDoList class: This class keeps all the tasks in a HashMap (a Java library class). A HashMap is a Collection class that keeps objects using a key-value pair, where the key uniquely identifies the object (value). The key is used for ease of retrieval of objects. In this case the **key** is the **task description** and the **value** is the associated **task object**.

2. In this workshop, you are required to complete the `ToDoList` class. This class can later be used as part of a web application or backend for mobile app.

## Test Case Design

3. Download the skeletal code '`ToDoListTest.java`', '`ToDoList.java`', '`ToDoList.java`' and '`Task.java`' from **Luminus->Files->Workshop Instructions->TDD Workshop**. Inspect the code and complete the code for some methods using the TDD approach (For a method: define a test and then write the code to pass the test. Repeat.).
4. We have provided the code for the following methods, for you to get started: **getAllTasks()**. This method returns a **Collection<Task>.**

5. Edit the '`ToDoListTest.java`' to defining some test cases for the **addTask (Task task)** method first. Then add code in the **addTask** method to pass the test case.
   E.g.:
   - when a user adds a to-do list item, then an item with the same name is listed when retrieving the list
   - given that there is at least one to-do list item when retrieving the list, when a user deletes an item, then that particular item should not appear when retrieving the to-do list

   You can then do the same for the **removeTask** method.
   You should define at least 2-3 test cases for each method of the class.

## Creating Test Case in Java (Eclipse)

1. Create a new project depending on the type of application you are creating. For this workshop you can just create a Java project. Open Eclipse, select a workspace (default one is fine) and choose File > New > Java Project. You can accept the default project settings or customize it as you wish.

2. Add JUnit to the project:
   a. Go to project properties by right clicking the project in the Package Explorer and select Properties
   b. Select Java Build Path from the left pane
   c. Click "Add Library" button on the right side of the dialog
   d. Choose JUnit and click Next then Finish

3. Now you can create and run JUnit Test Case for your project.

4. To create a JUnit test case, do the following:
   a. From the menu select File > New > JUnit Test Case
   b. Fill up the class detail (package name and the class name) and Eclipse will generate the file according to a predefined template.
   c. A sample test case is provided as part of the template. Customize it as you like.

5. To run a JUnit test case, do the following:
   a. While opening a JUnit test case class, select Run > Run As > JUnit Test
   b. JUnit runner UI will appear (typically on the left side) to run the test cases in the selected file.

## Creating Test Cases in C# (Visual Studio 2017)

1. Create a new solution according to the type of application you are creating. For example, you can choose to create a new Console Application.

2. After you create your solution, right click on the solution on the Solution Explorer and select Add > New Project

3. Select Test on the left pane and choose Unit Test Project. Name the project

4. Make sure that your test project has a reference to the main project that you want to test. To add the reference, right click on References under the unit test project and choose Add References

5. On the left pane, select Projects, Solution and check the project that you want to test.

6. To create a new unit test, right click on the unit test project and choose Add > Unit Test