# Bootcamp Assignments

[Bootcamp Assignments](#_9g7nuc6602x5)

[Week 1](#_isd5iqw09fgi)

[Week 2](#_t7fqrkloz973)

[Week 3](#_f772upai8ky5)

[Week 4](#_th8yk962frml)

[Week 5](#_23m1zpit5y89)

[Week 6](#_6fscmhesfzni)

[Week 7](#_e1iv197ysbgk)

[Week 8](#_fqaph1pyb02p)

## Week 1

1. Identifiers >> Create a Java class for any one of the given components and create 15 names using the identifiers rule.   
   Component:   
   - University   
   - Company   
   - Country
2. Data Types >> Create and print values of boolean data types:
   1. String values
   2. boolean literals
   3. Implementation of Wrapper class (old + new structure)
3. Local/ Instance/ Static >> Create a class as per below template and pick any theme to implement it:   
     
   Methods:   
   - one main() method   
   - two non-static methods a() and b(), and   
   - one static method c().   
     
   Variables:   
   - Each non-static method should have one local variable.   
   - create two instance variables   
   - create one static variable   
     
   Object Creation:   
   - Create two objects in main() method   
   - assign separate values to instance variables   
   - print local, instance, static values for both the objects

## Week 2

1. Access Modifiers >> Create an application for a Company with following components:

* Company Details
* Product Details
* Employee Details
* Client Details

**What you need to do?**

* Decide packaging structure based on the components and it's variables.
* Add Meaningful instance variable in all the classes.
* Create object of one class in another class and show only information which is required for that class.
* Examples:
* Client will not be interested in knowing about who is an HR of that company.
* Client will be more interested to see the company details and the product details
* Client should not have access to the sensitive information of the company
* Client should only see all the live products or upcoming products. Client should not see all products under experiment or brainstorming.

1. Class and Object >> Create a SummerCamp Registration form where students from all over the world can apply.

**Store some basic information like:**

* Student Name
* Age
* Nationality
* Parent/Guardian Name
* Parent/Guardian phone number
* Attended any camp before
* If yes, get camp name
* Total Registration Application Count

**What you need to do?**

* Create few registration objects and store all the value using parametrized method.
  + Use one method just to store Student Information
  + Use another method to store Parent Information
* Once an object is created, print all its value in the console using a common method.
* Once you have displayed all the registration information, Display total number of registrations done so far.

**Tip:** Add comments on top of method/objects to make your code more readable.

1. Method Return Type >> Create a class and create one method using different values of primitive return type. Use that return value to print it in the console.
2. Class and Object >> Create assignment 2 with Constructors
3. Operators >> If a student has scored 80, 75, and 62 in subjects out of 300 then write a program to calculate total marks and percentage.
4. Operators >> Find remainder when 100 is divided by 9 and print output by adding 2 to the remainder.
5. Operators >> Find max of 3 numbers using conditional Operator
6. Operators (Optional) >> Research on Operator Precedence
7. If Condition >> Using if-else Find grade of a student who scored some marks out of 100 - if marks >75 → Result is Distinction   
   - if marks >60 → Result is Grade A   
   - if marks >50 → Result is Grade B   
   - if marks > =35 → Result is Grade C   
   - for marks below 35 → Result is Fail

## Week 3

1. Switch Case >> Using switch case Write a code to print Diet plan for the given day:   
   - Monday and Wednesday should have same meal plan   
   - Saturday and Sunday should have same meal plan   
   - Tuesday, Thursday and Friday should have a unique plan.
2. while/do-while >> Using while and do-while print sum of numbers from 1 to 5. Print sum of numbers after while loop.
3. for-each>> Using for-each loop, Print only prime numbers from 1 to 20
4. For loop >> Using Nested for loop create pyramid pattern

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

1. Inheritance IS-A >> Create below classes and create a parent child relationship using them.

- Employee

- Teacher

- Student

- Person

Points to Remember:

- Store all the common attributes in one class.

- Store class specific attributes/methods in respective class.

- Create objects in below combination and use attributes/method and print the values **Combination to try**:

- Child c = new Child();

- Parent p = new Parent();

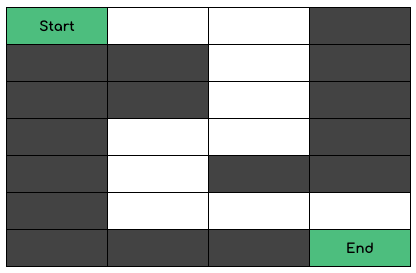
- Parent p = new Child();

- Child c = new Parent();

1. Implement Inheritance HAS-A and super (only with IS)>> Add any class to the above structure that can have HAS-A relationship.   
   - Add few parameters and method to the class.   
   - Assign and print the values from any child class.
2. Polymorphism >> Write a few overloaded method in any child class and write few overridden methods from parent in the child class. You can create any method based on the class you're working on. Try to use this, super, final concepts in this assignment. Create a separate class to print the values.

## Week 4

1. Interface >> Create an Interface named Bank. Write down a few common set of services that all banks should follow. Write down method implementations in all the implementing classes. Create 2-3 classes that will implement a Bank interface.
2. Encapsulation >>   
   - Create an Encapsulated class to store all Testcase Related Information(Can refer Testcase Execution assignment for this).   
   - Create a separate class to set all the values and write one method to print all the testcase details.   
   **Note**: Testcase details should be printed only when the testcase name is not null.
3. Enum >> Create an enum to store browser name that you can support with automation.   
   - Create a separate class with main method to trigger the execution.   
   - Create browser specific method in the same class   
   - Create a method that will accept enum constant and based on the constant value, call browser specific methods.
4. Varargs >> Solve this puzzle. Use String format for navigation, follow the white path
   1. Up
   2. Down
   3. Left
   4. Right



1. Arrays >> Given an array of ints length, return the sum of all the elements.
   1. 1, 8, 3 → 12
   2. 5, -5, 2 → 2
   3. 7, 0, -10 → -3
2. Arrays >> Given an int array, return a new array with double the length where its last element is the same as the original array, and all the other elements are 0. The original array will be length 1 or more. Note: by default, a new int array contains all 0's.

- getLast([0, 3, 5]) → [0, 0, 0, 0, 0, 5]

- getLast([-1, 0]) → [0, 0, 0, 0]

- getLast([9]) → [0, 9]

**Hint:** use variable length to determine length of input array.

1. Strings length>> Given a string of even length, return a string made of the middle two chars, so the string "string" yields "ri".   
   The string length will be at least 2.   
   - middleTwo("string") → "ri"   
   - middleTwo("code") → "od"   
   - middleTwo("Practice") → "ct"
2. Strings concat >> Given two strings, append them together (known as "concatenation") and return the result. However, if the concatenation creates a double-char, then omit one of the chars, so "abc" and "cat" yields "abcat".   
   - conCat("abc", "cat") → "abcat"   
   - conCat("dog", "cat") → "dogcat"   
   - conCat("abc", "") → "abc"
3. String substring >> Given a string, if a length 2 substring appears at both its beginning and end, return a string without the substring at the beginning, so "HelloHe" yields "lloHe". The substring may overlap with itself, so "Hi" yields "". Otherwise, return the original string unchanged.   
   - without2("HelloHe") → "lloHe"   
   - without2("HelloHi") → "HelloHi"   
   - without2("Hi") → ""
4. Exception Handling >> Create a class to store Passenger Ticket Details for Railways.   
   - If passenger age is more than 65 then assign lower berth to them.   
   - If age is less than 5 then assign lower berth to them as well.   
   - For any other age between 5 and 65, allocate upper berth   
   **Validations:**   
   - Passenger with age range from 6 to 64 cannot have lower berth. If lower berth is assigned to them then throw TooYoungException   
   - Passenger with age more than 65 cannot have upper birth. If upper berth is assigned then throw TooOldException.  
   - Passenger name should not be null, if it is null then update name to Unknown from catch block.

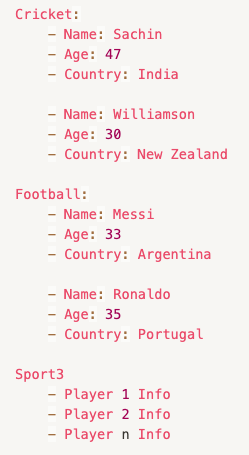
- Print passenger details from the finally block.

Note: Please try to use all the concepts that we have covered so far.

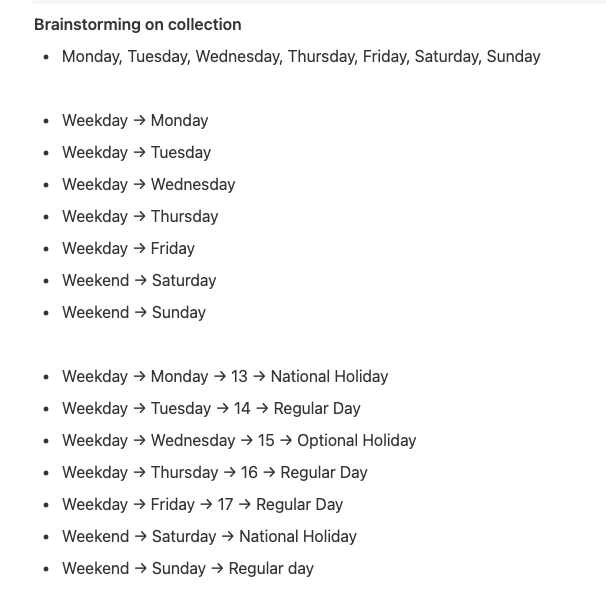
## Week 5

1. Collection >> Create a sports management application where you store Player information. All players who play the same sport should be grouped together. Also, share image representation of the output.

Refer Image on Next Page



1. For Personal Brainstorming >> Submit data structure on slack.



## Week 6

1. Element Interaction >> Navigate to [SkyScanner website](https://www.skyscanner.com.sg/) and perform below tasks:

* Look for One Way flight ticket
* Book ticket from Dubai to Singapore (Can select any airport in Dubai)
* Select Departure Date
* 3 Adults and 2 Children travelling in a Business Class
* Look only for Direct Flights
* Print Departure and Arrival Time of all flights

1. Find Elements >> Navigate to<http://demo.automationtesting.in/Register.html> and select Hobbies based on the hobby name provided by another method. If requested hobby is not present then print a message saying “Hobby $hobbyName is not part of the checklist”

## Week 7

1. Dynamic Tables: Navigate to<https://demoqa.com/webtables>

* Print all the available data on console
* Add 2 New Data rows
* Search for the data and print search result
* Delete any one created data
* Verify if data is available in result after delete
* Verify 4 rows remaining

1. Window Handling: Navigate to<http://demo.automationtesting.in/Windows.html>

* Automate behaviour for all three types:
* New Tab Window
* Separate Window
* Separate Multiple Window

1. DragAndDrop: Navigate to<http://demo.automationtesting.in/Static.html>

* Perform Drag and drop operation on all three images

1. Waits: Navigate to<http://demo.automationtesting.in/DynamicData.html>

* Click on Get Dynamic data and wait until elements are displayed
* Print values of FirstName and last name and verify if person image is present on the page
* Try with:
  + Implicit
  + Explicit
  + Fluent

## 

## 

## Week 8

**Test Login with Multiple Users**

* Create a class with @Test method to perform login Operation on<http://automationpractice.com/index.php?controller=authentication>
* Use Data Provider to pass data to the test method
* Store below Information for the User:
  + firstName
  + lastName
  + email
  + password
  + isActive == false -> do not use this user for execution
  + errorMessage
* Perform login only when isActive=true
* On Failed Login:
  + Capture the error message from UI and compare it with expected error message using Assertions.

**Test case Creation for Item Searching: TestNG + Selenium**

* Create a Testcase to search for multiple products on<https://demo.nopcommerce.com/>
* Search for 3 different products and add them to the cart
* Verify Items present in cart after adding each item
  + Add item 1 → total items in cart = 1
  + Add item 2 → total items in cart = 2 and so on
* Verify Item name and price listed on Item listing page matches with content present in the cart
* Verify total amount of 3 items from listing page matches with total amount displayed in the cart.

**Tips:**

* Use Assertions to decide if testcase is pass or fail
* Get item names to search from an external file(Excel/JSON)
* Use properties file to decide on which browser to execute