Agenda

1. Custom Exception
2. Interfaces, Abstract Classes
3. Getting input from User using Scanner.
4. Unit Testing (Junit)
5. Git branching & merging (Deep dive)

Day 7 Revisit

1. String is immutable Object. (It’s value can’t be changed after creation, when modifying a String it’ll create a new copy leaving the existing one as it is in memory.)
2. Strings will be saved in String pool. (It’s a special area in heap memory)
3. String is a special type in JAVA, where we can declare it as primitive as well as object
   1. String str = “Hello World”; //String literal
   2. String str1 = new String(“Hello World”); //String object
4. String is not a primitive type. String is a final class which stores the data in a array of characters.

String Buffer & String Builder

1. String Buffer & StringBuilder, both are Mutable Strings.
2. It contains append method, which will modify the original String instead of making a copy like in String.
3. StringBuffer is ThreadSafe where as StringBuilder is not ThreadSafe
4. StringBuffer will be little slow compared to StringBuilder
5. StringBuffer can be used in multi-threaded programs
6. While using StringBuilder in multi-threaded programs, we need to manually synchronize the methods

Access Modifers

1. Private (Least Accessible) – only with in the class by the public members.
2. Default/package. With in the class and with in the same package (No Keywords is used)
3. Protected (Within the class, within the package, sub classes in different package)
4. Public (Most Accessible) – Anywhere

Non-Access Modifiers (Behavior Modifiers)

1. Static [Once per class rather than once per instance] Used for Variable, methods & blocks. Accessed directly using the class name not the object references.
2. Abstract – Can be used for methods & classes only. Abstract tells the JVM that the method/class is in-complete. Abstract class can’t be instantiated directly.
3. Final – Used for variables, methods & Class. Final Variable is a constant (Value can’t be modified after initial assignment) Final methods can’t be overridden, final class can’t be extended [No derived classes for final class]
4. Volatile -
5. Transient – Preventing Serialization of sensitive properties like SSN, Credit Card Number etc.,

Arrays -Varargs – For-each loop (Enhanced For each)

* Arrays used to store same kind of data repeatedly.
* Useful to handle group of primitive or objects.
* It can be homogeneous (Same type of elements) – primitive Arrays
* It can be Heterogeneous (Different types of elements – Object Array)
* Arrays are fixed in Size.
* Single dimensional & Multi-dimensional array [int a[]=new int[5]; char studentsName[10][35]; //two dimensional array;
* Var-args = variable arguments. (… is used to represent variable arguments)
* Varargs can have zero or more arguments
* We can use varargs while sending a data to method
* Foreach loop is also called enhanced for loop.
* No need for condition checking, loop variable.
* Generally used for group of data like arrays, collections etc.,

WRT – With Respect To

Exception

1. It’s un-avoidable situation while running the code which leads to pre-mature closure of the program.
2. WRT the creators, it’s divided into Built-in/System/Pre-defined Exception and Custom/User-defined exceptions
3. WRT when it’s thrown, it’s divided into Compile Time Exception (Checked Exception) and RunTime Exception (Un-Checked Exception)
4. Two ways of Handling Exceptions
   1. Using try/catch & finally block (recommended approach)
   2. Using throws keyword [ informing the JVM to handle the exception]
5. Try block must have a catch block or finally block or both.
6. Try block with finally block alone is also valid but we are still not handling the exception and passing it to JVM only.
7. Try block can have multiple catch block or a single block which catches more than one exception.
8. The code in finally block will always get executed irrespective of the exception status.

Custom Exception is also called as User-defined exception.

Example – InvalidAgeException

* 1. Custom checked exception [extending Exception class]
  2. Custom un-checked exception [extending Runtime Exception class]

1. Method used in Exception classes
   1. printStackTrace() – It’s a highly resource consuming or costly method
   2. getMessage()

Note: If a process/method consumes more memory/resource or time, then it is called as costly process or costly method.

Getter & Setter is also called as accessor & mutator

Getter is used to get the data from the object.

Setter is used to modify the data of an object.

Both getter & setter are used after object creation. Where as constructor is used during object creation.

Constructor will initialize all the member properties with default values.

When using Eclipse or Intellij IDE use Ctrl+Space (This is the shortcut for Auto code completion & Code suggestion)

Useful Eclipse Keyboard Shortcuts (Windows)

|  |  |  |
| --- | --- | --- |
| Sl No | Short-cut | Purpose |
| 1 | Ctrl + Space | For Auto Code completion & Code Suggestion |
| 2 | Ctrl + F11 | To run the class (Starter class) – To Execute the Java Code |
| 3 | Ctrl +Shift + F | To Format the Code (Align the code properly) |
| 4 | Ctrl + Shift + O | To automatically manage the imports |

Custom Un-checked Exception

API – Application Programming Interface – Reusing already created classes & interfaces

Interfaces ( Contract ) – Specification

Vehicle – Generic Notation

Human – Generic (Man, Women, Boy, Girl, Employee, Customer, Supervisor, Doctor , Trainer, CarPenter, Painter)

TV, Washing Machine, Refrigerator, Electric Water Motor, Grinder, Mixie, Vacuum Cleaner, AC,

Power Outlet : Two pin/ three pin socket.

An interface acts as a contract for behaviours that a class can implement.

In Java, To differentiate class and interface, interface names usually ends with “able”

To defined that it’s capable of doing some actions

Example Interfaces

1. Cloneable
2. Serializable
3. Readable
4. Runnable
5. Comparable
6. Iterable

To define an interface “interface” keyword is used.

Interface will have Abstract methods – Methods with only declaration line no definition.

Normal/concrete/non-abstract method Example :

public void show() {

……

}

Abstract/in-complete/non-concrete method

public void show(); --- In interface can have this.

public abstract void show() --- in class we need to define abstract method in this way.

All the members of interface are public and abstract (Even though we are not specifying the access modifier)

* We don’t have Constructor for interfaces
* We can’t create object of interfaces directly (Indirectly – using anonymous inner class it’s possible)
* Variables are 'public', 'static', and 'final' To inherit interfaces, a class must *implement* them and they are REQUIRED to implement all methods, unless the class is abstract.
* Methods are public & abstract

Anonymous – nameless (Without a name)

Abstract class – In-complete or Non-Concrete class

Abstract Class – A class with zero or more abstract method.

Car or Phone

* Car is a Transport medium (Used to transport human from one place to another place)
* Phone is used to establish communication between two users

Car – Electric Car, Autonomous Car (Self-driving), Hybrid Cars, Super Cars

Audi Car

Honda Car

BWM Car

Car is a specification where as Audi Car is the implementation, Honda Car is the implementation

Car is a Interface , Audi, Honda, BWM they are all classes.

Interface vs Abstract Class

* Interface can’t have non-abstract methods
* Interface can have default & static non-abstract methods (since java 8)
* All the members of interface are by default public
* In abstract class, need to specify the access modifier for each property & behaviour
* In general, Classes won’t support multiple inheritance in JAVA (A class can’t extend two other classes) but interface can able to do it.
* Interfaces can have only static and final variables

<https://www.javatpoint.com/difference-between-abstract-class-and-interface>

Getting User Input using Scanner

Streams – represent flow of data

Input Stream – System.in (Keyboard)

Output Stream – System.out (Console – Monitor)

Error Stream – System.err (Console – Monitor)

Printf – C Style syntax

%d – Decimal/integer

%f – float

%s – string

https://www.geeksforgeeks.org/format-specifiers-in-c/#:~:text=The%20format%20specifier%20is%20used,d%2C%20%25f%2C%20etc.

Recap

1. Custom Exception (Creating checked & un-checked custom exceptions)
2. Interface & Abstract Class
3. Getting User input using Scanner class

What is the difference between final, finally, finalize

Final is non-access modifier (used to create constants, classes that can’t be extended, methods that can’t be overridden)

Finally is a block in exception-handling. The code inside it will always get executed and is the suitable place to release memory spaces, perform cleaning activity, closing resources etc.,

Finalize is a method which will be called automatically by the JVM before garbage collecting the object.

TDD – Test Driven Development (Writing Unit Test, Test cases before starting the application)

1. Write the Unit Test code with out the application code
2. So test will fail
3. Write the application code to make the test case pass
4. Continue it.

BDD – Behaviour Driven Development

Unit Test – Testing individual software Component

When writing a Class, A test case class also will be written. Each class will have a separate test class which is called unit testing class.

Earlier days Manual Testing is carried out using Excel sheets

Sample Excel Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sl No | Req ID | Pre Condition | Assumption | Expected Result | Actual Result | Test Case Status |
| 1 | 100 | Calculator Class is running | Calling Add method with arguments 5,8 | 13 | 13 | Pass |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Junit – Java Unit Testing Framework.

Test cases can be automatically produced by the Framework. When we add a new class automatically the unit test code also will be generated by the framework.

<https://junit.org/junit5/> -- Official Site of Junit

Latest version of Junit is Junit5.

It automatically creates test cases for each class.

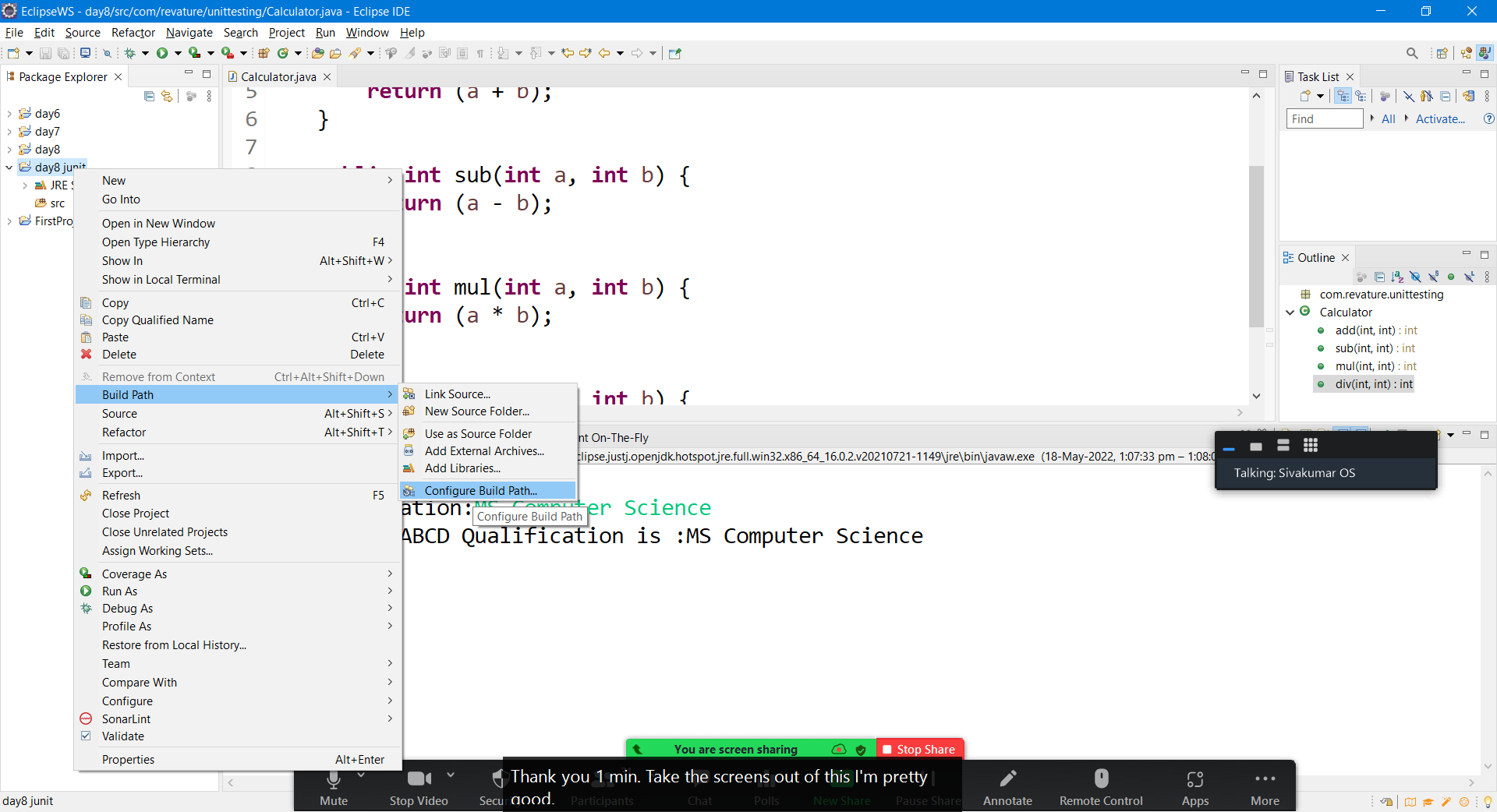
Junit4 Download Link -- <https://github.com/junit-team/junit4/wiki/Download-and-Install>

Maven URL to download Junit4 -- <https://search.maven.org/search?q=g:junit%20AND%20a:junit>

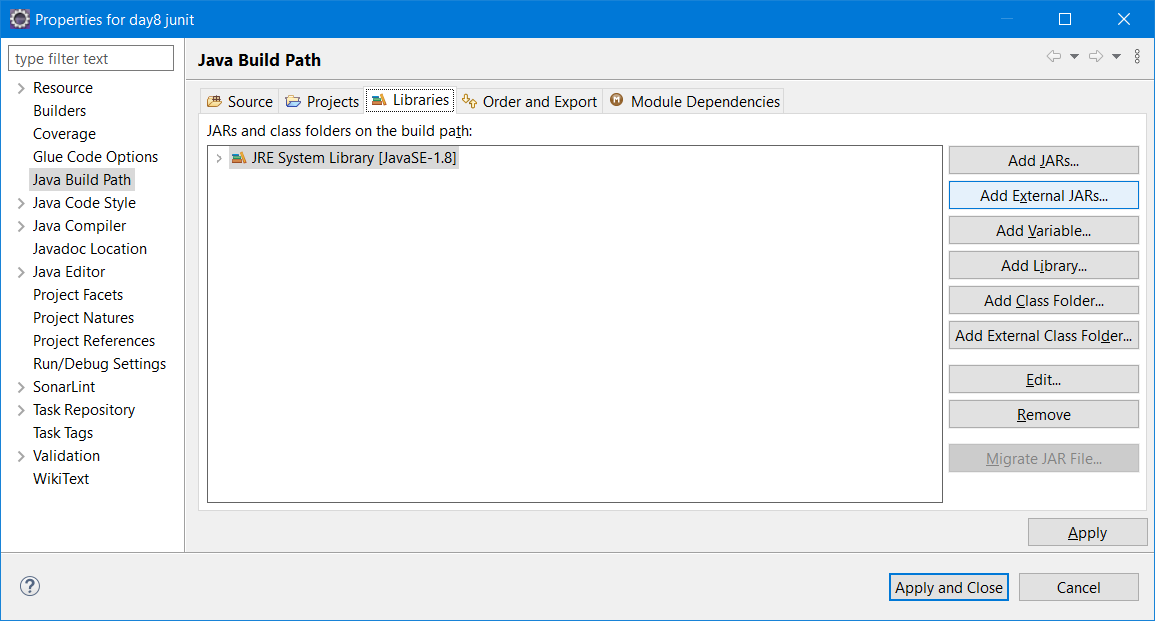
Harmcrest-core1.3.jar download link

Adding Junit & Harmcrest-core jar file to the project build-path

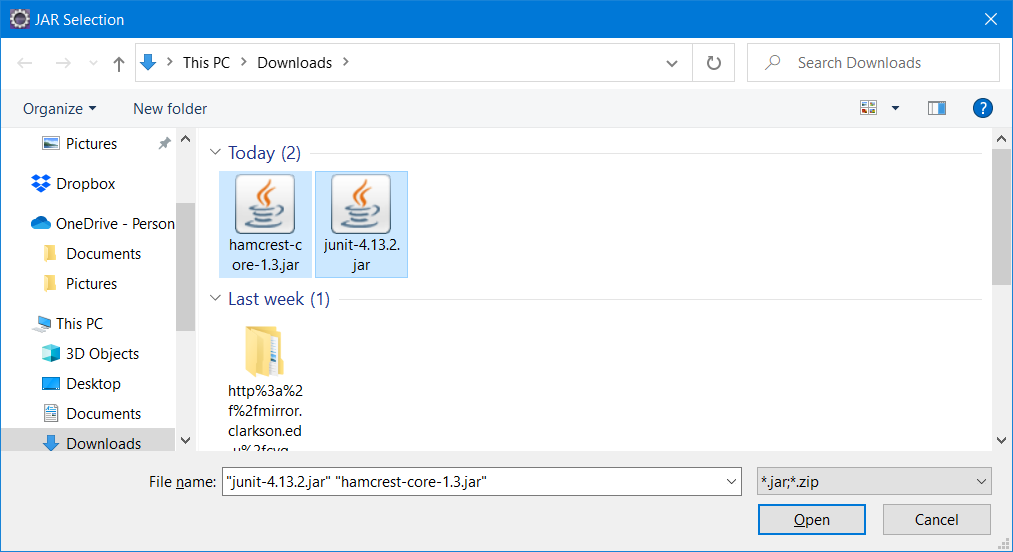
Right click the project, select “Build Path” -> “Configure Build Path” option as shown below.



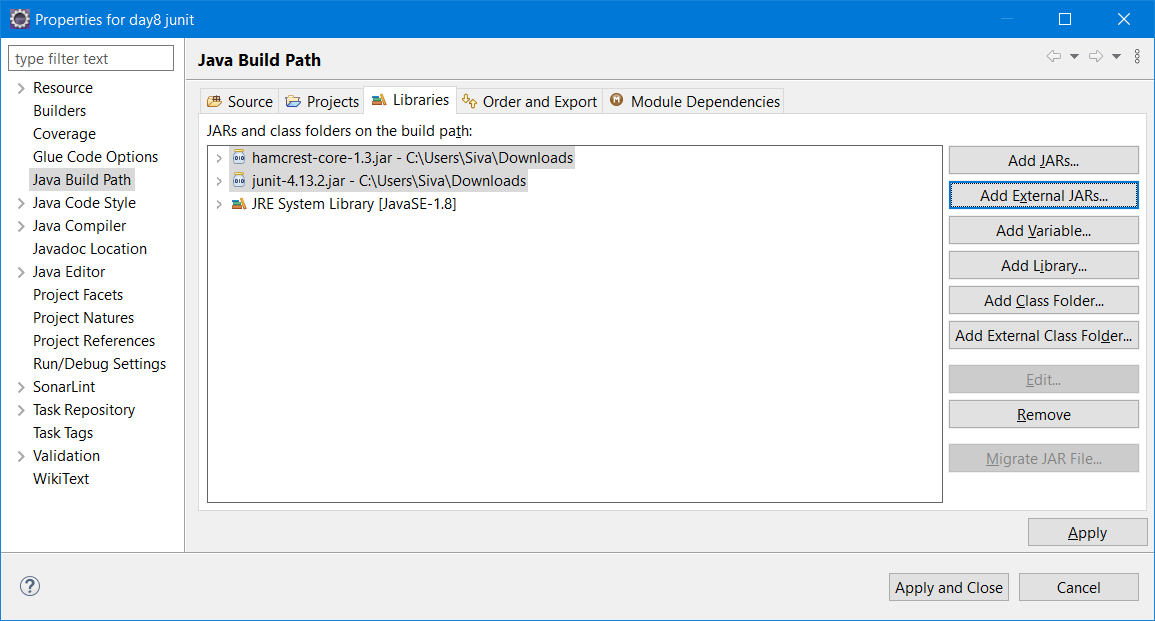
Click on “Add External jars” button in libraries tab as shown below



Select the recently download Junit4.jar & harmcrest-core1.3.jar file by pressing ctrl key and click on “Open” button to add them to build-path



Then click on “Apply and Close” button as shown below



Annotations used in Junit

@Test – Will be added to each test case method

@BeforeClass

@AfterClass

@Before

@After

beforeClass & afterClass will be executed only once i.e before starting any test case and after completing all the test cases respectively

But before and after will be executed for each test cases.

Assert Methods

1. assertEquals()
2. assertTrue()
3. assertFalse()
4. assertNull()
5. assertNotNull()

<https://www.javatpoint.com/junit-tutorial>

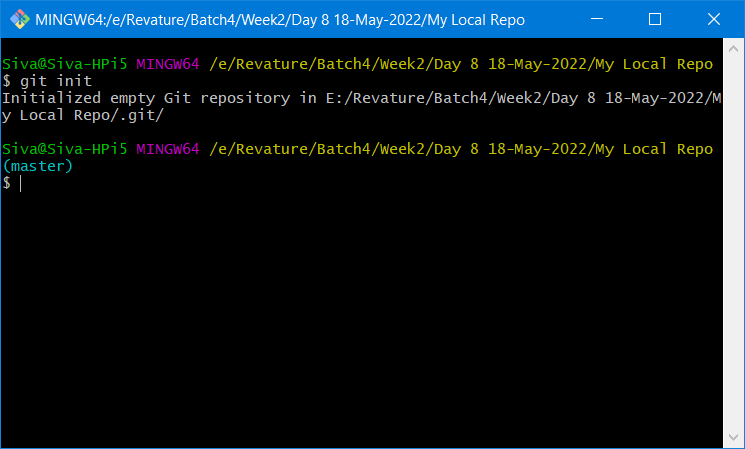
Git – Is a open source distributed version control system developed by linus Torvalds.

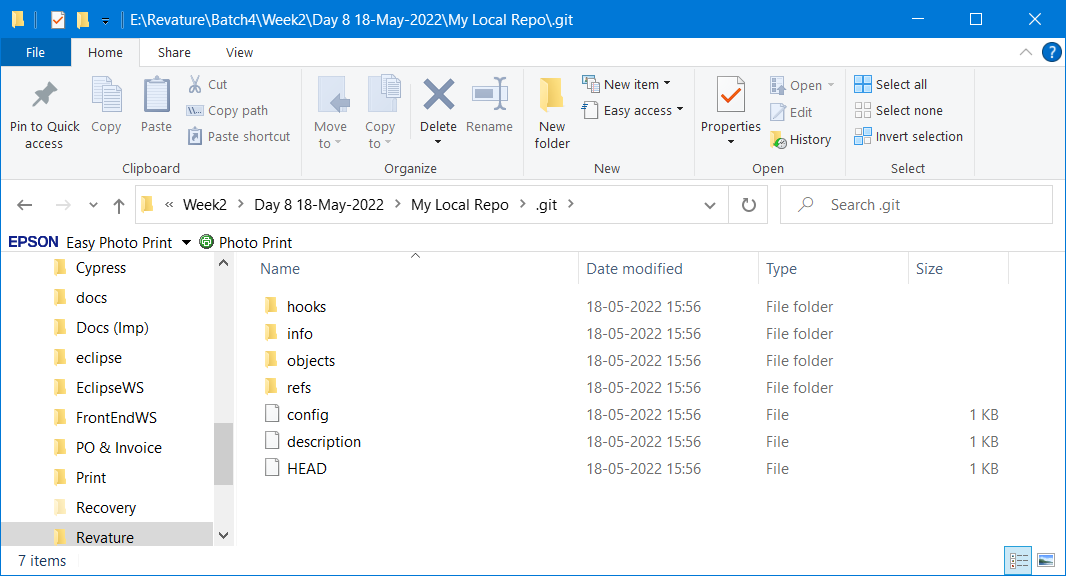
Git commands

1. git init -- To initialize empty git repository
2. git clone <git.url> -- To copy remote repo to the local repo
3. git add <file\_name> -- To add a file for tracking [To staging area]
4. git commit -m “commit message” – To commit the file [ making changes permanent and moving to repo area from staging area]
5. git pull – to download latest copy of remote repo to local repo
6. git push – to sync local repo with remote repo.
7. git log – to get commit info of a repo
8. git status – to display the local repo status
9. git branch <branch\_name> -- to move/create a branch

REPO – Repository (It’s a folder where all the source code will be saved & tracked)

* git init

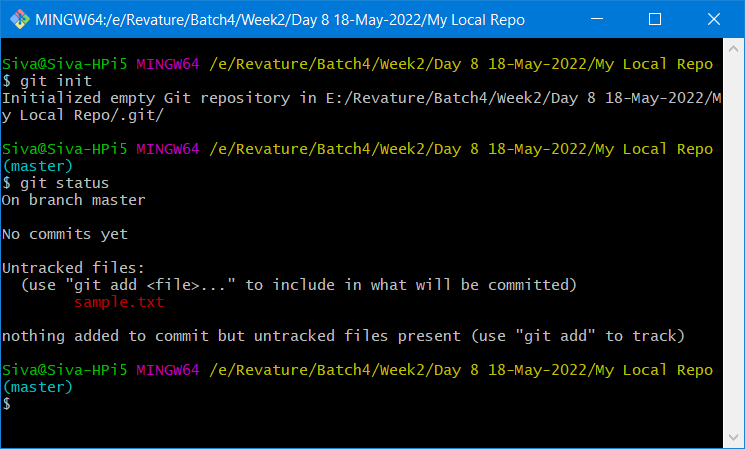




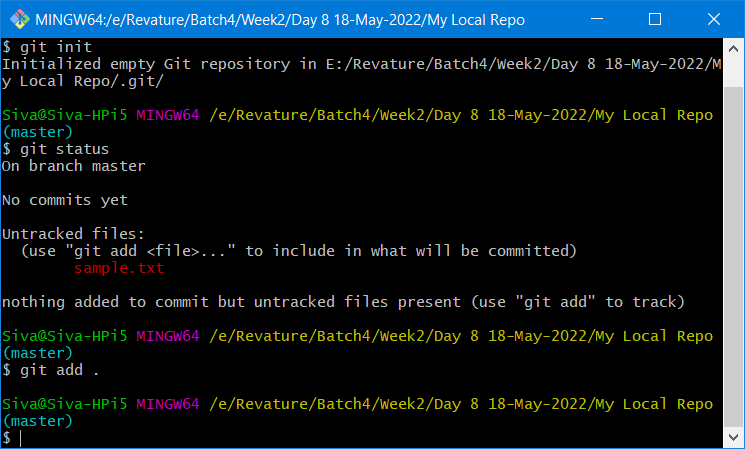
Working area (Local folder) – git will not track the local folder contents until it’s added to the repo

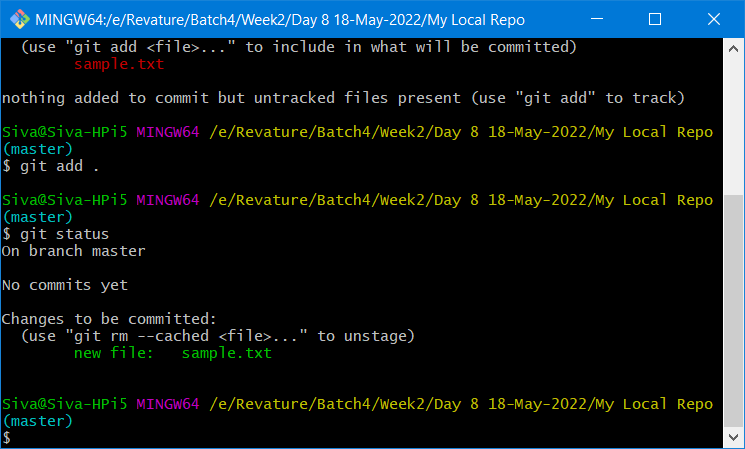
Staging area (Temp location) – git will neither track nor ignore the files/folders available in this area.

Repo area(after commit) – git will start tracking the contents in the repo area



git add . – To add all the files & folders present in that folder currently

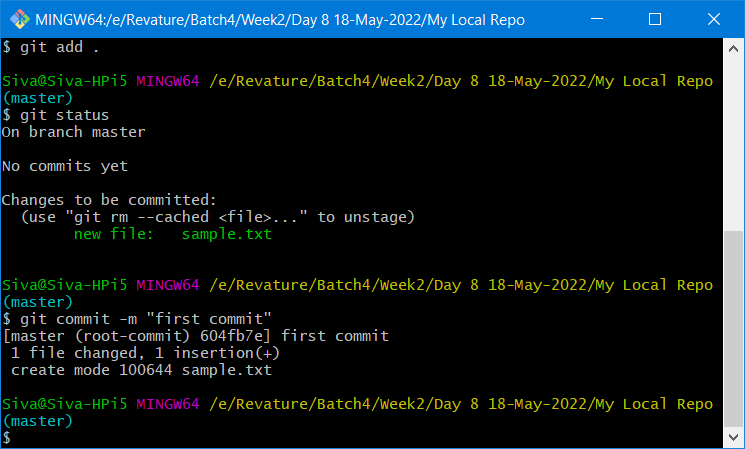




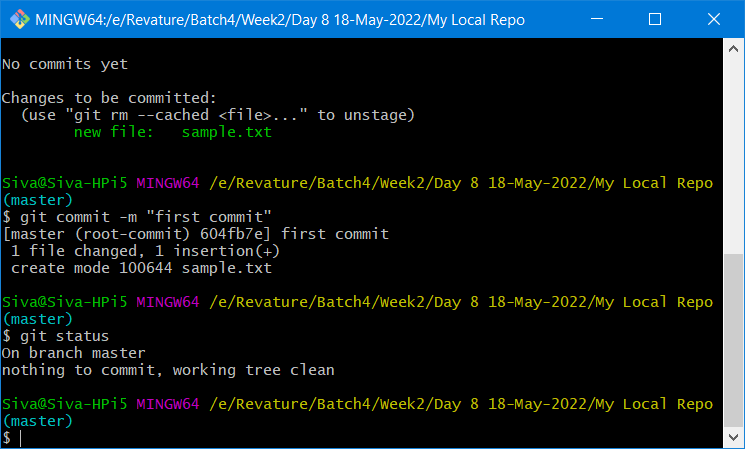
Unstaging – moving from staging area to the current folder back. (To avoid tracking)

git -rm --cached <file\_name>

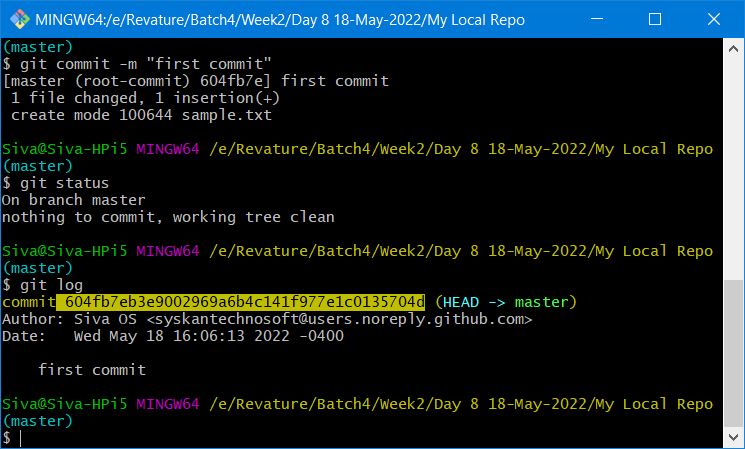
git commit -m “first commit”

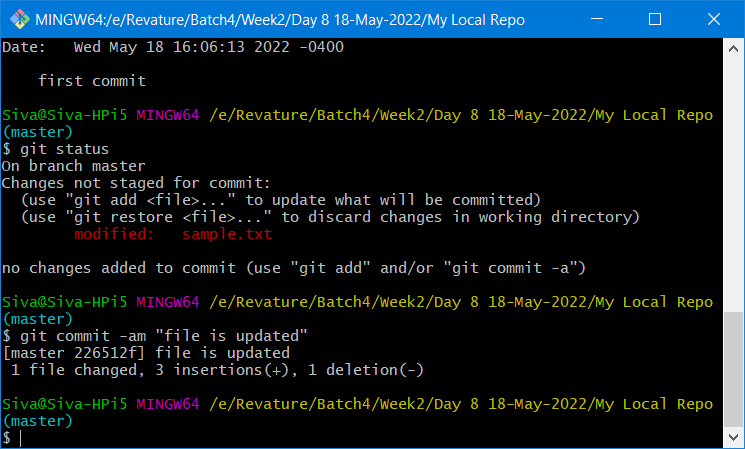


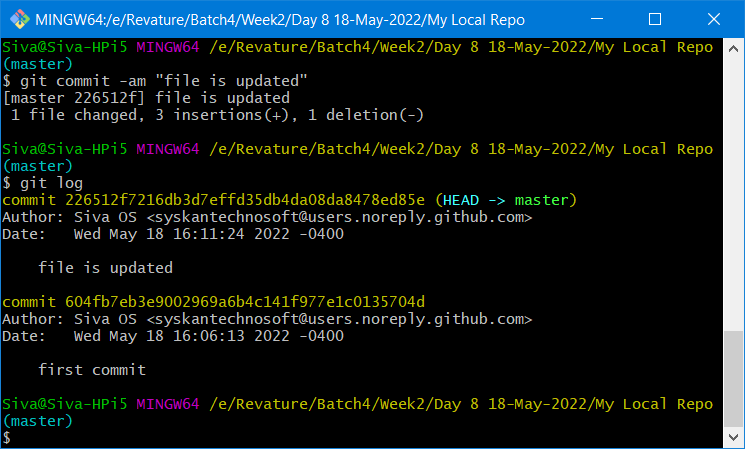
git status



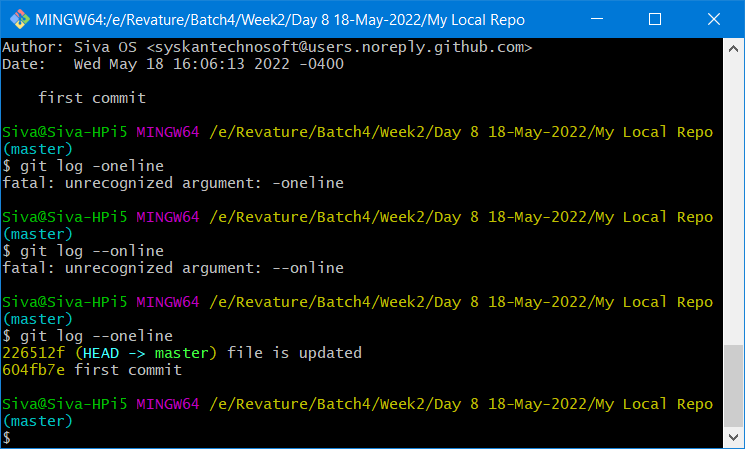
git log --- provides all the commit details







git log – oneline

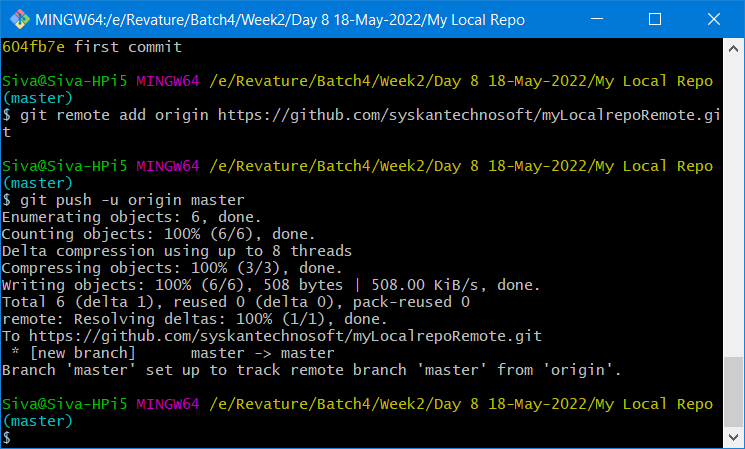


Local Repo (Resides in the local computer – A folder /directory in a local system)

Remote Repo (It resides in Web/cloud – Github/ gitlab/bit-bucket)

git remote add origin https://github.com/syskantechnosoft/myLocalrepoRemote.git

git push -u origin master



It generated the report by running all test cases.