**Cognixia Java Workshop June 2021**

**Class Notes**

**Summary Day 1:**

1. **Need to revise Access Modifiers**
2. **Accessing input parameters from command line**

**Day 2 - 01 June 2021**

**Recap Day 1:**

1. Introduction to Java
   1. Class
   2. Object
      1. **Properties** that describe the characteristics of that object
      2. **Behaviours** that define the actions performed by that object
   3. Method
   4. Keywords
2. Data Types
3. Int - 2 ^ 31
4. Byte - 2 ^ 7
5. Long - 2 ^ 63
6. Short - 2 ^ 15
7. Double - 2 ^ 63
8. Float - 2 ^ 31
9. Char - 2 ^ 16 = 65,536
10. Boolean - 1
11. Wrapper Classes
    1. Why is it required?
       1. Convert to other data types
       2. Using data types as objects
12. **String Manipulations**
13. Length
14. Concat
15. Compare with case sensitive or insensitive
16. Starts-with
17. Ends-with
18. **Access Modifiers**
    1. Private
       1. Can only access within the same class
    2. Public
       1. The whole world can access
    3. Protected
       1. Same class OR sub-class OR within package
    4. Default
       1. Same class and package
19. Packages
    1. When we have 2 packages
       1. we can access the classes in other package only after importing
20. Final
21. Variables
    1. Local
       1. Cannot be made static
       2. Declared inside a method or a block (if else) and can be accessed only inside the block
       3. Always private (No access modifiers required)
       4. They always have to be initialized
    2. Instance Variables
       1. Declared inside class, outside any method or block.
       2. Has a default value
          1. Int - 0
          2. Boolean - false
          3. String - null
          4. Double / float - 0.
       3. Also called global variables
       4. Have Access modifier
    3. Static
       1. Permanently store anything - CONSTANTS
       2. Use static method or property without creating an object
       3. Static variables are used as counters in programs
22. Constructors
    1. Always has the same name as class
    2. It does not have a return type
    3. It gets called when Object is created, hence used to initialize properties for an object
23. Conditional Statements
    1. IF Else
       1. For decision making statements
       2. Conditions always evaluate to a boolean value
       3. Else if can extend the conditions (nested if)
    2. Switch Case
       1. Used when there are multiple decision points for same data value

**Access Modifiers:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Modifier | Same Class | Different Class Same Package | Sub Class Same Package | Difference Class Difference Package | Sub Class Different Package |
| Private | YES | NO | NO | NO | NO |
| Protected | YES | YES | YES | NO | NO |
| Public | YES | YES | YES | YES | YES |
| Default | YES | YES | YES | NO | NO |

**Summary Day 2:**

Pending Questions:

1. If we import a class, then why do we need to extend it. Difference between importing and inheritance?
2. **Programming from scratch**
3. **Methods with parameters**
4. **ArrayListExample**
5. User Input in Array

**Day 3 - June 2, 2021**

**Recap of Day 2**

1. Access Modifiers Revision
2. Command Prompt Execution
3. Loops
   1. While
   2. Do-While
   3. For
4. Collections
   1. Arrays
   2. ArrayList

**Summary of Day 3:**

1. If we import a class, then why do we need to extend it. Difference between importing and inheritance?
2. User Input in Array
3. Iterator
4. Sorting of Collection by names

**Day 4:**

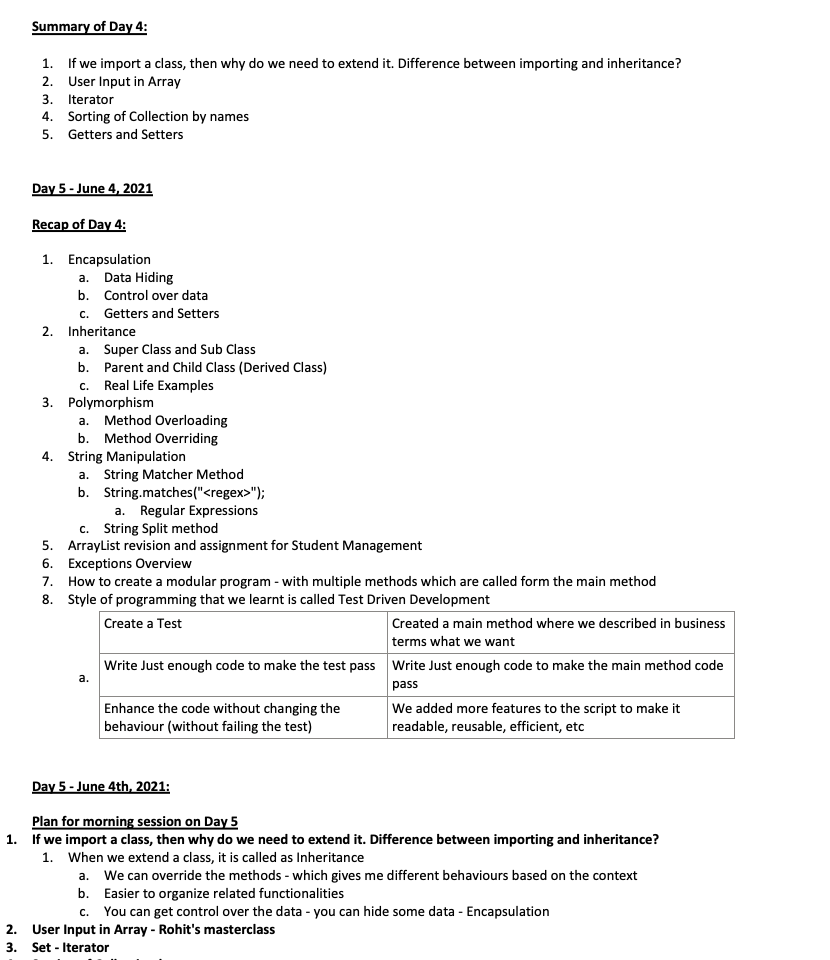
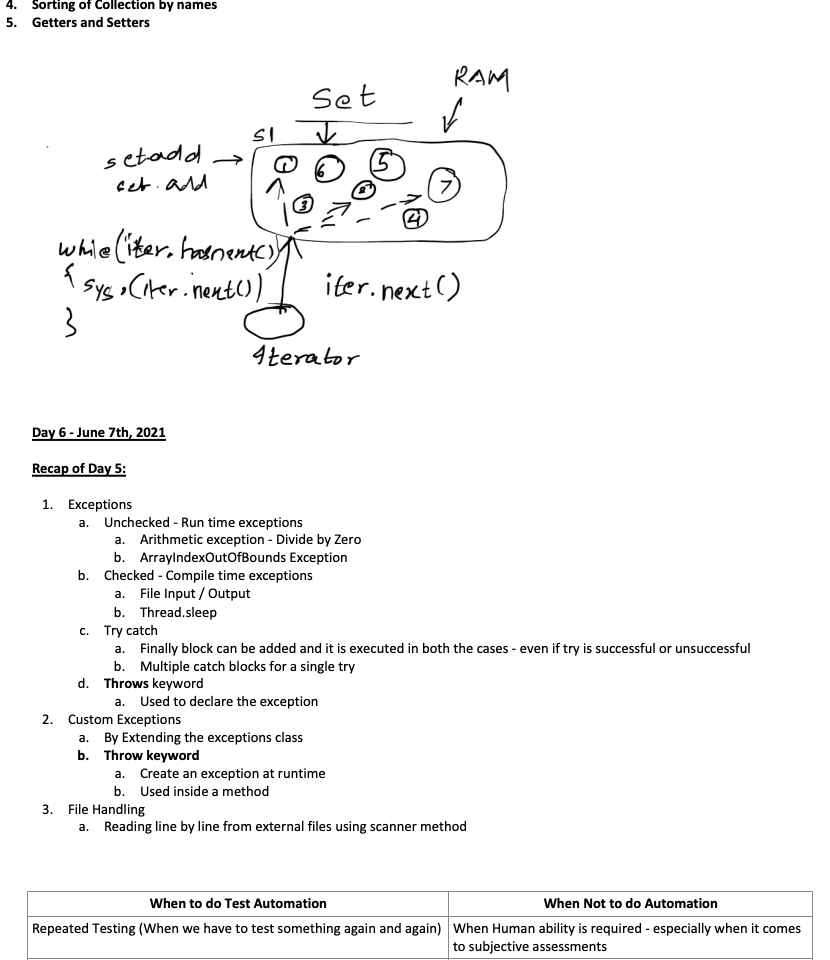
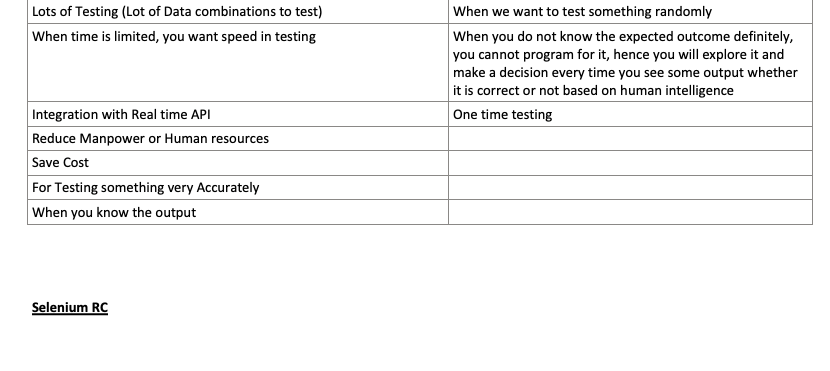
**Recap of Day 3:**

1. List - ArrayList
2. Set - HashSet
   1. Does not allow Duplicates
   2. Does not store values in order of entry
   3. Cannot access element by index
3. Packages
   1. Organize the code in groupings of like-classes together
   2. Same names for classes are possible in different packages
   3. Simplifies codebase (especially when it becomes very large)
4. Interface
   1. Interface is a contract between 2 parties
   2. It helps in architecting and planning of development activities
   3. It is a blueprint of class
   4. Defines the behaviour of the object
   5. Multiple classes can implement the same interface
   6. Single class can implement multiple interfaces - Multiple inheritance using interfaces
   7. **Interface allows complete abstraction in Java**
   8. We can create an reference of object to interface once it is implemented by a class. E.g. - Vehicle v = new Car();
   9. The implementing class has to mandatorily implement all the methods defined in the interface.
   10. Overriding of all the methods
5. Arrangement of Classes - Creating multiple functions for the activities and calling them from main method as well from each other.
   1. All the logic will be in methods / functions
   2. The main method will only describe the workflow (steps)
   3. The methods become reusable
   4. The script is more readable

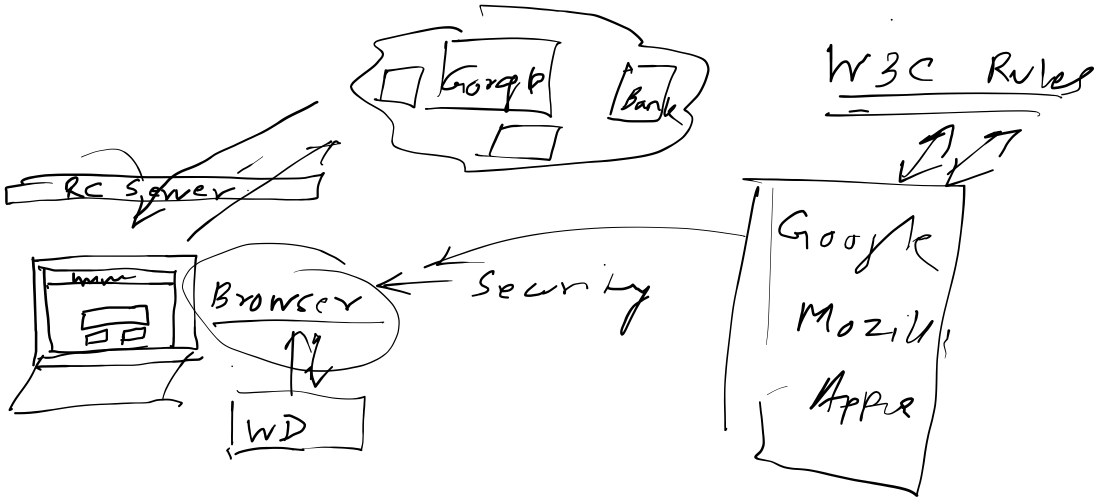
 Regex Character classes

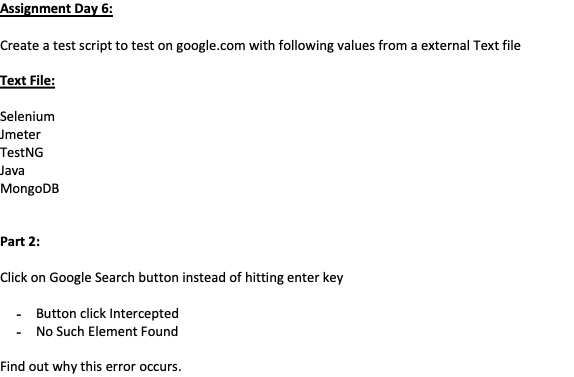
|  |  |  |
| --- | --- | --- |
| **No.** | **Character Class** | **Description** |
| 1 | [abc] | a, b, or c (simple class) |
| 2 | [^abc] | Any character except a, b, or c (negation) |
| 3 | [a-zA-Z] | a through z or A through Z, inclusive (range) |
| 4 | [a-d[m-p]] | a through d, or m through p: [a-dm-p] (union) |
| 5 | [a-z&&[def]] | d, e, or f (intersection) |
| 6 | [a-z&&[^bc]] | a through z, except for b and c: [ad-z] (subtraction) |
| 7 | [a-z&&[^m-p]] | a through z, and not m through p: [a-lq-z](subtraction) |

Credits: <https://www.javatpoint.com/>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | | | | |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | | |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | | |  |  |  |  |  |
|  |  |  | | |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |





**Day 7 - June 8th, 2021**

**Recap Day 6:**

1. Criteria for Automation
   1. When to Automate
   2. When Not to
2. Software Testing
   1. Actual vs. Expected
   2. Trial to assess the genuineness of the software (program / instructions) and its output by experimentation, investigation, exploration, verification, etc.
3. Open Source vs Closed Source
4. History of Selenium
   1. What was the problem
   2. What was the solution
      1. Selenium Core - Javascript based library
      2. Selenium RC - WebServer intercept between browser and actual server
      3. Selenium WebDriver - Does not use the WebServer
5. Selenium WebDriver
   1. WebDriver Interface
   2. Various Browser Drivers which implement the interface
   3. Locators
      1. Id
      2. Name
      3. className
      4. tagName
      5. linkText
      6. partialLinkText
      7. XPath
      8. CSS Selectors
   4. Methods of Selenium
      1. Get
      2. navigate().to()
      3. sendKeys
      4. Navigate().back, forward, refresh
      5. Quit, close
      6. Submit
      7. Click()
      8. **findElement(By.name())**
      9. driver.getTitle()
      10. System.setProperty("webdriver.chrome.driver","<path to chromedriver.exe>")
      11. System.setProperty("webdriver.gecko.driver","<path to geckodriver.exe>")
      12. System.setProperty("webdriver.edge.driver","<path to msedgedriver.exe>")
      13. System.setProperty("webdriver.ie.driver","<path to internetexplorerdriver.exe>")
   5. Java Project Buildpath
      1. Add External JARs - Selenium JAR
   6. Thread.sleep() - used this for Synchronization as of now. But we also know that this is not the best way to wait for the browser to load, etc.

**Assignment Day 7 : Automate the following Test Case using Selenium**

Test Case:

1. Go to flipkart.com
2. Close the Login Popup
3. Search for iphone
4. Verify that there are 285 models of iphones available on flipkart

**What did we learn:**

1. How to find elements in hierarchy (parent - child) (Webelement.findElement)
2. How to get index of a particular word from a string
3. How to get a substring from a string using starting and ending index

**Day 8: June 9th, 2021**

**Recap of Day 7:**

1. XPath
2. Synchronization
   1. Thread.sleep
   2. Implicitly Wait
      1. It waits for the element until the element is found or the timeout given whichever is earlier.
3. FindElements (to get multiple elements in a List)
4. How to find element using the hierarchy
5. The special ForEach method using Lambda expression
6. String manipulation - get a substring

**To Revise:**

1. Where to use Implicit Wait - Done
2. Xpath from scratch (Relative XPath) - Done

**Day 9: June 10th, 2021**

**To Revise:**

1. **Difference between Implicit and explicit wait**
   1. **Can we use it in a single program?** 
      1. **Just Explicit wait was not working, but when implicit was added it started working**
2. Writing Relative XPath from scratch
3. Xpath Functions - Contains, text, etc
4. **Errors in script for Rupa, others - Flipkart Suggestions**
5. Verify NOT contains in XPath

**Tips:**

1. Press ctrl+F11 to run the eclipse program

**Recap for Day 8:**

1. XPath Revision
   1. Absolute
   2. Relative
   3. How to create xpath in various ways
   4. When to use absolute xpath
2. XPath Functions
   1. Text()
   2. Contains(s1,s2)
   3. Starts-with(s1,s2)
   4. Last()
   5. Position()
3. XPath Operators
   1. AND
   2. OR
   3. =
   4. !=
   5. <
   6. >
4. Explicit Wait
   1. Explicit wait is used to WAIT to full fill the specific condition on any element
5. String Manipulations
   1. Replace()
   2. Converting String to Double
   3. Convert long to string
6. ArrayList
   1. Indexof()

//Implicit Wait is only worried about finding the element. (presence of the element)

//Implicit Wait does NOT care about what is Inside the element,

//e.g. it does not care about what is the text in the element

//Implicit wait also does not care about HOW MANY such elements are present

//As soon as ONE element becomes present, Implicit wait will resolve and proceed

//Explicit Wait

//Waits for some property of the element

//presence of element is just 1 property

//some other properties are:

//The text inside element

//The number of elements

//The @name attribute of element should have value

//The color of the element

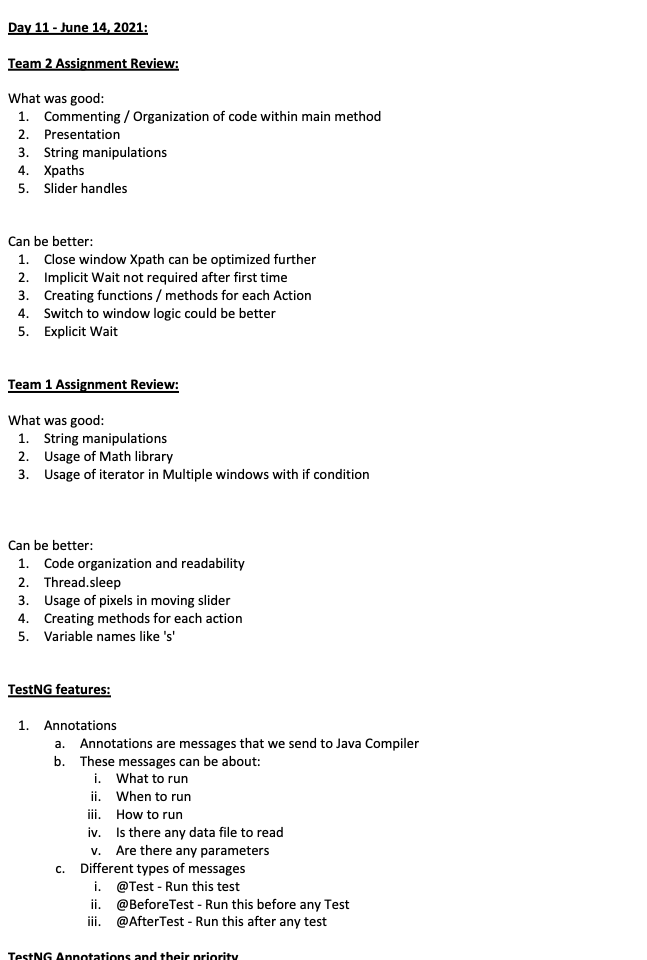
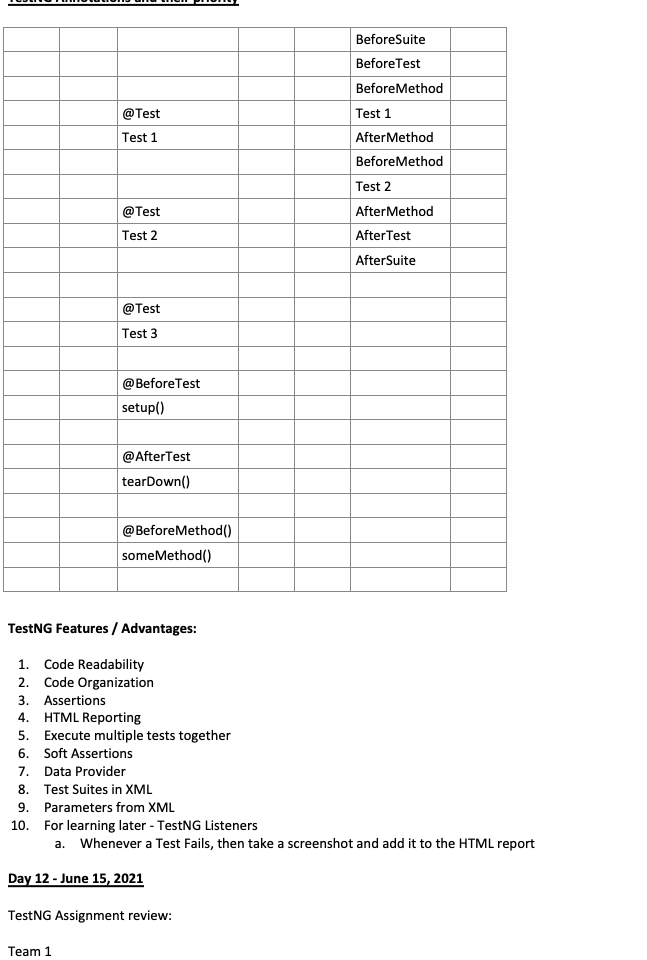
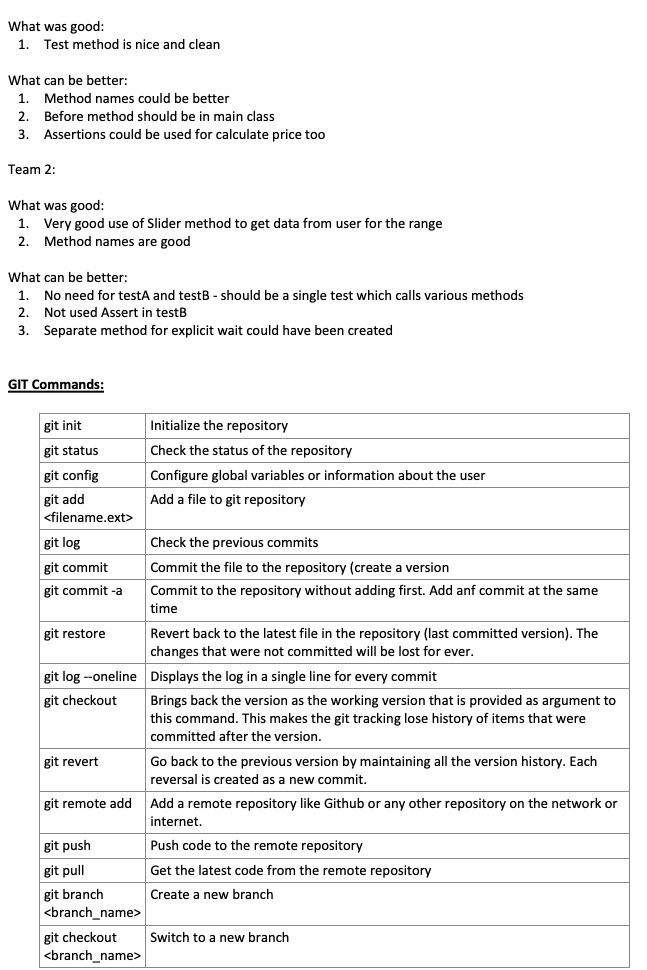
**Day 10 - June 11, 2021**

**To revise:**

1. **Verify NOT contains in XPath - Done**
2. Handling multiple windows

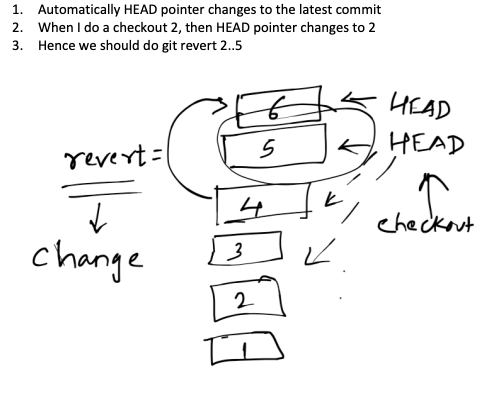
**Recap of Day 9:**

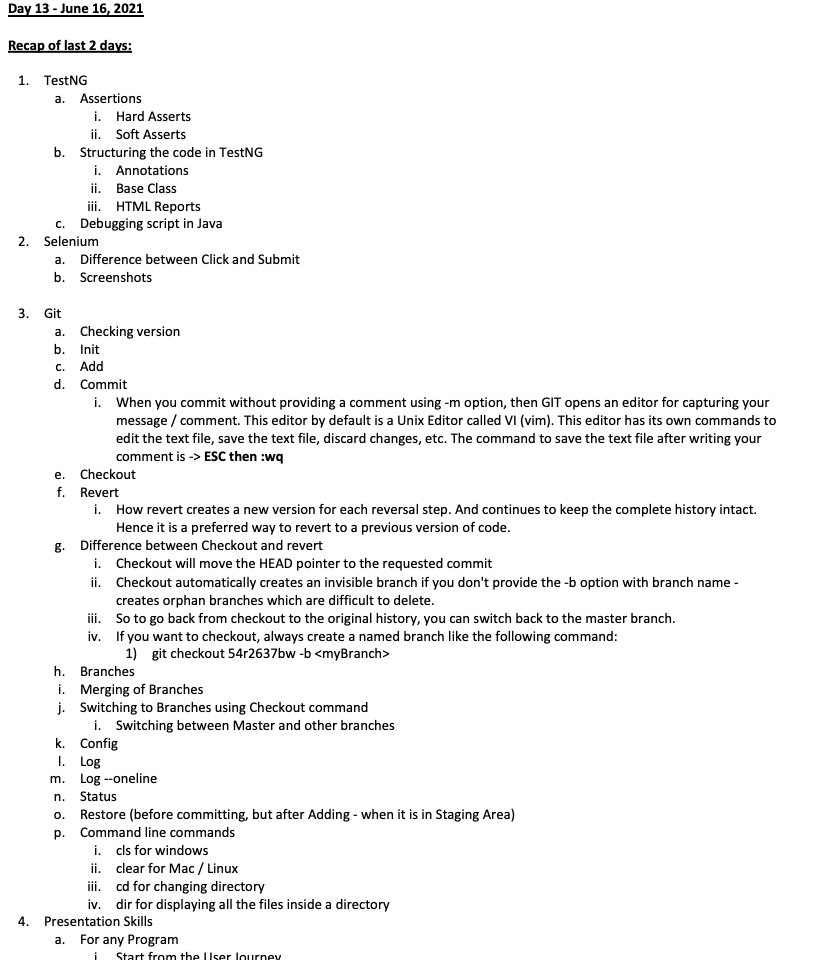
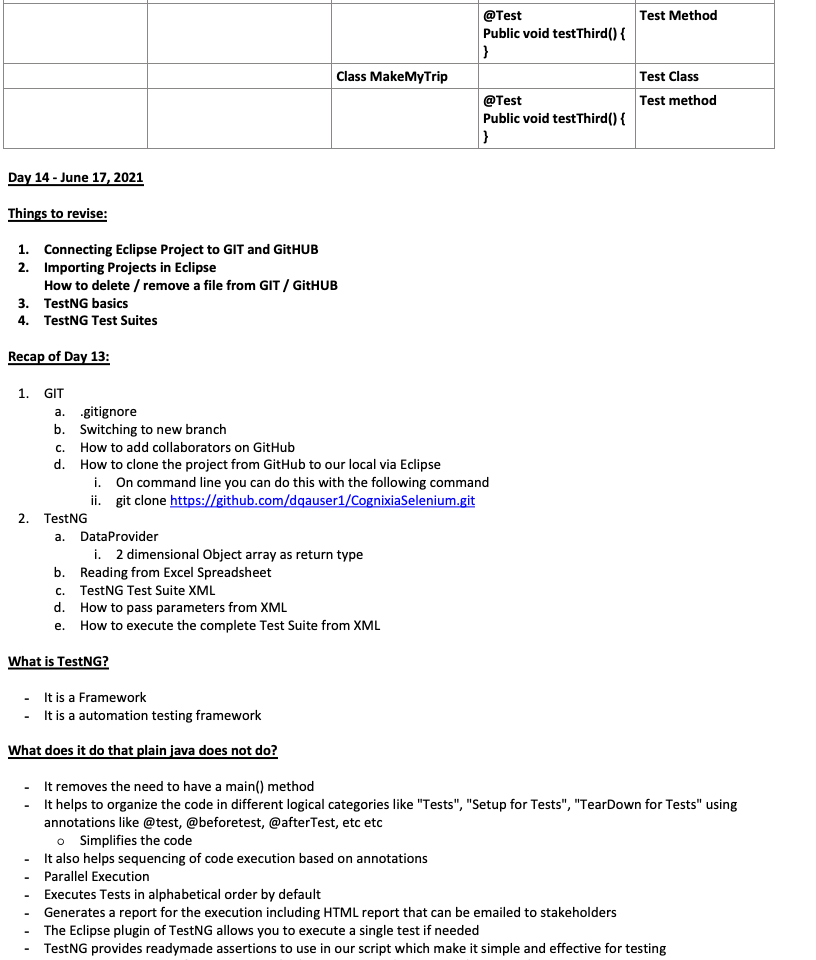
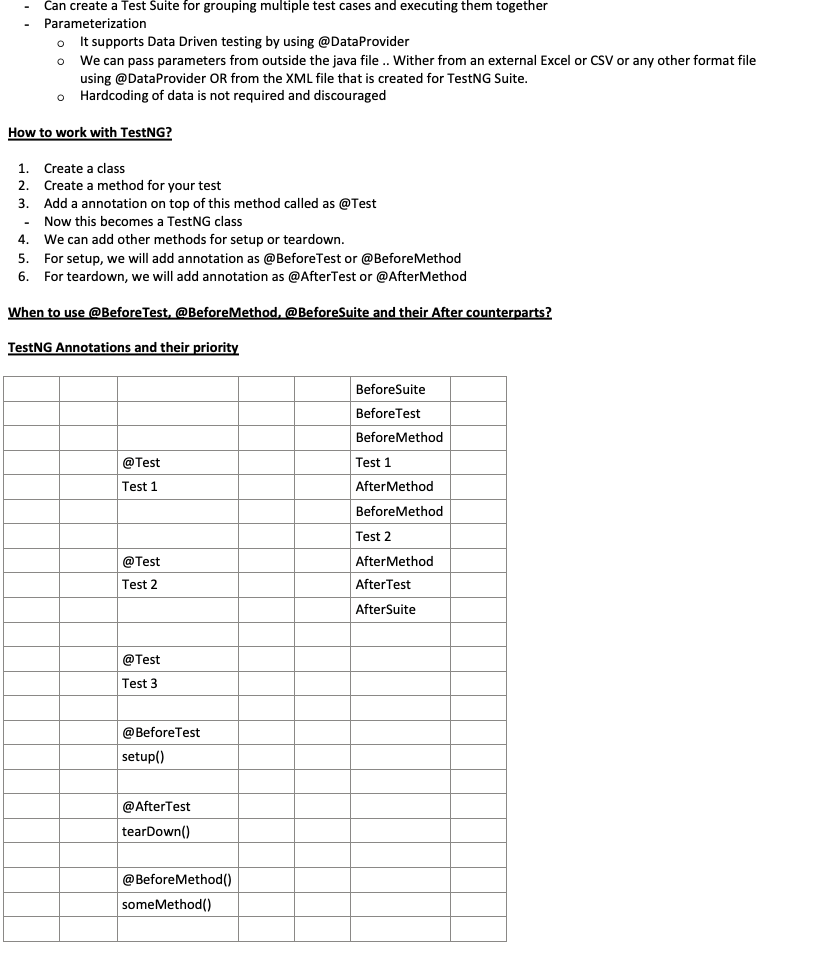
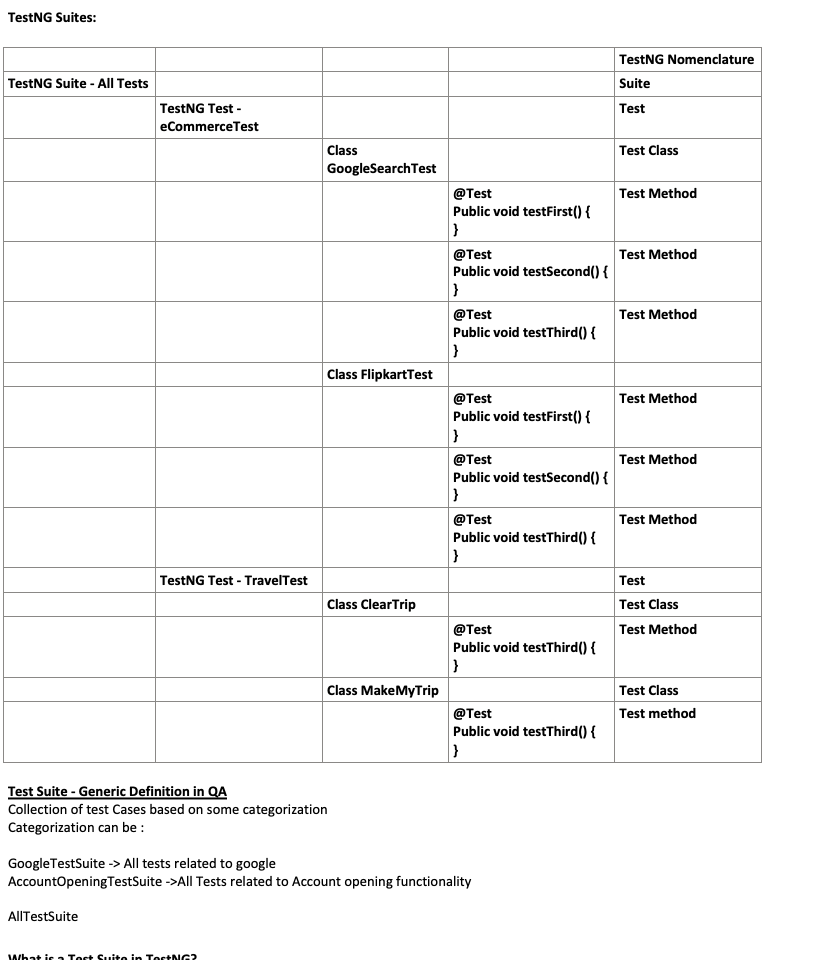
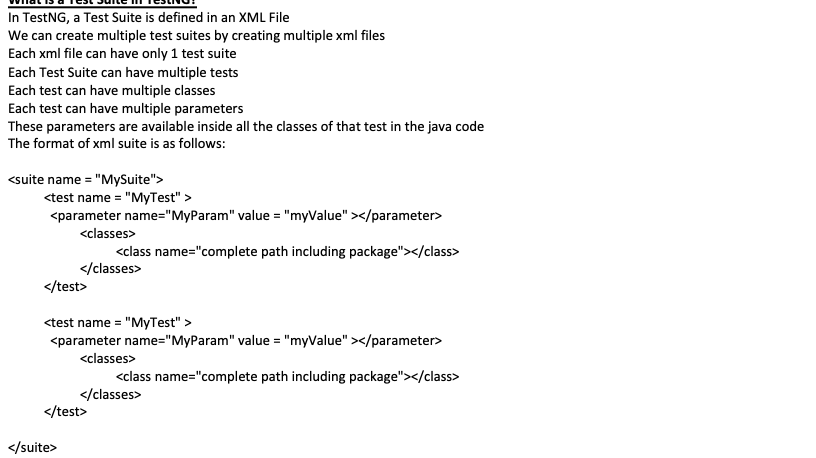
1. How to take a screenshot
2. Handling Multiple Windows
3. Handling Dropdowns with Select Class
4. Handling popups with Alert feature
5. How to deal with dynamic locators
6. String Manipulation
   1. Get the index for any character or string within the string
7. Implicit Wait vs Explicit Wait

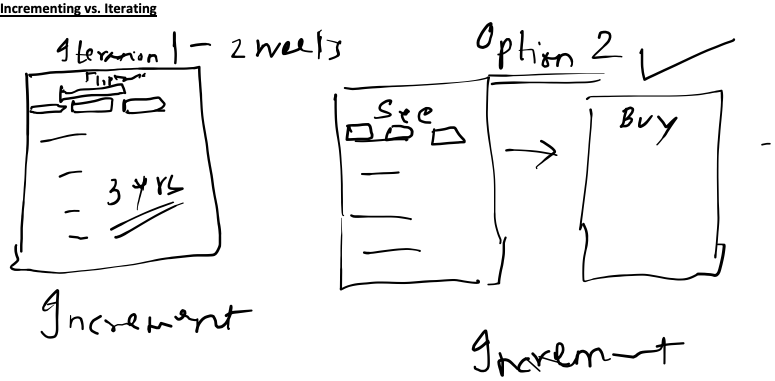
  
  


**How Git Revert works / Difference between Git checkout and Git Revert**

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Comment | Position of Head  in case of checkout | Position of head  In case of revert |
|  |  |  |  |
| 8 | Reversal to second |  | <--HEAD Goes from 3 to 2 |
| 7 | Reversal to third |  | Goes from 4 to 3 |
| 6 | Reversal of fourth |  | Goes from 5 to 4 and creates a new version |
| 5 | Fifth Commit | orphan | <-- OLD\_HEAD this is where you started with revert command |
| 4 | Fourth Commit | orphan |  |
| 3 | Third Commit | orphan |  |
| 2 | Second commit | <-- Head |  |
| 1 | First commit |  |  |





|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  | | |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |  |

**Day 15 - June 18, 2021**

Fundamentals of Performance Testing

How to find realistic Response time for Flipkart home Page?

1. Delete Cache
2. Execute 3-4 times and take Average
3. Execute on different devices and OSes and different browsers for 3-4 times each and take average
4. Simulate 10000 (avg. number) users accessing Flipkart at the same time, and in this situation, execute the test on multiple devices for 3-4 times and take average
5. Access from different Networks - Different ISPs and different speed bands
6. Access from different locations or geographies for 10000 users using a variety of devices, Oses and browsers, and testing for 3-4 times and then taking average

**Day 16, June 21, 2021**

**Things that we have learnt in the workshop so far:**

1. Java
2. Selenium
3. Maven
4. TestNG
5. Git and GitHub
6. SDLC Models
7. Quality Assurance Fundamentals and Mindset
8. Agile
9. Performance Testing Fundamentals
10. JMeter

**Things to revise**

1. JMeter
2. Difference between GIT and GITHUB

**Plan for this week:**

1. Finish JMeter learning - today
2. Learn MongoDB and its usage for testing - tomorrow
3. Projects
   1. A
   2. B
   3. C
4. Interview Tips
5. More Projects
   1. D
   2. E
   3. F

**Day 17, June 22, 2021**

JMeter Assertions and How they work

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Transaction** | **Requests** | **Responses** | **Assertions** | **Remarks** |
| Open Google |  |  |  |  |
|  | /open url |  |  |  |
|  |  | /html | assertion | Main sample only - This will apply to only openurl |
|  |  | /image |  | Sub samples - This will apply to /image and /some other files also |
|  |  | /some other files, etc |  |  |
|  | assertion |  |  | Main sample only - This will apply to /open url and /search and their main samples only |
|  |  |  |  | **Main and Sub Samples** - IT will apply to main and sub samples of all the responses |
|  | /search |  |  | **Sub Samples only** - IT will apply to main and sub samples of all the responses |
|  |  | /html |  | So, for this location, the above 2 are one and then same |

**Recap of Day 16**

1. JMeter
   1. How to load images or other resources
2. Recorder
   1. Add the Certificate to browser
   2. Set the Proxy to localhost:8888
   3. Named the transactions while recording before doing the next action
   4. Used the prefix filter to name the requests inside the recording
   5. Counter to number the requests within the transaction
3. Transactions
   1. Requests
   2. Generate Requests for parent sample - This will give us the aggregate response time for the transaction and not only the requests.
4. Assertions
   1. Xpath Assertion
   2. Response Assertion
   3. HTML Assertion
   4. Duration
   5. Size
   6. Status Code
   7. Status Message
5. Listeners
   1. Results
      1. View Results Tree
      2. View Result in Table
      3. Aggregate Report
      4. Summary Report
   2. Exporting Results in a external file
   3. Mailer
6. Graphs
   1. Response Time Graph
   2. Aggregate Graph
7. Timers
   1. Constant Timer
   2. Uniform Timer
8. Parameterization
   1. Created a User Variable
   2. Create a variable by extracting response data
9. **Correlation - This is the most critical concept in scripting for performance**
   1. Extract data from one request and pass it as a input to another request
10. Threads
    1. No of Users
    2. Ramp Up time
    3. Iterations
11. Metrics
    1. Response Time
       1. Average
       2. Mean
       3. **90% line - Is most widely used as a indicator of performance in the industry**
       4. 95%
    2. Error %

**Things to revise over the next couple of days**

1. **Difference between GIT and GITHUB**
2. JMeter Parameterization and Correlation
3. **Selenium**
4. **Maven**
5. **MongoDB update method using $set: keyword**
6. **MongoDB Aggregation concept**
7. **MongoDB with Java code**

Day 19 - June 24, 2021

Presentations by Teams:

Team 1 - Cleartrip Project:

1. Very well organized code. Good breakup into functions.

Team 2 - AutomationPractice Project

1. Very well organized code
2. Line 54 - Faded Short Sleeve T-Shirt

Common Feedback:

* Select method not used by both teams

**Overview of Maven:**

**What is Maven:**

* Maven is a Project Build Lifecycle tool. It automates the build lifecycle.
* It organizes the **project workflow** and assets
* It helps in **building the project / Compiling the java files into class files**
* It helps in organizing the code - It has a default organization template. But we can also change these templates. We can use from a library of readymade templates or create our own template.
* In Maven, template is known as '**Archetype**'
* Maven works on the principle of **Convention** over configuration
* It helps in downloading the **dependencies**
* It can generate reports
* It can package the code in JAR file or ZIP file and automatically copy it on the server
* **It does all of this using a configuration file called as POM.xml**

**How to use Maven in your project?**

* Download and install the Maven Plugin for Eclipse
* Create a new Project using Maven template
* A POM.xml file will be automatically created.
* We need to update this POM.xml to start using Maven's capabilities
* We can also convert an existing non-maven project into a maven project by going to the right click menu and selecting maven->Convert to Maven
  + This will generate a new POM.xml

**Overview of Selenium:**

**What is Selenium:**

* It is a Software Web Test Automation Engine
* It is a Open source tool
* It supports multiple languages to create the code like Java, C#, PHP, Perl, Javascript, Ruby, python, etc
* It can automate Web Pages (HTML)
* Selenium GRID can help execute tests in parallel
* It supports all leading browsers - Chrome, Firefox, Opera, Safari, Edge, IE, etc
* It supports multiple Oss - Windows, Mac, Linux

**How can you identify elements on a webpage in Selenium?:**

* Using various **locators:**
  + Id
  + Name
  + ClassName
  + LinkText
  + TagName
  + XPath
    - Has more features than CSS
    - Is more robust than CSS
  + CSS
    - Is more performant than Xpath
    - Some people feel that the syntax of CSS is easier
    - It is preferred if there are many classnames as identifiers in your application

**What are the different Synchronization Strategies in Selenium?**

* **Implicit Wait**
  + Strategy where Selenium waits for the element to be present on the webpage for a maximum about of time specified. If the element is found earlier than this time, then selenium immediately proceeds to next command. It does NOT wait for the complete time interval.
* **Explicit Wait**
  + Strategy where Selenium can wait for a particular condition to be met - this is beyond just waiting for the element to be present - but this is about the various attributes of the element and their values - e.g. we can wait for the text of the element to be equal to something OR we can wait for the number of elements on the page to be equal to or greater than some number OR we can wait for the title of the page to become something.
  + There are many readymade Conditions provided by Selenium in the Expected Conditions class.
  + But if you want, you can define your own Condition for waiting.
  + For dong this you have to implement the FluentWait interface in Selenium
* **Why should Thread.sleep not be used?** 
  + It’s a static wait.
  + We cannot provide a condition of success
  + Hence we have to wait for the time
  + Even after this, we cannot be sure that the script will execute successfully

**What are some of the other features of Selenium?**

* Mouse Actions
* Select API (For dropdowns)
* Multiple Window Handling
* PopUp handling (JavaScript Alerts) - Alerts Class
* *Selenium GRID*

**Overview of GIT and GITHUB**

**What is GIT?**

* It is a version control system
* It is open source
* It uses the **Distributed** Version control method
  + Distributed means that - Everyone in the team has access to the complete version history.
  + All code can be available on all machines
    - Redundancy (Replication)
  + Makes code accessibility faster
  + It can make the storage requirement higher
* Tracks code history
* Revert to any version of code
* Create Branches and work separately and then again merge the branches to work together

**What is GITHUB?**

* It is not a version control system
* It is a service or repository which supports version control systems like GIT
* It can store the version history and allow GIT to run commands on the repository
* It is useful to share code on the web across team members
* Helps in collaboration
* ***It has additional workflows for reviewing and accepting new code***

**Day 20: June 25, 2021**

**Presentations:**

**Team 2:**

* Good Introduction to the project
* During presentation and explanation of navigation, could have shown the website along with code.
* Name "clickDropdown" could have been "SelectCity"
* Explicit Wait instead of Thread.sleep() when clicking on plus button

**Team 1:**

* Good Start to the presentation
* Heard Divya clearly for the first time :-) Well explained
* Thread.sleep() on line# 129
* Very clean code
* Assertions created very well
* Object identification (xpath) is good and meaningful

**JSON Format:**

**{**

**Key1 : "value1",**

**Key2: "value2",**

**Key3: {**

**Subkey1: "subvalue1",**

**Subkey2: "subvalue2"**

**}**

**}**

db.collection.find({key3.subkey1 : "subvalue1"})

**{**

**Key1 : "value1",**

**Key2: "value2",**

**Key3: {**

**Subkey1: "subvalue1",**

**Subkey2: "subvalue2"**

**},**

**Key4: "[ {**

**Arraykey1: "Arrayvalue1",**

**Arraykey2: "Arrayvalue2"**

**},**

**{**

**Arraykey3: "Arrayvalue3"**

**} ]"**

**}**

**Key4[0].Arraykey1**

**Test Automation:**

Selenium is open source

Selenium is the leading tool

Used widely in the Industry

W3C Standard

**Resume Update pointers:**

Trainings and Workshops:

1. Hands on workshop with real projects on Selenium
   1. Used multiple locator strategies
   2. Used XPath from scratch (relative Xpath)
   3. Used Implicit and Explicit wait for synchronization
   4. Used Keyboard Mouse Actions API in Selenium for performing Mouse hover, drag & drop, scroll, etc
   5. Other features like popup handling, Select API, etc also used in real time
2. Maven & TestNG
   1. Upgraded the Selenium projects by using Maven as a build automation tool.
   2. Created and updated POM.xml to configure project properties and download dependencies.
   3. Used TestNG to organize and execute the Selenium scripts.
   4. Used TestNG to create Suites and generate reports for the Test Cases
3. MongoDB
   1. Used MongoDB to create databases
   2. Used MongoDB databases to create Test Data and read it in the Selenium script at run time for externalizing the data
4. Jmeter
   1. Created Performance tests using Jmeter
   2. Used Think time and Thread configurations to do Load testing
   3. Used various assertions in Jmeter like response, xpath, duration, size, etc
   4. Parameterization and Correlation in Jmeter

When developers can do automation, why do companies need Test Engineers?

* Big Picture Thinking
* Mindset
* **Confirmation Bias**
* Developers can miss some important scenarios
  + They have assumptions
* Testers not only want to confirm that the requirements are met, but also want to test whether the system can withhold stress
  + Primary Job: Testers try to break the system
  + Secondary Job: Confirm whether the system meets criteria
* Testers test on various platforms

Before testing

After testing

What should a tester do day to day:

1. Information radiation
   1. Tell all stakeholders the REAL feedback about the quality of the system
   2. Highlight issues
   3. Document issues and information
   4. Share Observations
   5. Create test Cases
   6. Create Automation

**Layers in Project Development - A Tester has to test at each of these layers / levels**

Client Side Development - UI Development - UI Developers -> UI Design and HTML Code writing (CSS, HTML, Javascript)

Server Side Development - Business Functionality Development -> Developers -> Java, Python, C#, etc

Database Development

A Tester has BIG Picture of all the different layers