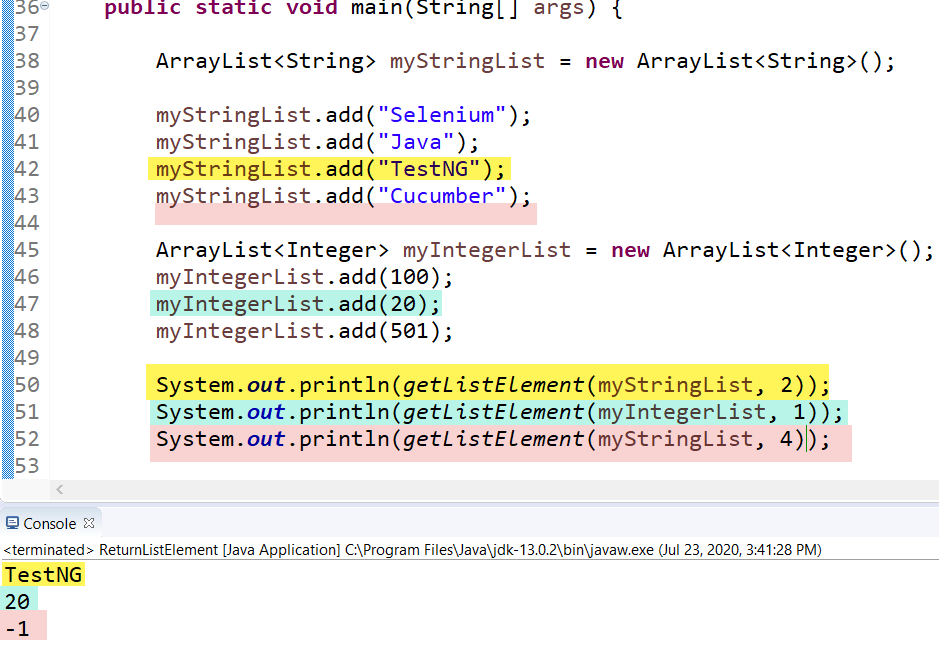
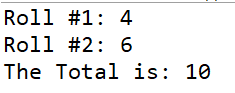
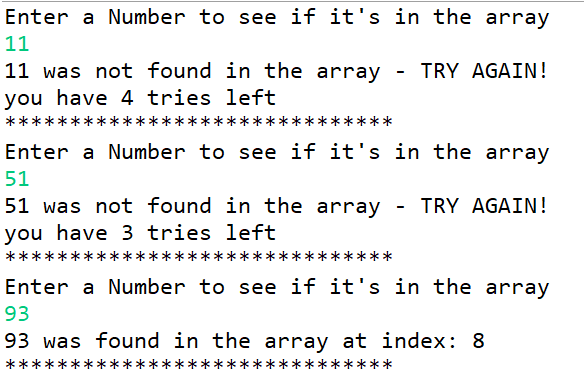
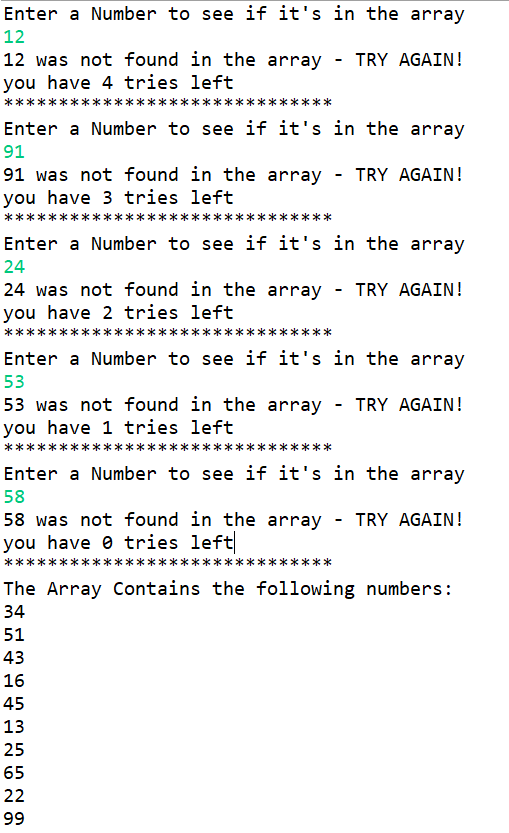
**Test Automation Bootcamp**

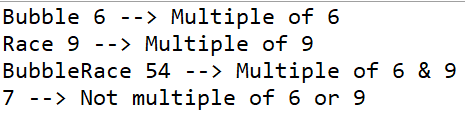
Java Coding Exercises

**Submission Instructions:** Please work on all of the coding exercises within Eclipse. Do not copy-paste the code into this or any other word document. You will be submitting all project code files via email or Google Drive folder. Instructions for submission were explained in class when we started the bootcamp and can also be found in the email sent by Jay.

Be sure to carefully read each exercise completely before beginning to work on it.

1. Write a method that returns the specified element of a list. If the provided index exceeds the list size, then return -1. The lists could contain values of any type. See the screenshot below for guidance. **Create 3 lists** 🡪 1 list of Strings, 1 list of Integers, 1 list of Doubles. Using the method that you’ve created, display a value from each of the 3 lists. Test the method to verify that *-1* is returned when the index provided exceeds the size of the list. ***Java Focus:*** Generics  
   
2. Create a program that will simulate the rolling of 2 dice. Create a method that will return a random number between 1 and 6. Call the method twice and record the results as show below. (Your numbers may be different as they should be generated randomly) ***Java Focus:*** Methods, Variables, Data Types  
   
3. Build a program that will create an array of 10 random numbers between 1-100. Then, the program should ask the user to guess a number that is in the array. If the user guesses correctly, then a message is printed saying “*guessedNumber* was found in the array at index: *indexValue*”. The user should have a maximum of 5 tries to guess a number in the array. Each time the user guesses an incorrect number (a number not in the array) they are presented with the following message: “*guessedNumber* was not found in the array – TRY AGAIN!”. Another message is printed telling the user how many tries they have left. Once the 5 tries have been used a message should be printed saying “The Array Contains the following numbers:”. And the array of numbers should then be printed out. **NOTE**: *guessedNumber* is the value provided by the user & *indexValue* is the index where the number was found in the array. Use the screenshots below to help you build this program. ***Java Focus:*** Arrays, Loops, Conditionals, simple logic  
     
   **Number was found in the array:** *You don’t need the \*\*\*\*\*\* in your output*

**Number was NOT found in the array:**

1. Write a non-static method that returns **‘Bubble’** for multiples of 6 and **‘Race’** for multiples of 9. For numbers which are multiples of both 6 & 9, return **‘BubbleRace’**. For numbers that are neither multiples of 6 nor multiples of 9, return the input number. Test the method by calling it in a different class from where it has been defined. Output should be as shown below. ***Java Focus***: Methods, Classes, Objects, Operators  
   
2. Create an HR program that will on-board new employees (add new employees to the system). You must create 2 classes. **Class1** must be used to contain all variables (employee’s name, employee’s department ID, employee’s job title, employee’s salary, employee’s bonus, employee’s PTO (paid time off)). It must also contain the following methods:
   1. Method to determine the PTO for an employee based on their assigned department ID. Employees of Department
      1. 101 (Sales) should be given 8.0 PTO hours.
      2. 201 (Accounting/finance) should be given 12.5 PTO hours.
      3. 301 (Marketing) should be given 16.0 PTO hours.
      4. Employees for all other departments should be given 4.0 PTO hours.
   2. Method to calculate Monthly Pay with Bonus (monthly salary + (monthly salary x bonus)). Monthly pay must be returned.
   3. Each time a new employee is added they must automatically be assigned a name, department ID, and Job Title. This information must be displayed as shown in the screenshot below.

Additionally, a counter must keep track of the number of employees currently added to the system (during the execution only).

**Class2** must be used to create (add) new employees and display their relevant details (as shown in the screenshot below).

**Keep in mind:**

* All variables must have a *private* access modifier
* The variable for bonus must be declared as type ***double***
* You may need to create additional methods to achieve the same output as shown below
* Your output must match ONLY the format as shown below (names, salaries, and Job Titles can be as you like)
* There should be 3 departments (101 🡪 Sales, 201 🡪 Accounting/Finance, & 301 🡪 Marketing)
* Create a minimum of **4 employees** – 1 for each of the departments and 1 additional employee that belongs to a different department

***Java Focus***: Encapsulation, getters/setters

**Your output should look similar to this & should include details for 4 employees**

