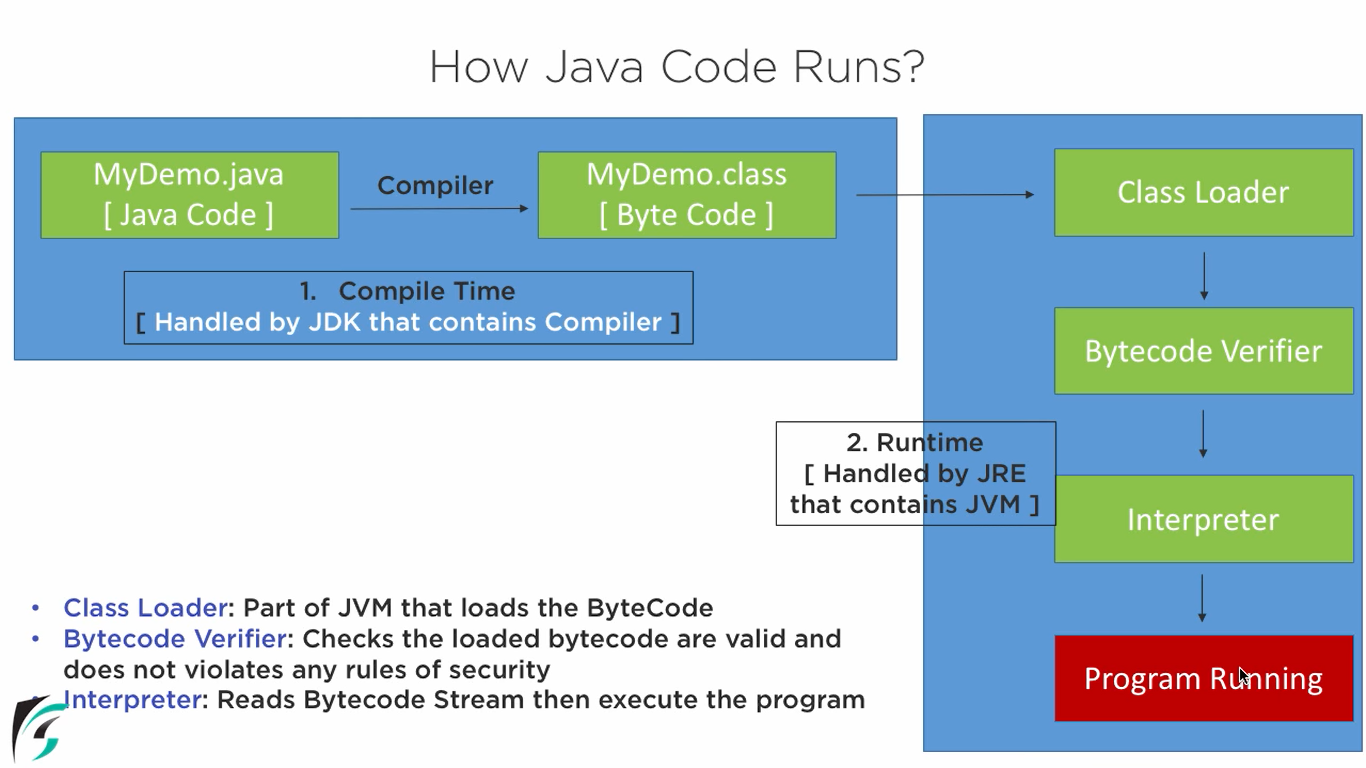
DAY1

Fundamentals of JAVA

* Java is a platform independent language because it uses Virtual Machines for run the code.



**JDK – Java Development Kit** helps to develop/compile a java program. Inside JDK we have JRE and JVM.

**JRE – Java Runtime Environment –** handles run the program which contains files and library.

**JVM- Java Virtual Machine** – do memory allocations in RAM for the created objects.

* **IDE – Integrated Development Environment** – support code completion, syntax, highlighting, etc.
* **IDE software** – Eclipse, Netbeans, Jdeveloper – oracle, RAD –IBM, Notepad.

**Installation:**

1. **JDK**

Search 🡪 jdk 1.8 download later

Check the oracle website 84341

Window x86 installer 🡪 32 bit

Windows x64 installer 🡪 64 bit

1. **Download Eclipse and install3.**

**Features of JAVA:**

* Open Source software (free license)
* Platform independent(OS, windows, linux, solaris,mac)
* Multiple Threading – run multiple programme at the same time.
* Secure – creates a virtual firewall between the computer and programme. So it doesn’t grant unauthorized access.

1. **OOPS**

- its method of implementation in which program is organized as collection of object , class, and methods.

Inheritance

Polymorphism

Abstraction

Encapsulation

**Structure & Key Words in JAVA**

Project {

Package {

Class {

Methods() {

Main methods() {

Objects creation;

Method call();

}}}}}

**Project name and Class name -🡪 Pascal notation**

Eg, ProjectName, ClassName

**Object name and Variable name 🡺 Camel notation**

Eg, myStud, myEmp

**Package Name**

Eg org.tcs , com.tata

**Object –** run time memory allocation

**Method –** set of actions to be performed

**Class –** collection of objects and methods

**Package –** collections of classes

**Project –** collections of packages.

DAY2

Access Modifiers

* We cannot declare class as “Private”
* Main method is always “Public” 🡪 because it should call other methods & libraries. So it must be in public.
* “print” 🡪 just print the line
* “prinltln” 🡪 print and move the cursor to next line.
* Private is a class level access.
* Public is a global level access.
* To call class/methods of another package. We should use the below syntax in-between package declaration and class declaration.
* Syntax : import packagename.classname;

Data Types:

Primitive:: cannot able to store sequence of the charecter

Integer – byte, short,int, long (whole numbers)

Float—float and doble(decimal)

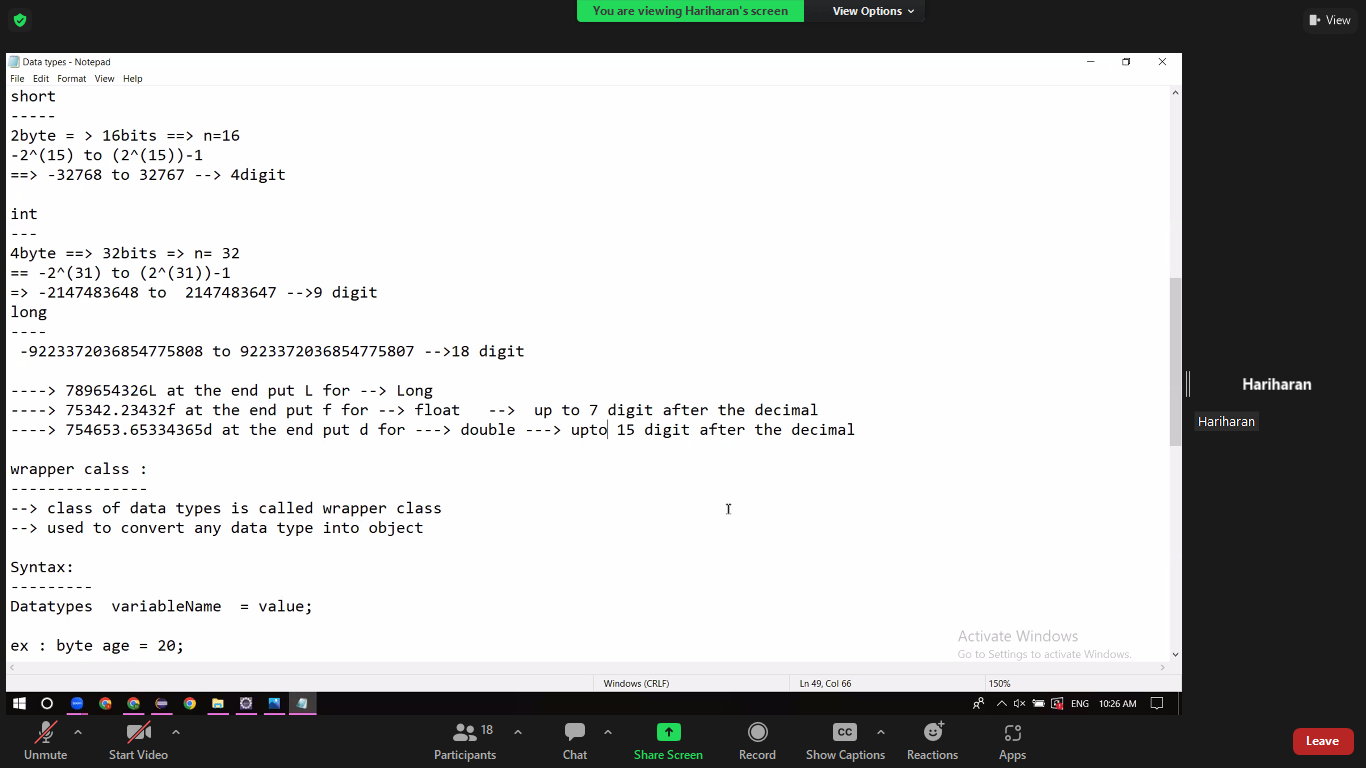
Char – to store sing character (^. #.d, 1)

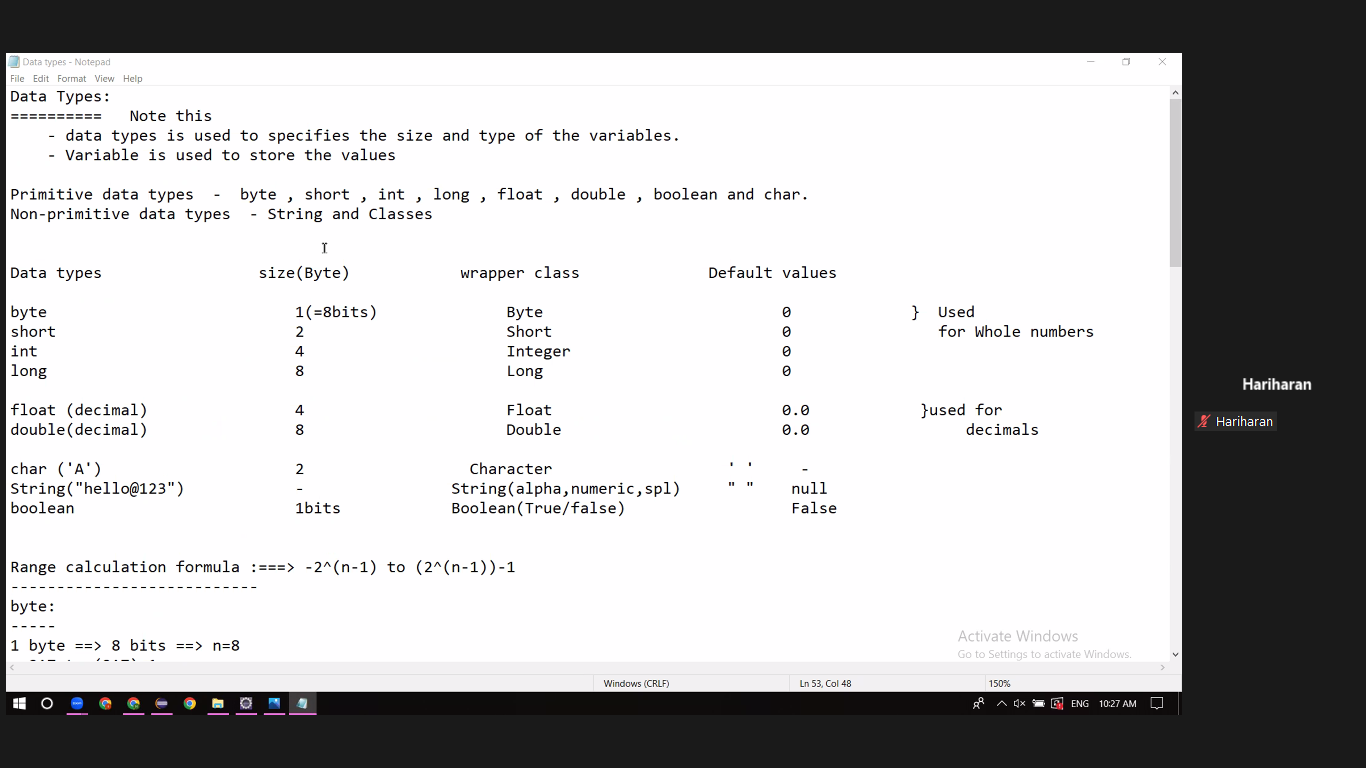
Boolean – stores true or false

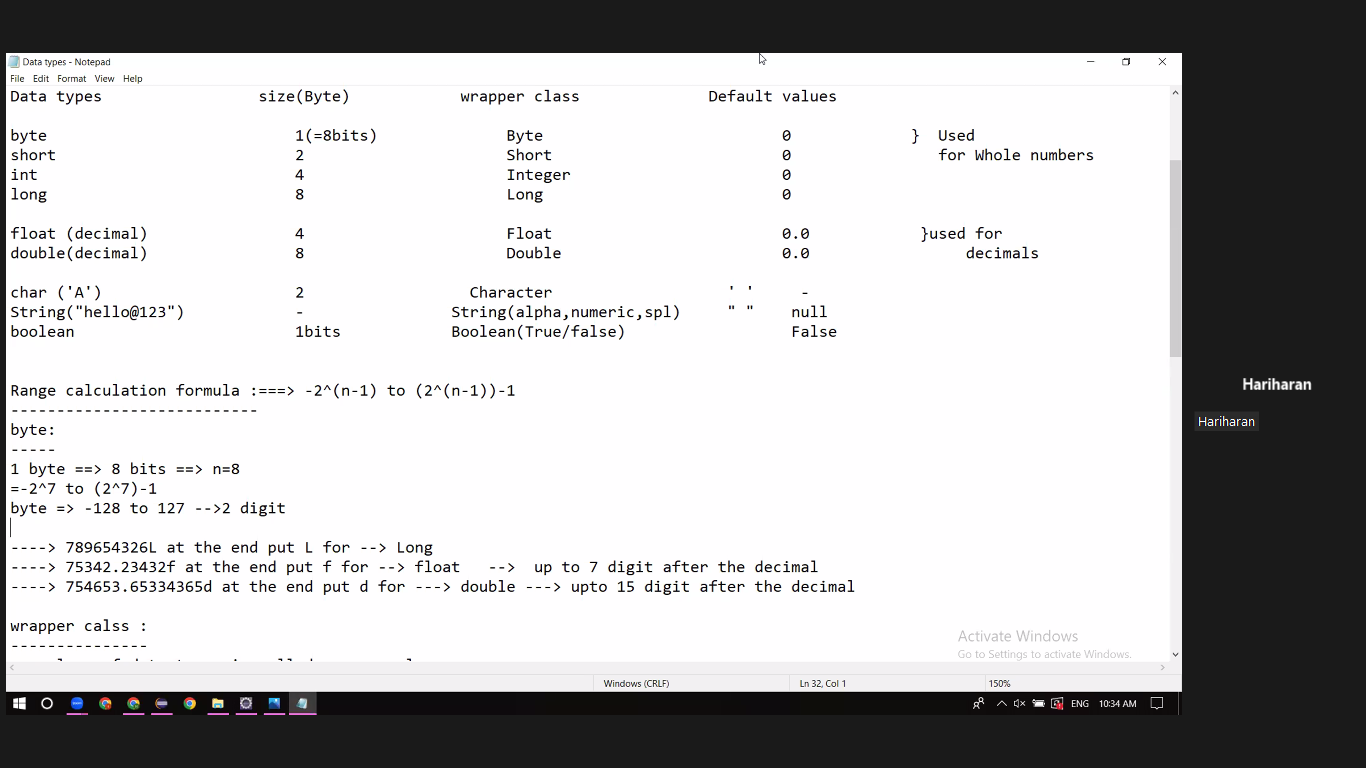
Non-Primitive::

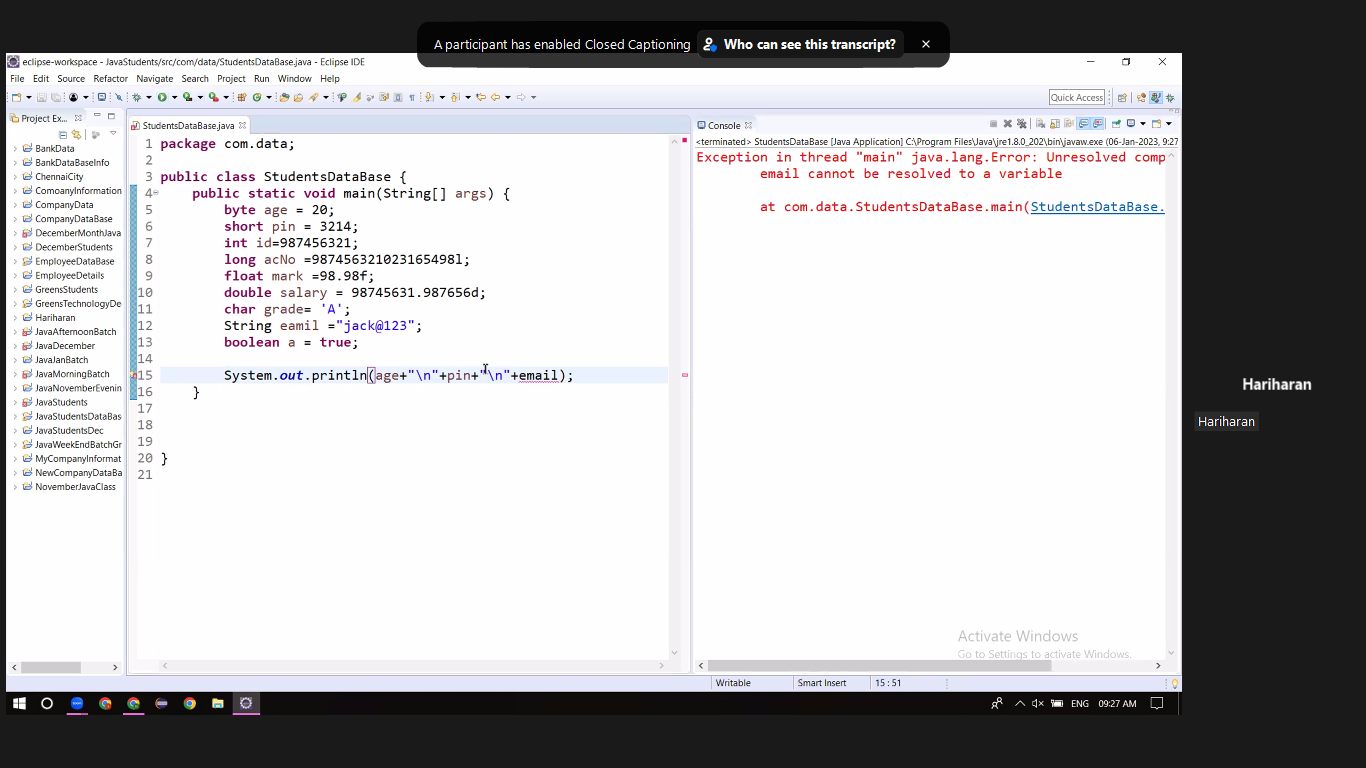
Can able to store sequence of character like char, number, special char, sequence of char











\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY3

**Inheritance:**

We can access one class property(method) from another class using extends keyword.

It uses **extends** keyword.

In inheritance we no need to create the object for parent class to use its properties. Here we need to mention

**Syntax:**

*Parent Class:*

Public class ParentClass{

}

*Child Class:*

Public class ChildClass extends Parent\_Class{

}

**Why?? Already we can use other class’s properties by creating object for other class.**

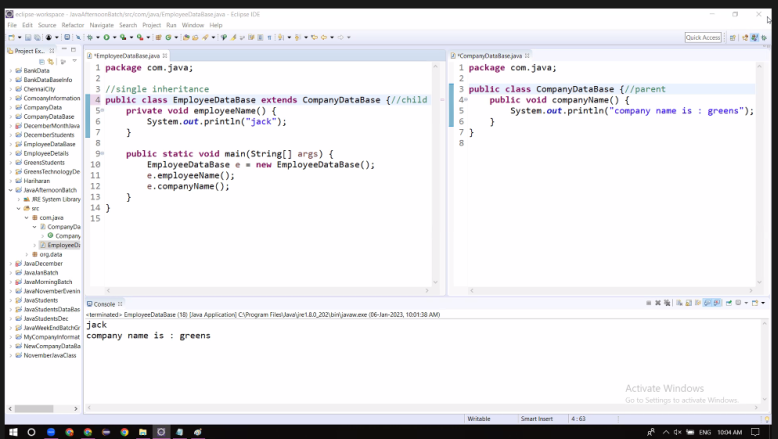
Because, when we create object for other class, its takes RAM memory for it.

But, if we use inheritance method, it will use the child’s class memory.

**Types of inheritance:**

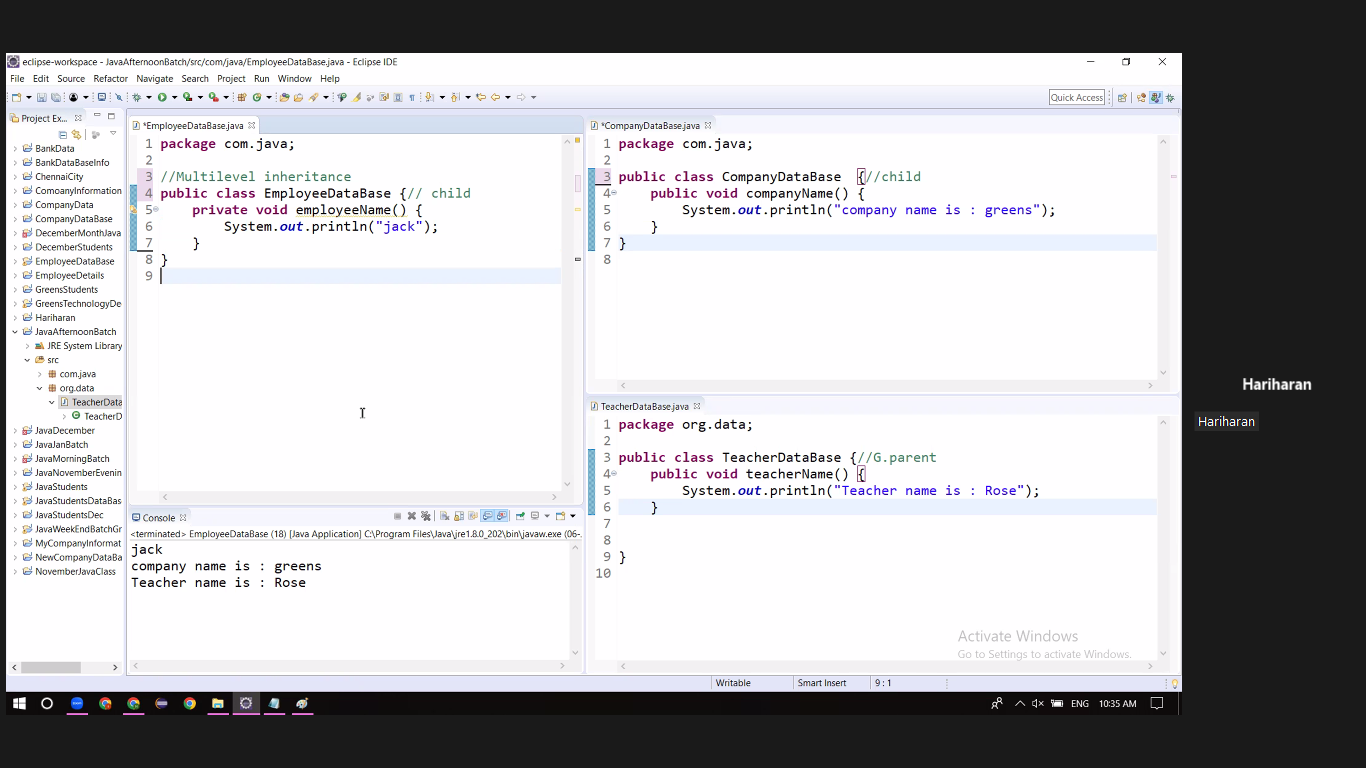
**Single inheritance:**

Combination of single child and single parent.



**Multi-level inheritance:**

More than one parent class accessing the child class in a tree level structure.



**Multiple inheritance:**

More than one parent class accessing one child class parallel at the same time.

**Hybrid inheritance:**

Combination of Single and Mutiple inheritance.

**Note:**

Multiple Inheritance is not possible in java using the class.

* Priority problem
* Sytax error (By using extends keyword we cannot access two class parallely)
* To Overcome this issue we use interface keyword.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY5

**Polymorphism:**

One method than can be executed in many ways is called polymorphism. Executing methods in more than one forms.Poly🡪many, Morphism –-> form

**Two types of polymorphism:**

1. Method Overloading or compile time Polymorphism./ early binding
2. Method Overriding or Runtime polymorphism./ late binding
3. **Method Overloding:**

Same method name with different arguments in a single class. When we are going to overload a method again and again with same method name by different its argument. The arguments depends on data type, datatype order and data type count. Same classname, Same methodname, Different arguments.

1. **Method Overriding**

When we are not satisfied with the parent class method. We can create the same methos in our child class and we can write our required business logics. Different classname, Same methodname, Same arguments.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

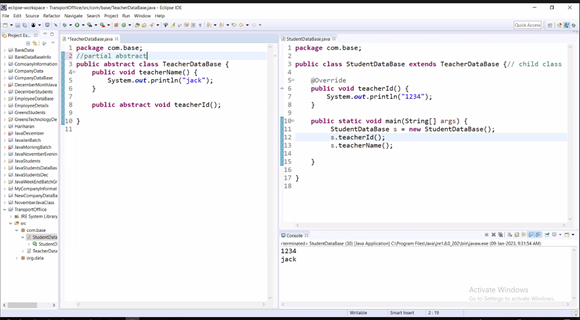
DAY6

**Abstraction**:

Hiding the implementation details or business logic details.

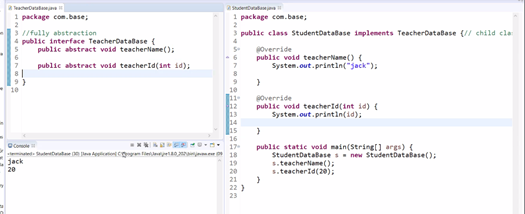
**Types of abstraction:**

1. **Partial abstraction:**

* Combine both abstract and non abstract methods
* Contain keywords extends
* Use the keyword abstract in class and method.
* We cant create object for abstract class.
* Dnt have any default return type.
* 

1. **Fully abstraction**

* Contain only the abstract methods.
* Contain keywords implements
* Use a keyword interface instead of class
* We can’t create object for a interface
* Default return type is public abstract.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY7

**Control Statements:**

1. Conditional
2. Looping
3. Breaking or terminating
4. **Conditional Statements:**

This statement is used to validate the condition. If the condition is satisfied the program will be executed, if the condition is not satisfied the program will not be executed.

**If Loop:**

If(condition){ business logic}

**Comparing Operator:**

* Greater
* Lesser
* Greater and equal
* Lesser and equal
* Equals to
* Not equals

**If Else Loop:**

If(condition){business logic;}

Else{business logic;}

**Logical Operators:**

If you pass more than one condition in if block

AND Operrator –Multiplication

OR Operator 🡪 Addition

AND Operator &&:

OR Operator ||

**Bitwise Operators:**

AND(&)

OR(|)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY8

**Looping:**

Rather than repeating the same logic again and again means we can go for looping.

**For Loop:**

for(int i=0;i<=3;i++){

system.out.println(i)

}

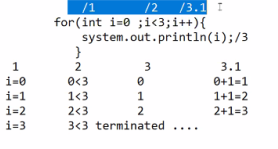
Step1: i=0 initailazation

Step2: i<=10 condition check. If conditions pass, move to Step3. If fails terminate.

Step3: print statement

Step 4: increment/decrement.

Step5: loop conti.

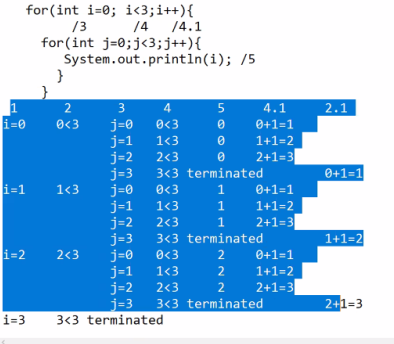


***Nested For Loop***

For (intial, Condition, in/dec)

For(intial, condition. Inc/dec)

Business logic;



Step1) I loop 🡪 initial

Condition checking

J loop 🡪 initial

Condition checking

Bussiness logic exection

J++;

***Nested While Loop/Entry check loop***

Initialization;

While(condition) {

Business logic

Inc/dec

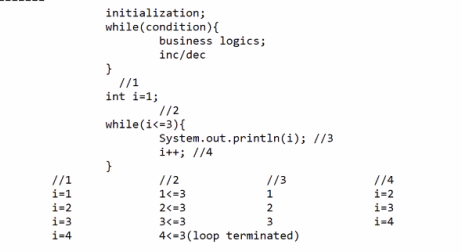
}

Step1: initialization

While condition

Business logic

Inc/dec



**Do while loop / Exit check loop:**

**Syntax**:

Inialization;

Do

Business logic;

Inc/dec

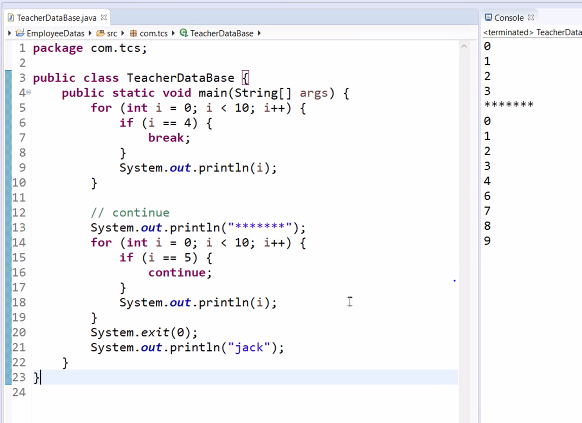
Condition check

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY9

**Breaking/Terminating statements:**

1. Break; - to break only the loop for a given condition
2. Continue ; to skip the loop for the given condition only
3. System.exit(value) – to terminate the entire programme



**Switch Case:**

It is used to check the Quality of a variable against multiple values.

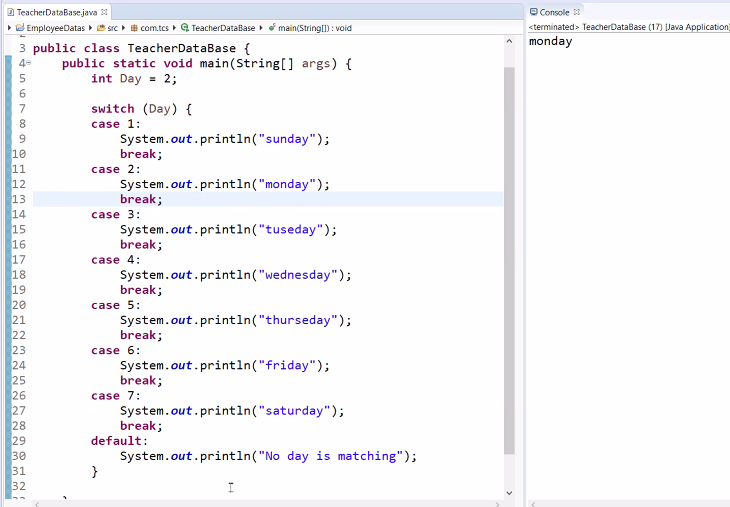
Syntax: switch(variable name)

Case value: business logic;

Case value: business logic;

Case value: business logic;

Default: business logic



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY10

**Array:**

* We can store only the similar datatype.
* Once we fixed the size we cant modified.
* Memory wastage is high.
* Complie time memory allocation.
* Before assigning data in the array,the array always has default value.

Datatype var[] = new datatype[size];

Int arr[] = new int[5];

If we give print(arr) – without index, it will display the memory adderss.

**Note Exception** -🡪 java.lang.ArrayIndexOutBoundsException 🡪 Index 5 out of bounds for length 5

|  |  |
| --- | --- |
| **For Loop** | **For Each Loop** |
| Index based | Value based |
| Control the iteration | We cant control the iteration |
| Initialization, condition, inc/dec | No initailazation , no condition, no inc/dec |
| IndexOutOfBoundException | No Exception |

**Array:**

We can store multiple values of similar datatypes in a single variable.

Similar datatypes

Index Based

Index starts for 0 to n-1

It will support duplicates.

Array Syntax: Datatype variable[] = new Datatype[size];

**For Each Loop:**

for(datatype variable name: variable)

Loop will terminate , if there is no new value to the variable.

**Disadvantages of Array:**

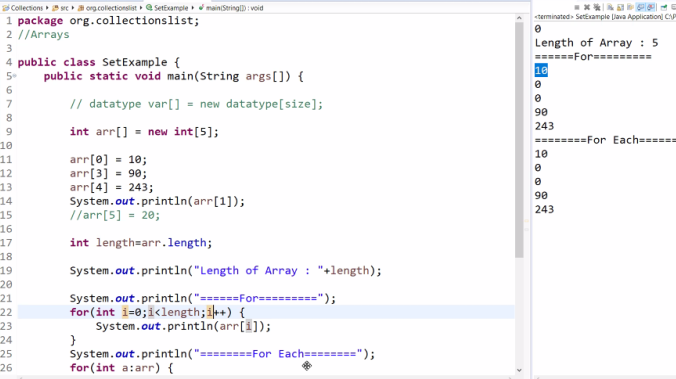
We can store only the similar datatypes

Once we fixed the size we cant modified.

Memory wastage is high.

Compile time memory allocated.

Note: To overcome, this disadvantage we have collections.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

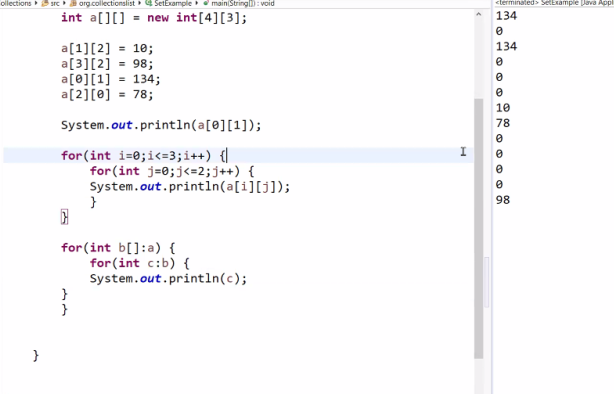
DAY11

2D Array:

Able to create multiple index.

Datatype var[][] = new datatype[size][size];

int a[][] = new int[4][4];



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY12

**Why do go for collections(is an interface):**

It will support dissimilar data types.

Provides runtime allocation/dynamic memory allocation

No memory wastage like array.

It has 2 types:

1. List 🡪 Interface
2. Set 🡪 Interface

Class a = new Class();

Interface objRef = new Class();

List Li = new ArrayList();

**List(Interface):**

Its used to store multiple values of dissimilar data types in a single variable.

List is index based (0 –n-1).

List will allow the duplicates.

List accept null value.

**ArrayList(class)**

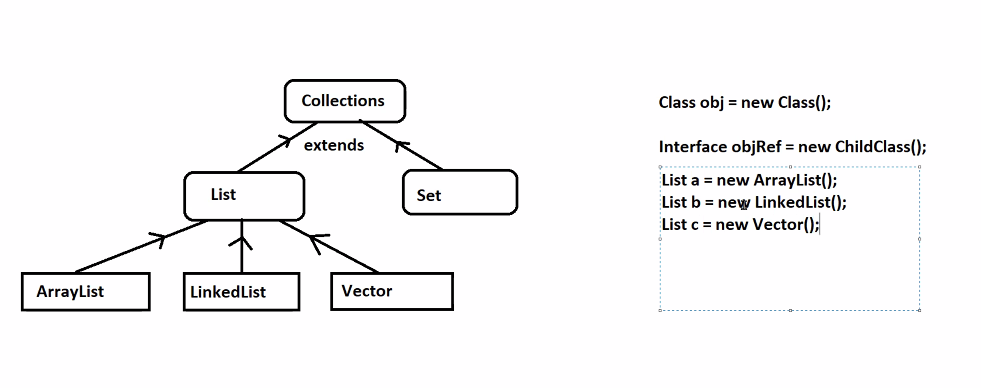
**LinkedList(class)**

**Vector(class)**

**List ref = new ArrayList();**

**List ref = new LinkedList();**

**List ref = new Vector();**



We can’t create an object for interface. We can create object reference.

Note: when import java.utils.list. we should choose java.utils only, we have same class in aws also(java.aws.list)

Object is the super-most parent class of all predefined and user defined classes and interface.

Wrapper Classes:

Is the classes of the datatypes.

Byte -🡪 Byte

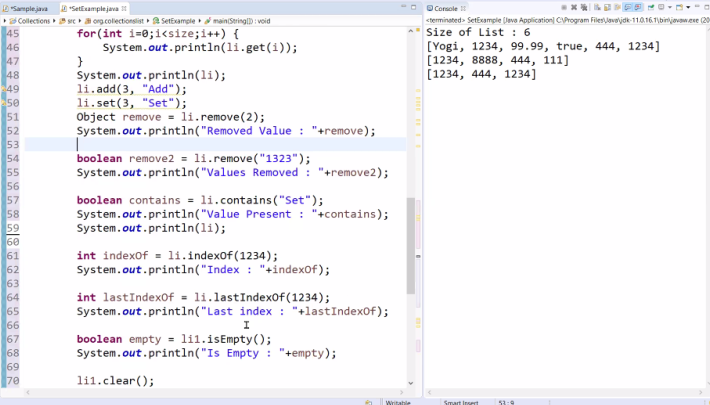
Short 🡪 Short

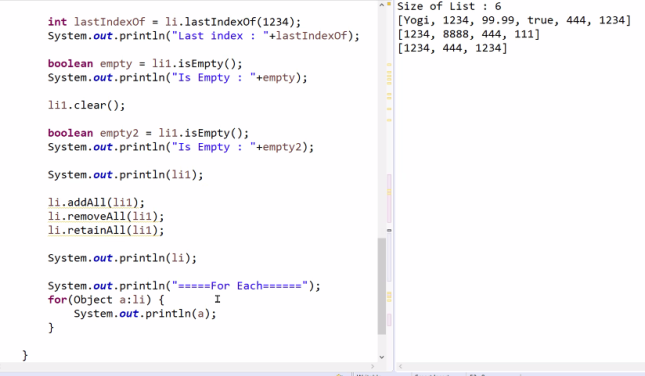
Int 🡪 Integer

Char 🡪 Character

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY13





**ArrayList:**

* After jdk 1.5
* Parallel Execution(Asynchronous)
* Fast
* Non Thread safe
* Index Based
* Add/Remove – Difficult
* Searching and retrieving a value is easy.
* Get()

**Linked List:**

* After jdk 1.5
* Parallel Execution(Asynchronous)
* Fast
* Non Thread safe
* Nodal Based
* Add/Remove – Easy
* Searching and retrieving a value is easy.
* Get()

**Vector:**

* Before jdk 1.5
* Sequential Execution(Asynchronous)
* Slow
* Thread safe

**List:**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Datatype** |
| add(“value”) | Add the value into the list | Boolean |
| add(index, “value”) | Add the value into the respective index | Boolean |
| addAll() | Copy onelist to another list | boolean |
| clear() | Clear the complete list | Boolean |
| isEmpty() | Check whether the list is empty or not. | Boolean |
| int hashcode | Prints the hashcode of the list | Integer |
| Set(index, element) | Replace the element in the particular position. | Boolean |
| Get(index) | Pint the values present in the list | Object |
| Indexof(value) | Prints the index of the given value . | Integer |
| lastIndexOf(value) | Prints the index of the given value .(in case of duplicate) | Integer |
| Remove(value) | Remove particular value from list. | Boolean |
| removeAll(value) | Remove all the elements from the list. | Boolean |
| RetainAll() | Retain all the elements in the list. | Boolean |
| Size() | Return the size of list | Integer |
| Contains() | Check whether the element is present in the list or not | Boolean |
| ContainsAll() | Check all the elements of the list are present in the set or not. | Boolean |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY14

**Set:**

* Used to store multiple values of dissimilar data type in a single reference.
* Based on – value
* It don’t allow duplicate.
* Window Handling use set, because it will not allow duplicate.
* Set s1 = new Hashset();

**Classes of Set:**

1. Hashset(C) ----random Order ---one null – no duplicates
2. Linked Hash Set(C) --- insertion order ---one null
3. Tree set(C) –ascending Order -- doesn’t allow null vaule

**Note: In set, we cant use for loop🡪 we dnt have index.**

**Null:**

* All Wrapper classes and classes default value is null.
* Undefined/unknown/unassigned value
* Won’t create any memory.
* Tree set will give exception in compile time, if its null.

**Methods Present in Both List and Set:**

* Add()
* Size()
* Contains()
* Clear()
* addAll()
* remove()
* removeAll()
* retainAll()
* -🡪 forEach

**Methods Supported by list but not in set:**

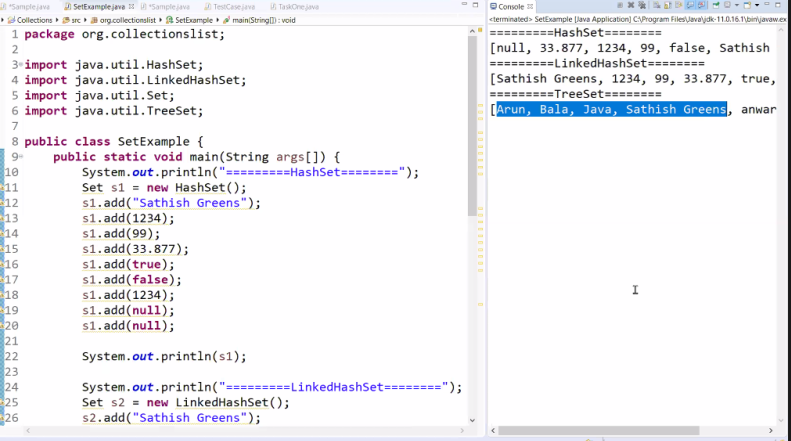
* Add(index, value)
* IndexOf(Value)
* lastIndexOf(value)
* set(index,value)
* get(index)
* -🡪for

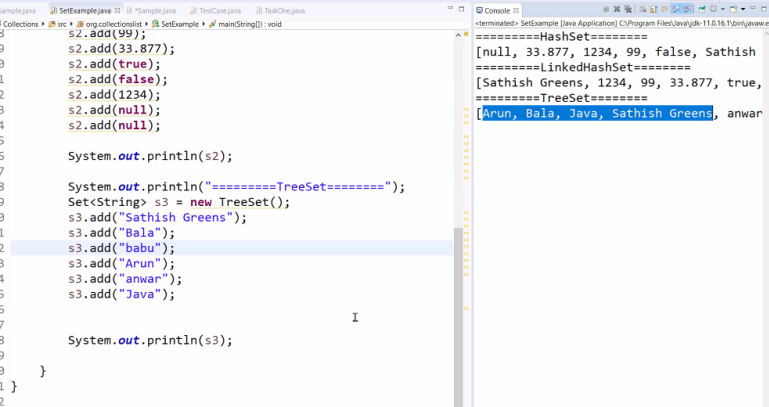
Treeset --🡪 Generics -🡪 ClassCastException

In Treeset, we cant use dissimilar data types. Print in ascending order.

SET:

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Datatype** |
| add(“value”) | Add the value into the set | Boolean |
| addAll() | Copy onelist to another set | Boolean |
| clear() | Clear the complete set. | Boolean |
| isEmpty() | Check whether the set is empty or not. | Boolean |
| Remove(value) | Remove particular value from set. | Boolean |
| removeAll(value) | Remove all the elements from the set. | Boolean |
| RetainAll() | Retain all the elements in the set. | Boolean |
| Size() | Return the size of set | Integer |
| Contains() | Check whether the element is present in the set or not | Boolean |
| ContainsAll() | Check all the elements of the set is present in the set or not. | Boolean |

****

****

**Note:**

**List – index based(allow duplicates) 🡪 get(0).. 0-index**

**Set – value based(not allow duplicates). In set, we dnt have index. So we cant use index based methods.**

**MAP:**

* We can add pair of values.
* Not comes under collections.
* Mapping one data with other data
* Key,value 🡪 pair or combination
* Key don’t allow duplicates
* Values allow duplicates
* Map<k,v>
* Map<String, Integer> m = new HAshMap<string, Integer> (Note: this side generic is not mandatory)
* M1.put(“Kar”,65)

**Classes in MAP:**

HashMap --- Random 🡪 key =1 null -🡪 values = n null

LinkedHashMap --- Insertion 🡪 key =1 null -🡪 values = n null

TreeMap --- Ascending Order 🡪 key =ignore null -🡪 values = n null

HashTable --- Random 🡪 key =ignore null -🡪 values = ignore null

**Difference between hashtable and hashmap:**

Hashmap:

* Asynchronize -🡪 allows users parallel
* Key allow 1 null and value allow n null
* Non thread safe

HAshTable:

* Synchronize 🡪 Allows user one by one
* Ignore null values in key and value
* Thread safe

{DD=10, MI=15, CSK=12, RR-18}

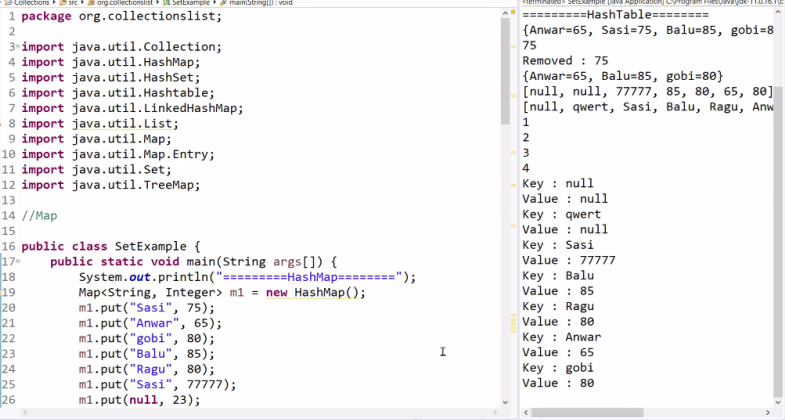
* Entry<string, Integer>
* DD=10
* MI=15
* CSK=12
* RR =18

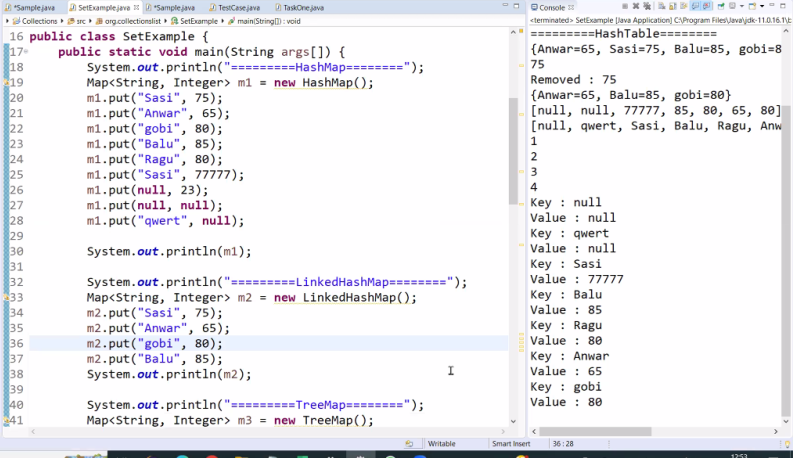
**MAP Iteration:**

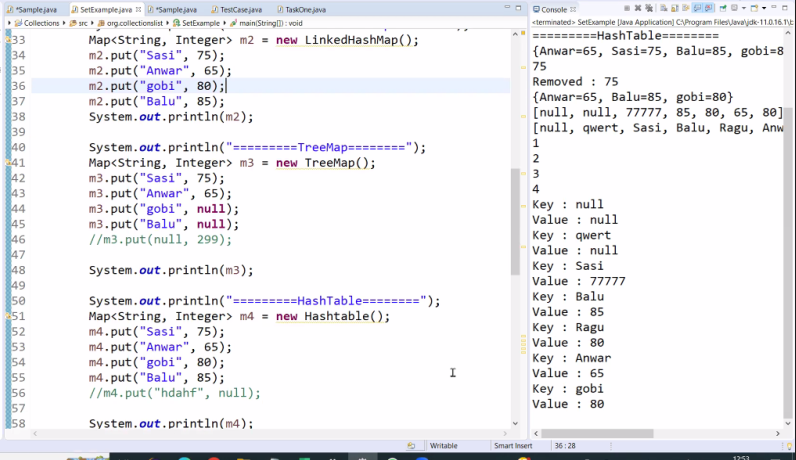
We should convert map to set. Because map has 2 set of value. So we need to convert it into single value.

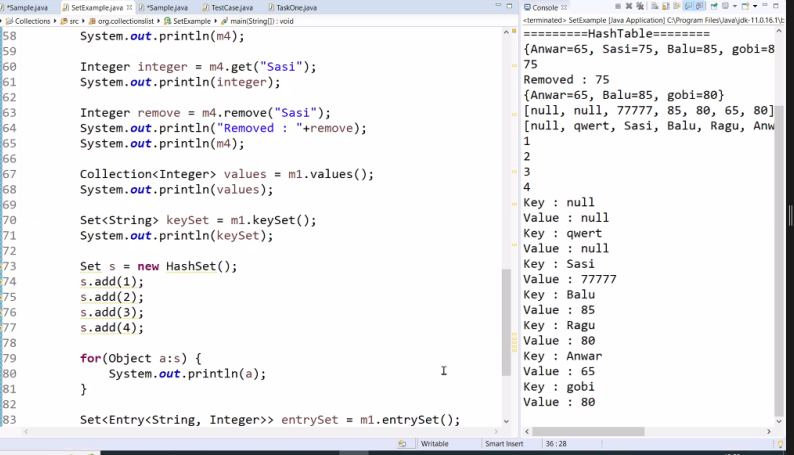
M1.entryset();

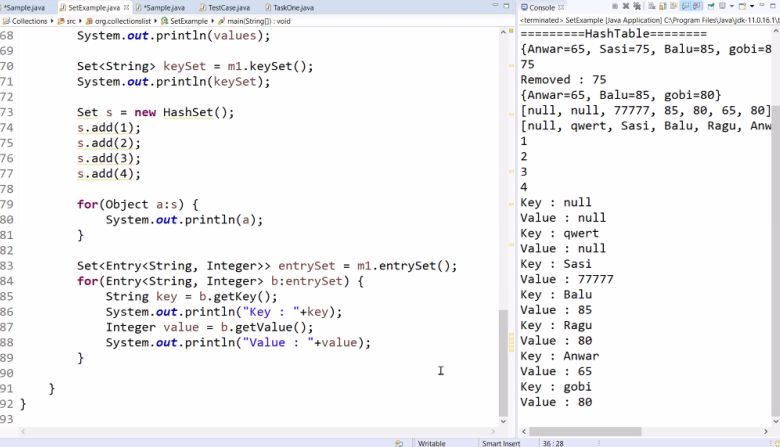
Entry is a predefined class. Which converts(key, value) into single entry.











**Map**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Return Datatype** |
| Put(key, value) | Used to insert an entry in the map. | Boolean |
| Get (key) | Get the value of object key | Object |
| Remove(key) | Remove an entry for the particular key. | Boolean |
| Size() | Return the size of map | Integer |
| Values() | Displaying the values only | Object |
| Keyset | Displaying only the keys | Set |
| clear() | Clear the complete list | Boolean |
| containsValue(Object value) | Check the given value is exists within the map | Boolean |
| containsKey(Object Key) | Check the given key is exists within the map | Boolean |
| putAll() | Insert all the specified map into another map | Boolean |
| entrySet() | For iterating map | Set <<Entry>> |
| getKey() | Diplaying the corresponding keys values | Object |
| getValues() | Displaying the corresponding values | Object |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY14

**Exception Handing:**

* Sudden interruption which cause a flow of programme to terminate.
* Exceptions are predefined classes java.lang
* Exception should be handled, because it will terminate the entire program.

Types of Exception:

* Checked Exception
* Unchecked Exception

**Unchecked Exception[run time]**

Whenever the exception will occur in runtime it is called run time exception.

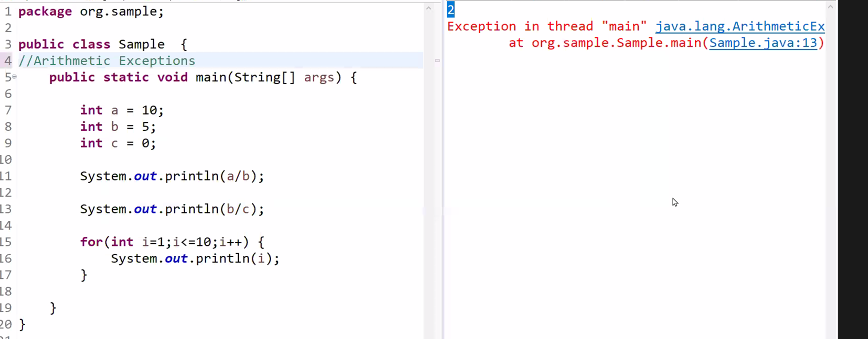
* Arithmetic Exception
* Null pointer Exception
* Array IndexOutofBound exception
* StringIndexOutofbound Exception
* IndexOutofBound Exception
* NumberFormat Exception
* NEgativeArraySizeException

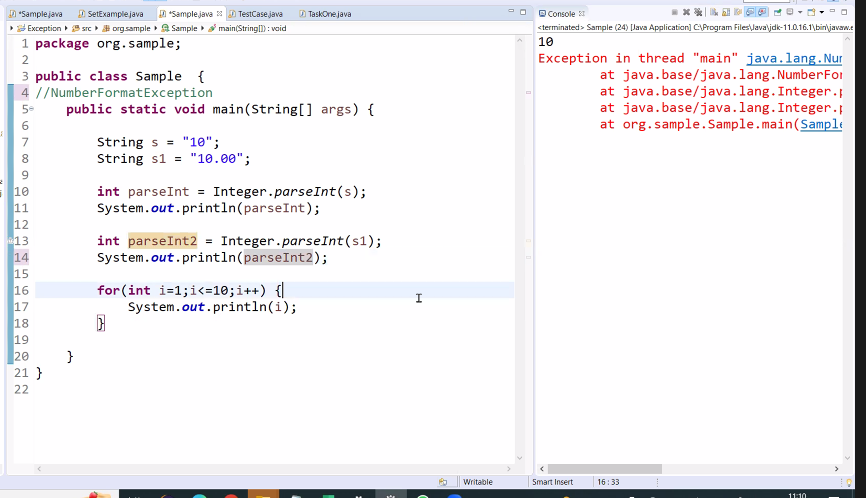
**Checked Exception[compile time]**

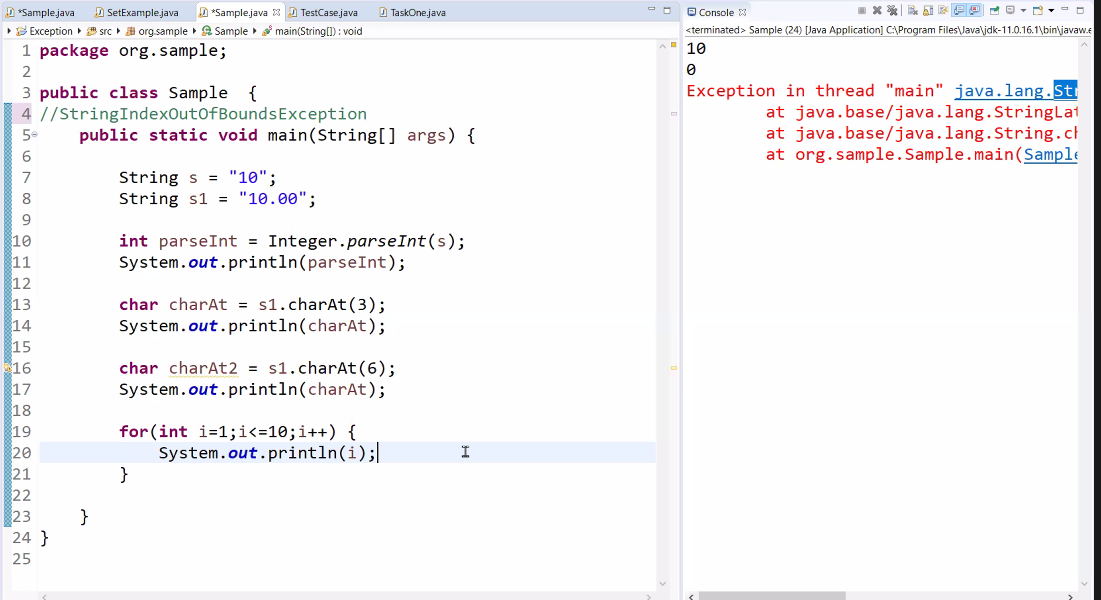
Whenever the exception will occur in compile time its called compile time execution.

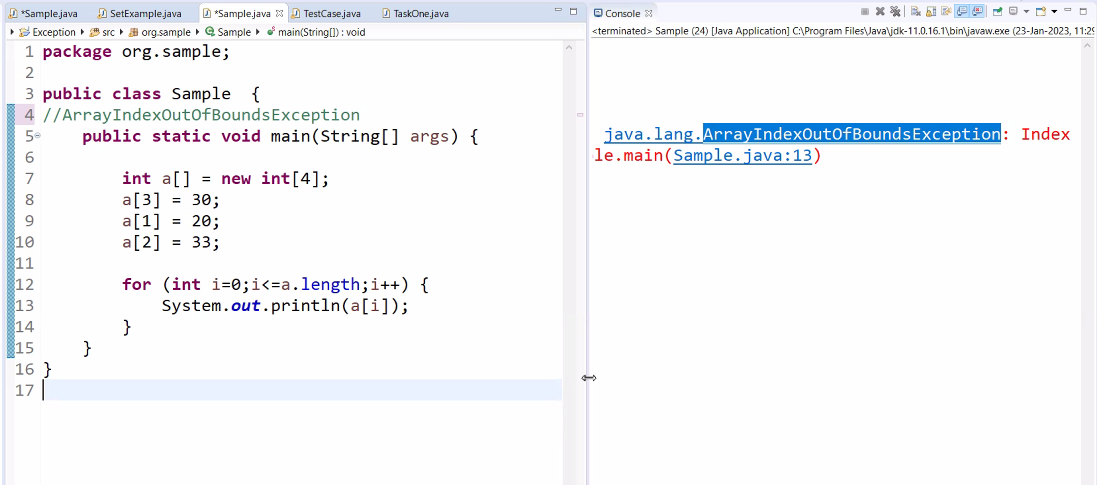
* File not found exception ---🡪 whenever we create a new file.

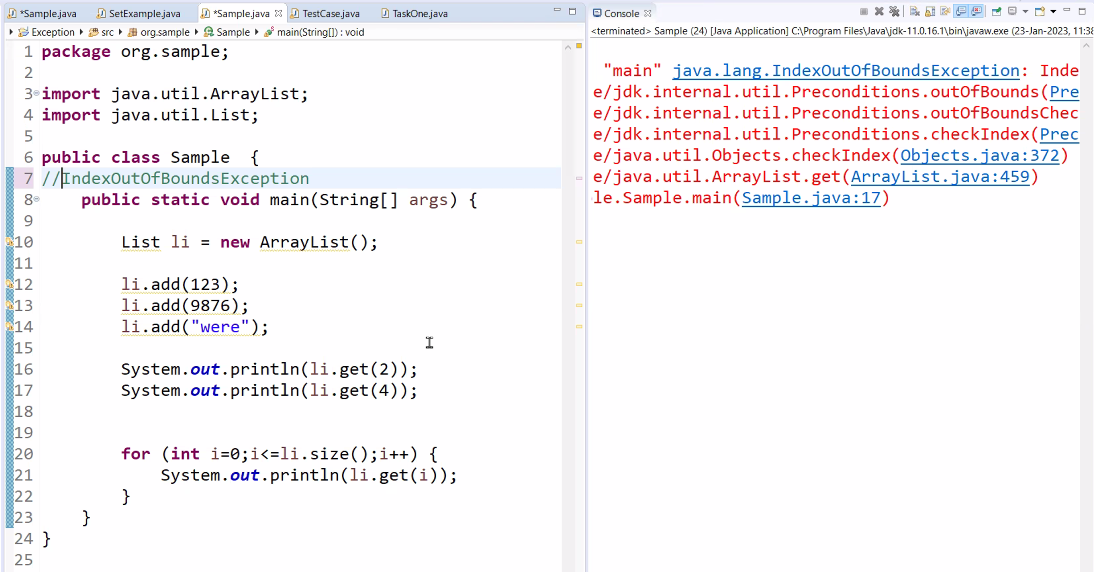
**Arithmetic Exceptions:**

****

****

****

****

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY15

Exception Handling:

* If we don’t know that specific Exception, we need to used general exception class.

catch(Exception e)

* All exception has some message (message is an predefined method in the respective exception class.)

Try

Catch

finally

**Combination:**

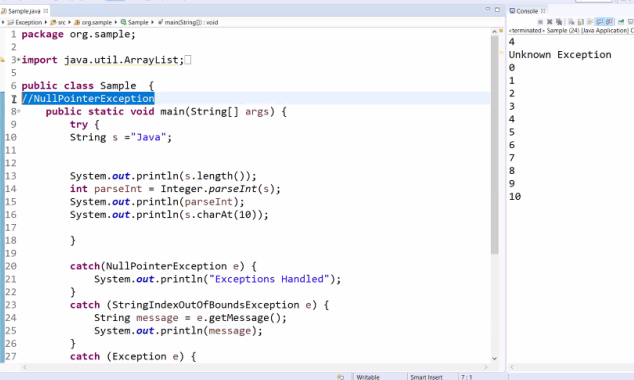
Try- catch

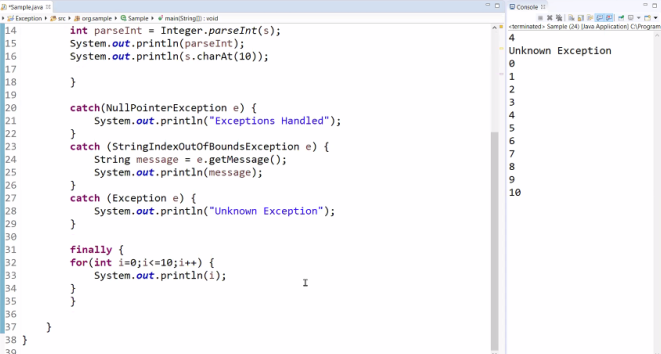
Try – multicatch

Try –finally

Try – catch – finally

Try – multicatch – finally

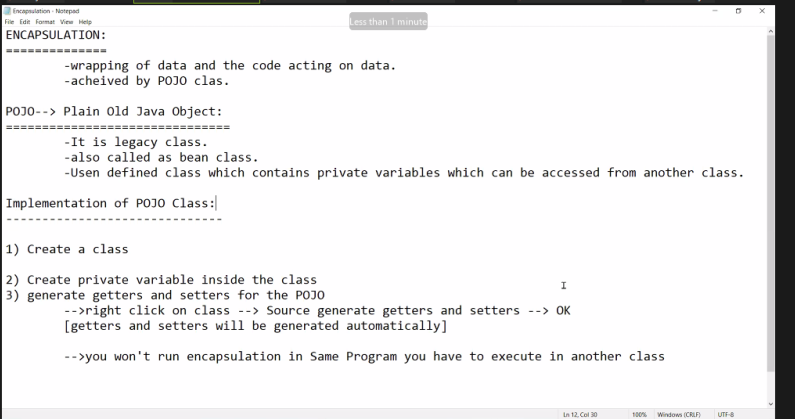


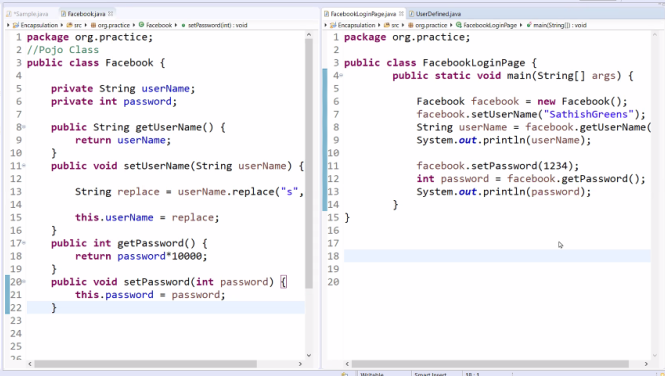


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DAY15

**Encapsulation:**





File Operation

The interaction between the files in local machine and Java Programme.

Possible Interaction:  
1) how to create New Folder on Directory: 🡪 File (Predefined Class in Java.io)

File file = new File(“Path of the file”); 🡪 return type is Boolean

🡪 file.mkdir();

IO Exception: Compile time exception

🡪 Boolean mkdir(); = file.mkdir();

1. To create New File:

Create object for File class

🡪 createNewFile();

3) To Check the created file is readable or not:

🡪 obj.canwrite();Boolean

* Obj.canRead();

1. List File Name and Folder in console:

* Objref.list(); 🡪 string[]
* Objref.listFiles(); 🡪 File[]

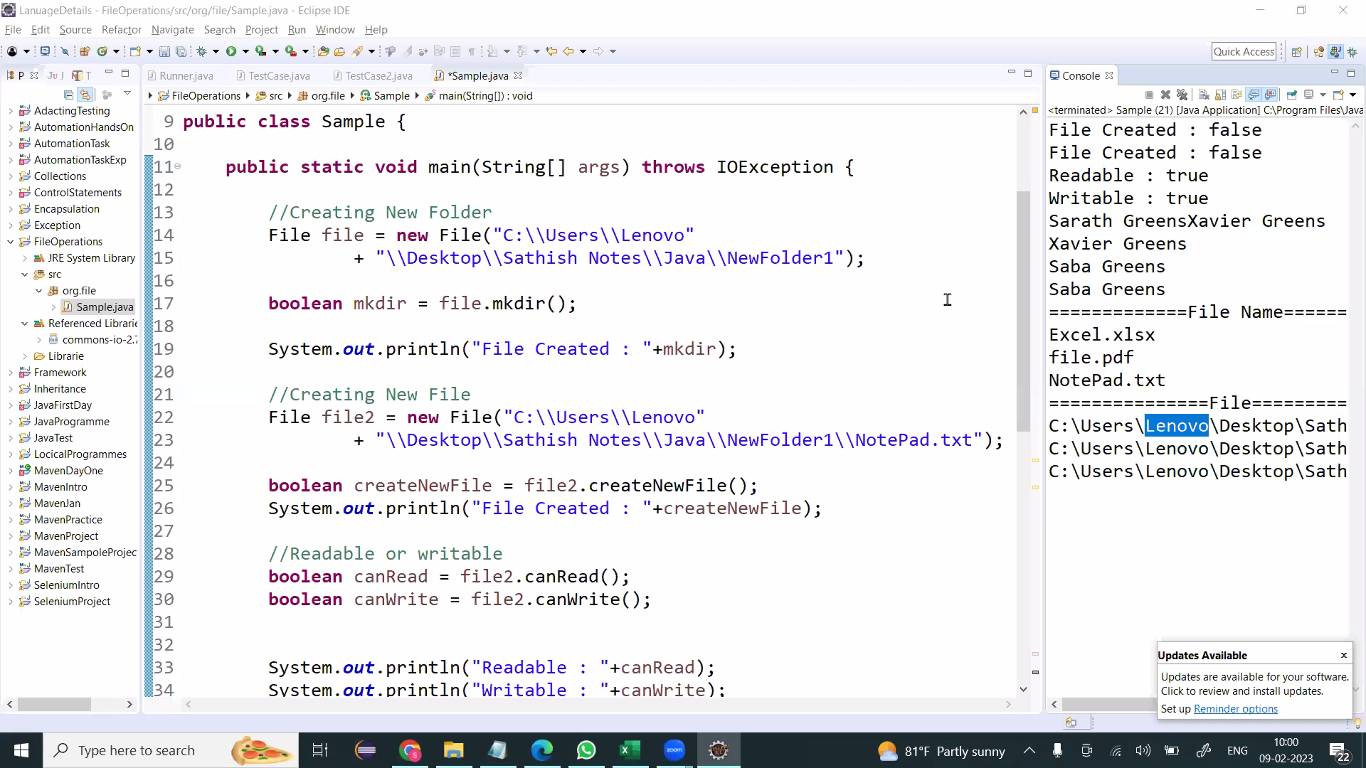
1. How to Read the data from File, How to modify the existing data

* FileUtils – Predefined class(commons.io package not in java (JRE))
* (we have download jar) (copy jar and create new folder and part and have to integrate by right click build path -> buildpath)
* List<string> readline = FileUtils.readLines(file);
* Sysout (readLine);

1. To Modify Data

Write();

FileUtils.write(file, “Maven”)





To Import File Utills

