Stage 1 - Core Java & advance java

Stage 2 - Selenium (4 hours (web automation) & AutoIT (windows automation)

Stage 3 - BDD framework

Stage 4 - Git and Jenkins (CI/CD pipeline)

Github

Java - <https://github.com/balaji-githubstore/JavaConceptHealthAsyst.git>

Selenium - <https://github.com/balaji-githubstore/SeleniumConceptHealthAsyst.git>

1. Java

* Language developer by Sun microsystems
* High level programming
* Platform independent
  + Write once run anywhere

Java Architecture

Source code (.java) 🡪 Byte code (.class) 🡪 O/P

1. JVM - Java Virtual Machine

* Source code - can be written in simple text file (.java)
* Java compiler - converting source code (.java) to byte code (.class)
* Byte code interpreted by JVM
* JVM varies from platform to platform or JVM is platform dependent

1. Installation
   1. JDK 8 (Java Development Kit)
      * By default JRE (Java Runtime Environment) will be installed

<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>

* 1. IDE - Eclipse/Intellij/netbean

Option 1-

* Download eclipse.exe
* Choose - Eclipse IDE for Java developer
* Install it

Option 2-

* Download installed package as zip folder.

<https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2021-12/R/eclipse-java-2021-12-R-win32-x86_64.zip>

1. Compile time - Source code to byte code

Run time - Bytecode to O/P

1. UpperCamelCase - MyFirstProject

lowerCamelCase - myFirstProject

1. Eclipse Structure

Workspace - lowercase

Project 1 - UpperCamelCase

Package 1 - lowercase (com.healthasyst.purpose) or (org. healthasyst.purpose)

Class 1 - UpperCamelCase

Methods/variables - lowerCamelCase

Class 2

Methods/variables

Package 2

Project 2

1. Datatypes
   1. Primitive datatype/pre-defined datatype
      * Byte - 8 bits

1 bit - sign (+ or -)

7 bit - storing the number

* 1. Non-Primitive datatypes/ Non - predefined datatypes - collection of primitive datatypes
* String
* Array
  + Size should be defined
  + Zero based index

1. Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 6

at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)

at java.base/java.lang.String.charAt(String.java:712)

at com.healthasyst.datatypes.DatatypesDemo.main(DatatypesDemo.java:44)

1. Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5

at com.healthasyst.datatypes.ArrayDemo.main(ArrayDemo.java:14)

1. Debugging

* Resume
* Terminate
* Step into
* Step over

1. Conditional statement
   1. If condition
   2. Switch condition
2. Relational operators

==, !=, >,<,>=,<=

1. Logical operator

And (&&), Or (||), not

1. Iterative statements
   1. For loop
      * Initial condition
      * End condition
      * Iteration
   2. For each loop - designed for handling the array and collections
   3. While
   4. Do while - minimum once it will execute
2. break - to stop the further iteration

Continue - just moves to next iteration

1. Pre/post increment
2. **Methods - It’s a building block of your program**

* Reusability
* Maintenance
  1. **Create and call static method**

//accessmodifier static returntype methodname(arguments)

**public** **static** **double** areaOfCircle(**int** r)

{

**double** area = 3.14 \* r \* r;

**return** area;

}

**To call the static method** - **classname.methodname(args)**

* 1. **Create and call non-static method**

//accessmodifier returntype methodname(arguments)

**public** **double** areaOfTriangle(**double** base, **double** height) {

**return** (base \* height) / 2;

}

**To call the non-static method** -

* **Create object**
* **Use *objref.methodname()***

1. **Object** 
   1. **Declaration -** Area obj
   2. **Instantiation - new**
   3. **Initialize -** Area()
2. **Variable** 
   1. **Create and call static variable**
   2. **Create and call non-static variable**

1. Static vs Non-static with variable
2. Class & Object
3. Access modifier
   1. Private - accessible within the class
   2. Default - accessible within the package
   3. Protected - - accessible within the package and also in the inherited class
   4. Public - anywhere
4. SOLID Principle

S-Single Responsibility Principle - A class should have one and only reason to change, it should have only one job.

1. Exception in thread "main" java.lang.NullPointerException: Cannot read field "aNS" because "obj2" is null

at com.healthasyst.variable.VariableDemo.main(VariableDemo.java:32)

1. **this** keyword can be used under any method to refer the current object.
2. Stack and heap
3. String - immutable
4. Git - Git is a free and open source **distributed version control system.**

Download and install - <https://git-scm.com/>

**Architecture**

Project (local system) 🡪 local repository (local system) 🡪 remote repository (github, AWS code commit, Bit Bucket)

Steps

Local repo

1. git init 🡪 initialize the local git repo
2. git add README.md 🡪 collect the files,folder ready to move
3. git commit -m "first commit" 🡪 update the local repo

Server repo/remote repo

1. git remote add origin <https://github.com/balaji-githubstore/JavaConceptHealthAsyst.git> 🡪 register the remote url with the name origin
2. git push -u origin main
3. Constructor

* Pre-requisite for the object
* Constructor name and class name should be same. There is no return type.
* There will be default constructor which helps to load all the non-static (instance) variable with default value.
* We can override the default constructor by creating the constructor explicitly
  + Constructor without arguments
  + Constructor with arguments
* Whenever constructor created then we need to call that constructor on creating the object.

1. Static Polymorphism/ Compile time polymorphism/ early binding - Method to be called is resolved during compile time.
2. Constructor Overloading

By change in

* 1. Number of parameters/arguments
  2. Sequence of parameters
  3. Datatypes of parameters

1. Method Overloading

Multiple method with same name

By change in

* 1. Number of parameters/arguments
  2. Datatypes of parameters
  3. Sequence of parameters

1. Inheritance - reusability
   1. When “is-a” relationship existing between two classed, we use inheritance
   2. If a parent class is having constructor with parameter then you need to call it using super keyword.
2. Understanding more about String
   1. String - immutable
   2. String pool
   3. String builder
3. == and String method (.equals)
   1. == 🡪 compares the location and text
   2. .equals 🡪 compares only the text
4. Dynamic Polymorphism/ Method Overriding/ Run time polymorphism/ Late binding

Method to be called is resolved during run time.

* You need create the method in parent and override the same method in child with different definition. Signature, parameter should be same as parent.
* Create an object for child and store it in parent reference.
* Based on the object created, if the method is overridden then that method will be called during runtime.

1. Abstraction
   1. Abstract class
      * Can have method without definition
      * When the method is abstract then the class should be abstract
      * Whichever the class inherit the abstract class then the child class should give the definition
      * Cannot create an object for abstract class (cannot instantiate)
      * When you have method with definition (non-abstract methods )and without definition then you can go with abstract.(abstract method)
   2. Interface - to establish the set of rules/define standard

Few rules

* + - Method declared inside interface is **public and abstract method**

1. All lower level module will be depends on higher level module (like abstract class or interface)
2. Exception in thread "main" java.lang.ClassCastException: class com.healthasyst.oops.ChromeDriver cannot be cast to class com.healthasyst.oops.Employee (com.healthasyst.oops.ChromeDriver and com.healthasyst.oops.Employee are in unnamed module of loader 'app')

at com.healthasyst.oops.BrowserTest.main(BrowserTest.java:101)

1. Type Casting
2. Collections
   1. Non-Generic type
   2. Generic type (recommended)

Selenium - A suite of tools

1. Selenium IDE
   1. No need of programming knowledge
   2. Record and playback feature
   3. Plugin - chrome, firefox, edge
   4. Used for Simple scripting
2. Selenium RC (Remote Control)
   1. Programming knowledge is must
   2. Java, C#, python, ruby, javascript
   3. Architecture

Source code (java+selenium RC) 🡪 RC server (turn on/off) 🡪 Browser

1. Selenium WebDriver
   1. Programming knowledge is must
   2. Java, C#, python, ruby, javascript
   3. Architecture

Source code (java+selenium webdriver) 🡪 Browser

1. Selenium Grid
   * + Hub and Node

Selenium WebDriver

1. Create Project
2. Configure the selenium jar
3. Configure the driver based on the browser and browser version.
4. Exception in thread "main" java.lang.IllegalStateException: The path to the driver executable The path to the driver executable must be set by the webdriver.chrome.driver system property; for more information, see https://github.com/SeleniumHQ/selenium/wiki/ChromeDriver. The latest version can be downloaded from <https://chromedriver.storage.googleapis.com/index.html>

All drivers can be found at

<https://www.selenium.dev/downloads/>

To fix:

1. Download the driver and keep it in project home directory
2. Download the driver and use System.SetProperty to driver directory
3. Download and add the driver path to environment path variable
4. Configure using WebDriverManager - which will check the current browser, browser version, platform and then auto download the driver.
5. Selenium webdriver doc - <https://www.javadoc.io/doc/org.seleniumhq.selenium/selenium-api/latest/org/openqa/selenium/WebDriver.html>
6. Inspect - tagname, attribute, text or not
7. Click, sendkeys, select
8. Basic Locators
   1. Id
   2. Name
   3. Classname
   4. Tagname
   5. Linktext
   6. Partial linktext

When there is duplicate locator then findelement will pick the first one.

1. Advance locator
   1. Xpath
   2. CSS
2. To get By object 🡪

By.*id*("authUser")

1. To get WebElement object 🡪

WebElement ele= driver.findElement(By.*id*("authUser"));

ele🡪L1-> (

<input type="text" class="form-control" id="authUser" name="authUser" placeholder="Username:">

)

1. Page load 🡪 // wait for the page load to complete
2. findElement 🡪 check for the presence of element in 0.5s
3. Synchronization
   1. Unconditional wait

Thread.Sleep(5000) 🡪 wait for 5s

* 1. Conditional wait
     1. Implicit wait
        1. Default implicit wait - 0s
        2. Applicable for all findelement and findelements method.
        3. Example: Implicit wait = 30s
           1. If element is not present, it will check for 30s then it will throw error
           2. If element is present then it will do operation immediately
           3. Polling time 🡪 0.5s (how frequently it checks)
     2. Explicit wait
     3. Fluent wait

1. Dropdown
   1. Dropdown with select tag
      1. selectByVisibleText(string)
      2. selectByValue(string)
      3. selectByIndex(string)
   2. Dropdown without select tag
      1. Use click operation to complete the task
2. Exception in thread "main" org.openqa.selenium.NoSuchElementException: no such element: Unable to locate element: {"method":"css selector","selector":"#authUser12"}
3. gettext, getattribute, gettagname
4. Frame, Multiple tabs/windows, alert - switchTo()
5. Frame
   1. Even though the locator is correct, we will get NoSuchElementException then we need to check for frame
   2. Check for tagname - frame, iframe
   3. To get into frame
      1. Using Index
      2. Using name or id as a String
      3. Using webelement

driver.switchTo().frame(driver.findElement(By.*xpath*("//frame[@name='login\_page']")));

1. List<String> vs Set<String>
   1. List can contains duplicates
   2. Set cannot contains duplicates
2. Close vs quit
   1. Close - close the current tab/current session
   2. Quit - close the browser and also kill the process associated with it.
3. Multiple tabs/windows
4. Exception in thread "main" org.openqa.selenium.NoSuchWindowException: no such window: target window already closed

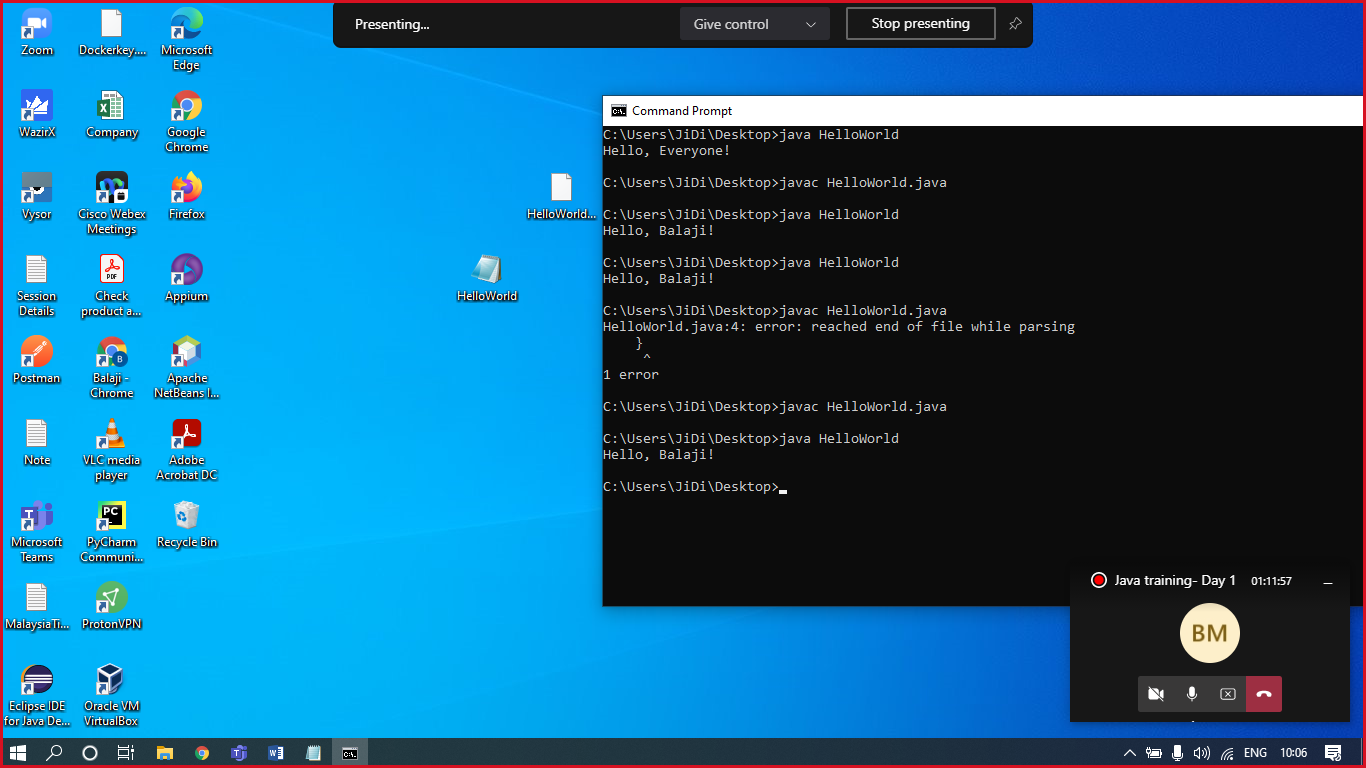
BDD Framework: -

Maven - Build Management Tool

* + - 1. Create Maven Project
         1. Group Id
         2. Artifact id = project name
      2. Configure the jars related to Selenium, Cucumber (BDD)

Reference:

Ref1: Img 1:



Ref2:

1GB - 1024 MB

1 MB - 1024 KB

1KB - 1024 B

1 B - 8 bits

1 bit (either 0 and 1)

Ref3:

<https://edabit.com/>

Task 1:

|  |  |
| --- | --- |
| **PERCENTAGE** | **GRADE** |
| 90 and above | A |
| 80 to 89 | B |
| 60 to 79 | C |
| below 60 | F |

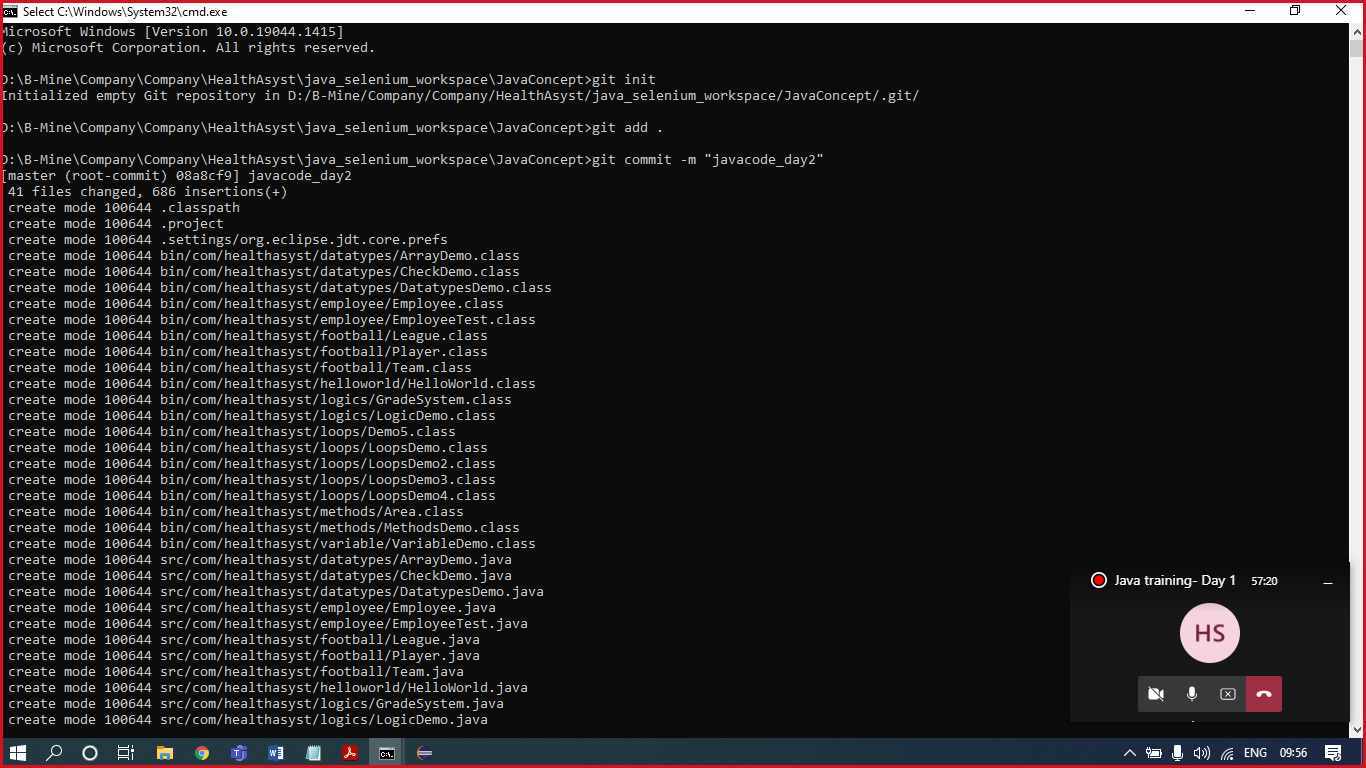
**int** i=1;

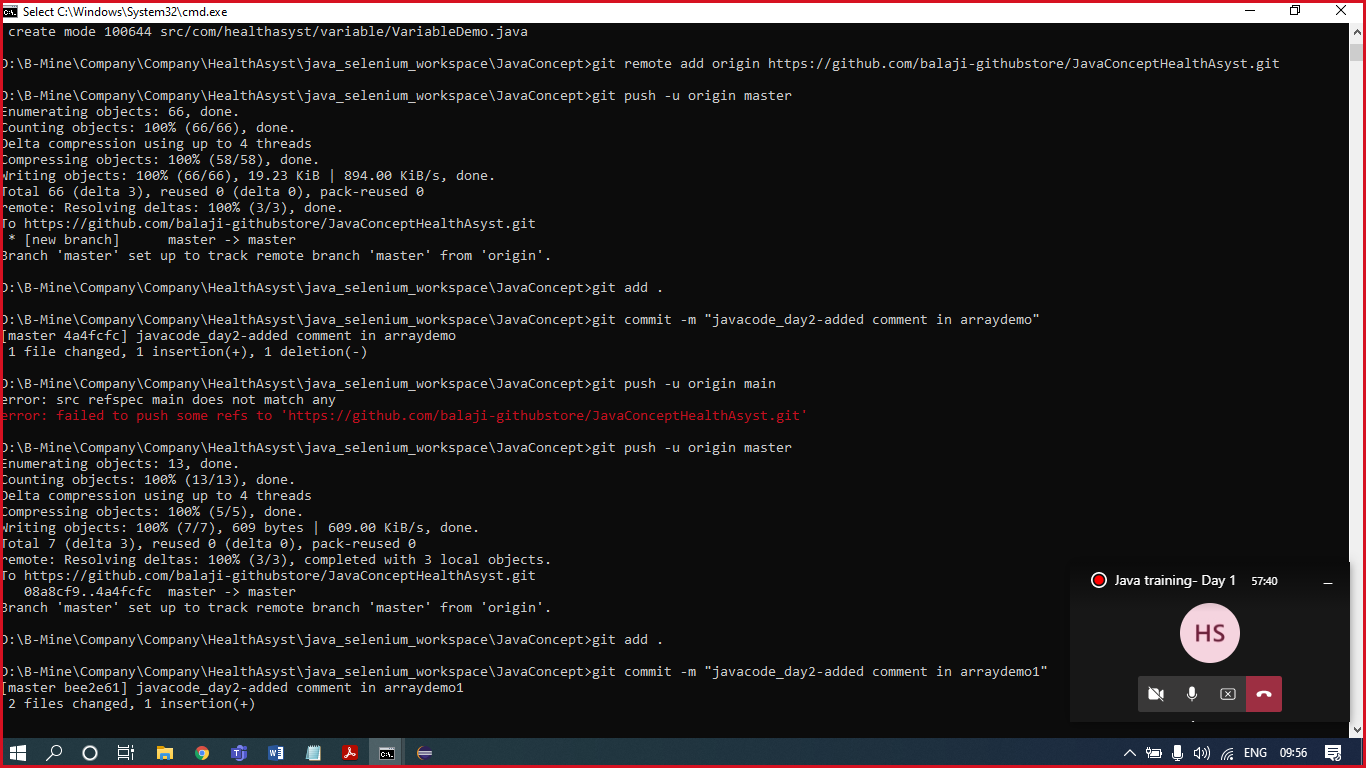
i++; // or i++ // i=i+1

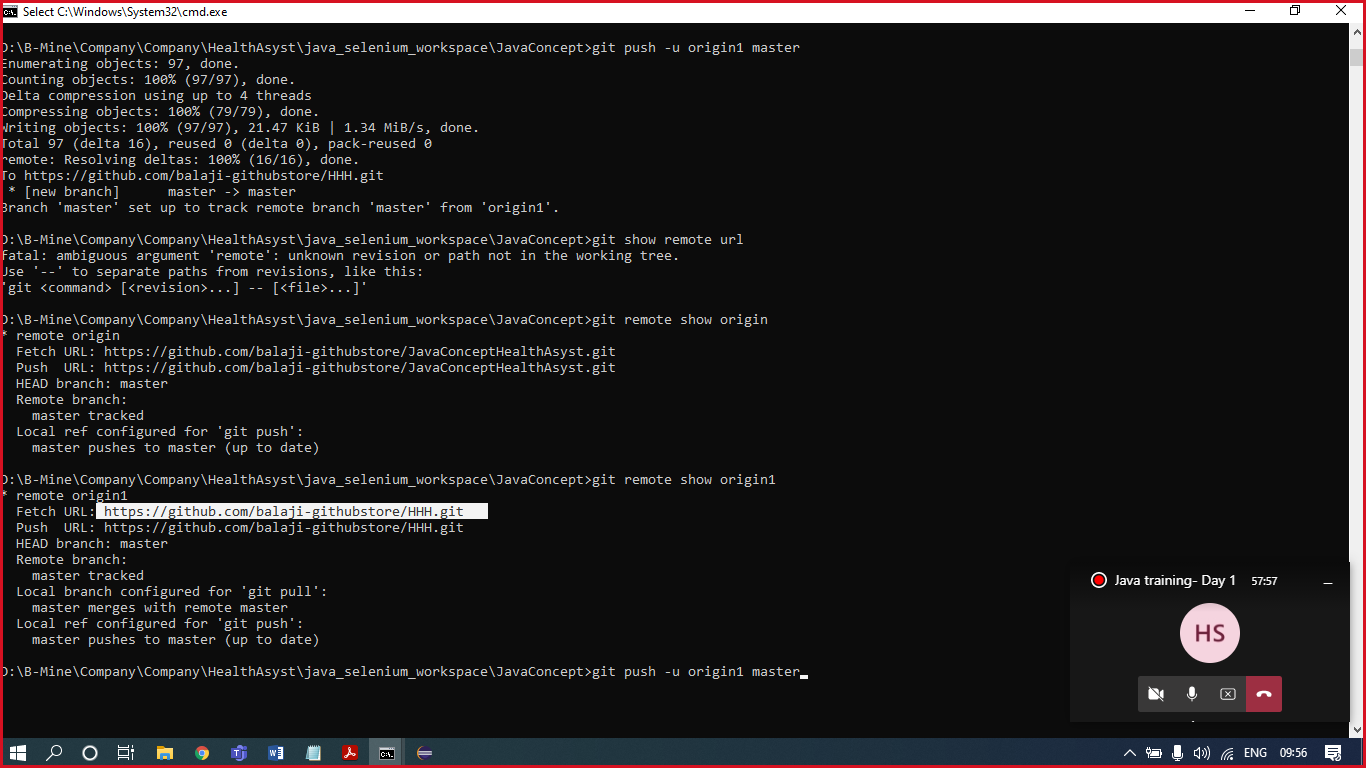
System.***out***.println(i);

Template for Student

Git:







<https://github.com/SeleniumHQ/selenium.git>