# Week 2 Java Exercises – Using Collections

In Java you can collect any type of class in a collection. We will start with Strings and Numbers

In this program we will put each exercise in a different method, and ask the user which exercise they want to run. The skeleton code supplied by your lecturer will demonstrate how to do this.

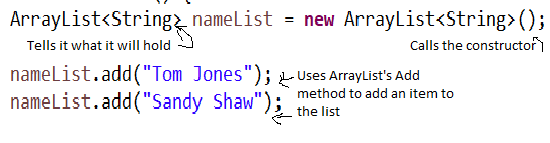
## Task 1

Write a program that stores 10 pupil names in a list. Ask the user to enter a name to search for in the list and output a suitable message if the name is found or not found in the list.

In the Task1 method in WeekTwoExercises

* Create an ArrayList to hold Strings
* Add 10 names to this list using the Add method (they can be anything you like)
* Ask the user to provide a name to search for and display an appropriate message
* Make sure your search is not case sensitive

Your code to create and populate the list should look something like this:



*Questions:*

* What ArrayList method can you use to perform the search?
* How will you make your search case insensitive?

## Task 2:

You have been given an Arraylist called priceList containing a list of double values

Use a for loop to print out the price list, one element per line

Hint: use an ‘advanced for loop’ to easily iterate an ArrayList

Now do the following:

* add the price 8.00 to the end of the list
* add the price 101.00 to position 3 in the list
* remove 1.89 from the list
* display your amended list

Hint: You can do all of these using the standard ArrayList methods

Now create a new ArrayList that also holds Double values. Call it listOfEvens. Iterate through the priceList and copy every element that is an even number into your new list. Check that your new list contains what you expect.

Hint: for a quick check of what is in a collection, just pass the collection to the System.out.println() method. The output won’t be pretty, but will give you a peek at the contents

Hint: if you don’t know how to find out if a number is even, check out the resources in iLearn

Calculate and print to the console:

* How many items are in the listOfEvens ArrayList
* The total of the values in the listOfEvens ArrayList
* The average of the values in the listOfEvens ArrayList

*Question***:**

* How can you format your output so that prices appear in the standard way, i.e. with 2 decimal places?

## Task 3:

A price list is not much use if you don’t know what the prices are for!

Using a HashMap, we can link each price to a specific item by storing them as key/value pairs

In this case, we will create a new menu for a café.

In the Task3 method create an empty instance of a HashMap, telling it the key for each pair will be a string, and the value will be a double. The key will be the dish name, and the value is the cost of that item. This will look something like this:



You may need to import the Java HashMap library – Eclipse will prompt you with an error if so

*Question*:

* Why would we choose to use the dish name as the key rather than the price?

Ask for the name of the café creating the menu, then use a While loop to ask the user to enter dish name and their cost. Each time, add the new dish to the HashMap and loop round to ask them to enter another. This will continue until the user asks to stop.

Once the user is finished entering dishes, display the finished menu as neatly as possible.

Hint: There are a number of ways to format console output to line up your menu nicely. Try researching Java’s printf command, using a C-style format string. Or try another method

## Task 4:

Still in the Task3 method, ask your customer what they want to order, and tell them the total price for their meal.

* Use a Do While loop to let your customer order as many items as they want.
* Keep a running total of the items ordered so you can give them their total bill.