

INFO213-22SU1 - Object-Oriented Systems Development

[Dashboard](#) / [My courses](#) / [INFO213-22SU1](#) / [Sections](#) / [W4 — JADE Collections and Testing + GUI](#)
/ [INFO213 Tutorial 04 \(Part 1\) - Instructions](#)

INFO213 Tutorial 04 (Part 1) - Instructions

Tutorial Objectives

The first section of Tutorial 4 will cover iterative instructions (together with break and continue) and collection iterators. This will also introduce you to expose the JADE debugger and testing framework. After finishing this section, you will be able to accomplish the following:

- Define collections references in the root class.
- Use iterative instructions or iterators to iterate over collections.
- Use iterative instructions to filter objects in a collection.
- Define new unit tests in the JADE testing framework.

This tutorial will exclusively use screenshots (where necessary) to demonstrate fully new concepts. It relies heavily on previous tutorials, in conjunction with concepts previously discussed in lectures.

We encourage you to try out possible solutions.

Exercise 0: Preparing Development Environment

In this session, we will continue developing the '**SimpleBankModel**' code that you completed in the previous session.

Downloads Needed

- If you have not completed the last tutorial, you can download the complete solution from **INFO 213 Tutorial 3 - Solutions on Week 3** and import it into JADE.
 - Please keep in mind that loading the downloaded SimpleBankModel schema may cause part of your code in the SimpleBankModel schema to be overwritten if it differs from the solution code.
- Download and import a schema called **UnitTestExample** from **Week 4** into JADE.

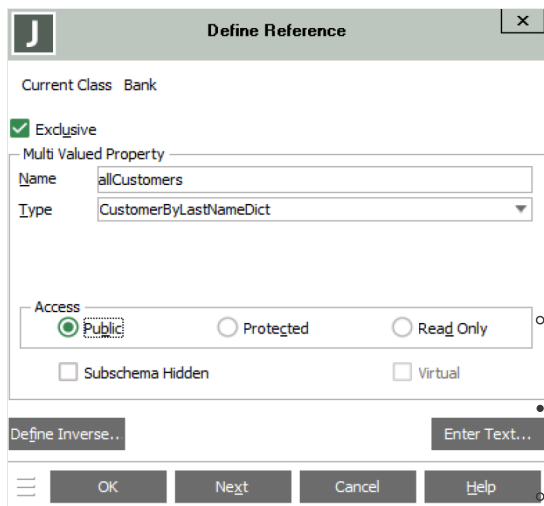
Exercise 1: Adding Collections To A Class

As discussed in previous tutorials, the '**Bank**' class is the root object of the '**SimpleBankModel**' schema.

In this exercise, we will add a **reference** to a **dictionary (collection)** of all customers who are registered with the '**SimpleBank**' bank.

We will also make sure that every '**Customer**' object also 'knows' of its '**Bank**' root object.

- Add a reference to a collection type we created before, '**CustomerByLastNameDict**', in the Bank class as '**allCustomers**'.



Define Reference

Current Class: Bank

☒ Exclusive

Multi Valued Property

Name: allCustomers

Type: CustomerByLastNameDict

Access: ☒ Public ☐ Protected ☐ Read Only

☐ Subschema Hidden ☐ Virtual

Define Inverse... Enter Text...

OK Next Cancel Help

• Please keep in mind that in this case, we define this reference with public visibility only for the code simplicity. This will be corrected shortly.

• Ensure an inverse reference pointing to the single instance of the '**Bank**' class is defined in the '**Customer**' class.

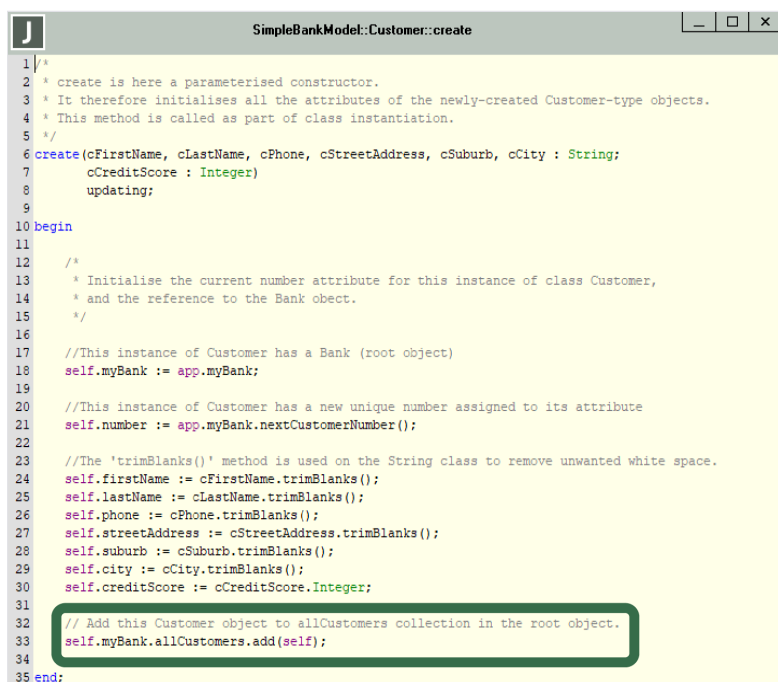
• If you did not do so in the previous tutorial, add this reference, '**myBank**', in the '**Customer**' class.

• This reference is best defined with **protected visibility**.

• Remove the '**customerDict**' variable (of type '**CustomerByLastNameDict**') and the (now redundant) code in the '**createCustomersFromFile**' method that creates and updates this temporary collection.

• All customers are now referenced by the '**allCustomers**' collection in the '**Bank**' class.

• Add the newly created '**Customer**' objects to the '**allCustomers**' collection at the end of the '**Customer**' constructor after the properties for the '**Customer**' object have been set.



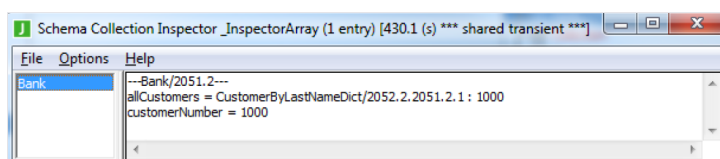
```

1 /*
2  * create is here a parameterised constructor.
3  * It therefore initialises all the attributes of the newly-created Customer-type objects.
4  * This method is called as part of class instantiation.
5  */
6 create(cFirstName, cLastName, cPhone, cStreetAddress, cSuburb, cCity : String;
7       cCreditScore : Integer)
8     updating;
9
10 begin
11     /*
12     * Initialise the current number attribute for this instance of class Customer,
13     * and the reference to the Bank object.
14     */
15
16     //This instance of Customer has a Bank (root object)
17     self.myBank := app.myBank;
18
19     //This instance of Customer has a new unique number assigned to its attribute
20     self.number := app.myBank.nextCustomerNumber();
21
22     //The 'trimBlanks()' method is used on the String class to remove unwanted white space.
23     self.firstName := cFirstName.trimBlanks();
24     self.lastName := cLastName.trimBlanks();
25     self.phone := cPhone.trimBlanks();
26     self.streetAddress := cStreetAddress.trimBlanks();
27     self.suburb := cSuburb.trimBlanks();
28     self.city := cCity.trimBlanks();
29     self.creditScore := cCreditScore.Integer;
30
31     // Add this Customer object to allCustomers collection in the root object.
32     self.myBank.allCustomers.add(self);
33
34 end;

```

• To ensure that your code is working properly, first purge all '**Customer**' instances (**purgeTestObjects**) from the database, and then reload all customers (**createCustomersFromFileDialog**).

• In the singleton '**Bank**' instance, use the **Schema Inspector** to ensure that all 1000 customers are now members of the '**allCustomers**' dictionary:



Schema Collection Inspector _InspectorArray (1 entry) [430.1 (s) *** shared transient ***]

File Options Help

Bank

--Bank/2051.2--

allCustomers = CustomerByLastNameDict/2052.2.2051.2.1 : 1000

customerNumber = 1000

Exercise 2: Iterating Over Collections With Foreach

Information - Foreach

Foreach
<ul style="list-style-type: none"> • <code>foreach [instance] in [instances] do</code> <code>[instructions];</code> <code>endforeach;</code>

How To Work With Foreach

- Given the syntax of the foreach statement above, write a new **JadeScript** class method called '**iterateOverCollections1**' that prints customer number followed by full name for all instances of class '**Customer**'.
 - You can access all '**Customer**' objects either using the virtual instances collection of the '**Customer**' class.
 - Use the '**getPropertyValue(name: String)**' method of the '**Object**' class to assess the value of the customer's number attribute, because it was defined as a **protected** attribute.
 - You will need to run the 'createCustomersFromFileDialog' method to instantiate all customers.**

Exercise 3: Iterating Over Collections Using Iterator

Information - Iterator

While	Iterator
<ul style="list-style-type: none"> • <code>while [iteration] do</code> <code>[instructions];</code> <code>endwhile;</code>	<ul style="list-style-type: none"> • <code>iterator : Iterator;</code> <code>...</code> • <code>iterator :=</code> <code>app.[collection].createIterator();</code> <code>...</code> • <code>iteration.next([instance])</code>

How To Work With Iterator

- Write a new **JadeScript** class method called '**iterateOverCollections2**' to print customer details in a format similar to the previous exercise (based on the collection Iterator object usage).
 - Feel free to refresh your memory with the examples provided in the JADE help system.
 - In this scenario, look for the '**createIterator()**' method among the general collection methods.
- In this method, use the '**app.myBank.allCustomers**' collection reference in the root (singleton '**Bank**') object to access all '**Customer**' objects, instead of the virtual instances collection of the '**Customer**' class.
- Q: In this exercise, we use the last instance of the 'CustomerByLastNameDict'. How would you check current number of instances of that dictionary in your schema?**
- Q: What transient object do you need to remember to delete in the epilog section of the method in this exercise?**

Exercise 4: Filtering (1) With The Where Clause

Information - Where

Where
<ul style="list-style-type: none"> • <code>where [value] = [value] do</code>

How To Work With Where Clause

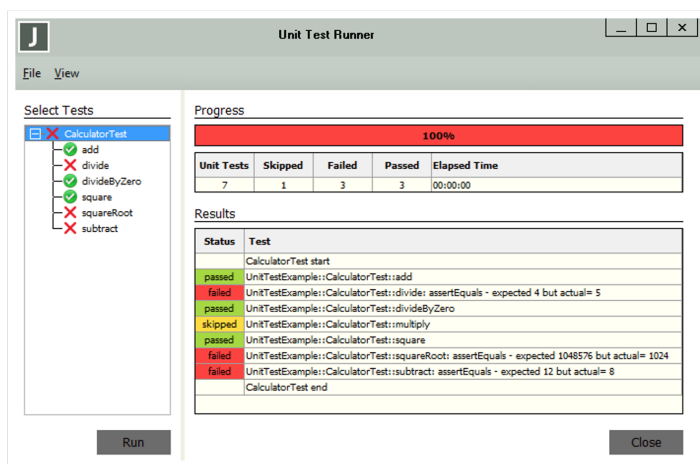
- Write a new **JadeScript** class method (using the foreach loop) called '**customerFilter1**' to find and print all customers who live in the suburb of "**Ilam**".
 - If the suburb is a protected attribute, you might need to use the '**getPropertyValue**' method to obtain the value of the suburb attribute.
- Q: How many customers (in the original list of 1000) from "Ilam" does the database contain?**

Exercise 5: Filtering (2) With The Where Clause

- Write a new **JadeScript** class method (using the foreach loop) called '**customerFilter2**' to find and delete all customers who live in the suburb of "**Ilam**".
 - The **Iterator object** is required in this scenario because the foreach method of looping through a collection creates a lock on the collection, which prevents collection alteration when you want to delete an object.
 - Using the Iterator object, you must need to check the suburb of each '**Customer**' object and remove from '**allCustomers**' (and also delete the object itself) those whose suburb attribute value is "**Ilam**".

Exercise 6: Adding And Running Unit Tests

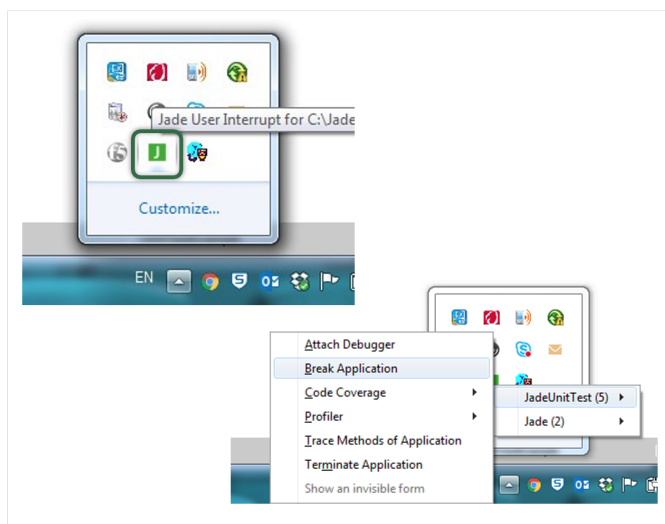
- Download a schema called '**UnitTestExample**' from the **Tutorial Material** on **Week 4** (if you have not already).
 - Run the tests after loading this schema.
 - Simply select the '**CalculatorTest**' class, which is a **subclass** of the '**JadeTestCase**' class, and click **F9** to run the tests.
- After all tests have been executed, you should see a window like this, indicating that three tests failed and one test was not ready to run:



- Examine the test methods in the '**CalculatorTest**' code of the tested methods in the '**Calculator**' class code.
 - Study the code for the tests that passed successfully.
 - Study the code of the tests that failed.
- **Q: Can you see the errors in the code? Can you fix them?**
- First, you will need to complete the implementation of the **multiply** method in the calculator class.
 - Please keep in mind that you will need to change the '**unitTestIgnore**' tag in the test method signature to '**unitTest**'.
- Correct the rest of the tests and the Calculator code methods to make sure all tests pass.
 - The errors in the **divide** unit test method and the **subtract** Calculator method might be less complicated than the problem with the **squareRoot** unit test method.

If you get stuck in an infinite loop

- If you happen to get stuck in an infinite loop, you may use the **JADE User Interrupt** menu available from the system tray at the bottom left corner of your screen.
- Right click and choose the **Break Application**.



[◀ INFO213 Lecture 04 - Slides](#)

Last modified: Tuesday, 25 January 2022, 2:52 PM

Jump to...

[UnitTestExample ▶](#)

For support please contact the [IT Service Desk](#) on 0508 UC IT HELP (0508 824 843) or on 03 369 5000.

Alternatively, visit us at Te Pātaka (Level 2, Central Library).

For support hours, please visit the [IT Services](#) page.

Unless otherwise stated, you may not sell, alter, further reproduce or distribute any part of this material on LEARN to any other person, without permission from the copyright owner. See [UC copyright policy](#).

[English \(en\)](#)

[English \(en\)](#)

[Māori Te Reo \(mi\)](#)

[Get the mobile app](#)