**Array list:**

Array list Is a class that Implements list interface In java which supports dynamic array that is the size of array can grow as needed.

It can re-size automatically by Itself.

**USED in SELENIUM:**

Store all links from page, store all buttons Id from page etc.. In all such case, You will not aware about how many values you have to store.

**Program:array\_list**

**package**arr;

//Import ArrayList class header file

**Import** java.util.\*;

**Public class** ArrayList\_Example {

**publicstaticvoid** main(String[] args) {

//Create object of ArrayList class. It will store only string values.

ArrayList<String> Sample = **new**ArrayList<String>();

//Now you can store any number of values In this arraylist as bellow. Size constrain will comes never.

Sample.add("giri") ;//Putting an Item In arraylist at Index = 0.

Sample.add("button2"); //Putting an Item In arraylist at Index = 1.

Sample.add("button3"); //Putting an Item In arraylist at Index = 2.

Sample.add("button4"); //Putting an Item In arraylist at Index = 3.

**for**(**int**i=0; i<Sample.size();i++){//loop will execute till size of arraylist.

System.***out***.println(Sample.get(i)); //print arraylist values one by one.

}

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//To get the Index of an Item from arraylist.

**Int** ItemIndex = Sample.indexOf("button3");

System.***out***.println("Index Of button3 Item = "+ItemIndex);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Sample.remove(1);//To remove an Item from arraylist.

**for**(**int**i=0; i<Sample.size();i++){

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

}

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Sample.set(1, "Button8");//To reset value of an arraylist item.

**for**(**int**i=0; i<Sample.size();i++){

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

}

}

}

**Hash table**

Hash table Is a class In java and provides us a structure to store key and its value as a pair In table format. Means we can store value with Its key and can access that value using Its key. There Is not any Index of value In Hash table. Key's Hash code will be used to map the value with key In Hash table.

**Program:hash\_table**

**package**hash\_table;

**import**java.util.Hashtable;

**publicclass** Hash {

**publicstaticvoid** main(String[] args) {

//Created hashtable class object to use Its different properties.

Hashtable<String, Integer>t1 = **new**Hashtable<String, Integer>();

t1.put("Legs", 4); //Store value 4 In key = Legs

t1.put("Eyes",2); //Store value 2 In key = Eyes

t1.put("Mouth",1); //Store value 1 In key = Mouth

//Accessing hash table values using keys.

System.***out***.println("Animal Legs = " +t1.get("Legs"));

System.***out***.println("Animal Eyes = " +t1.get("Eyes"));

System.***out***.println("Animal Mouth = " +t1.get("Mouth"));

}

}

**String:**

String Is an Inbuilt class of java. String class has many Inbuilt functions which we can use to perform different actions on string.

Program:string classes and functions

**package**strng;

**publicclass**Strng\_Example {

**publicstaticvoid** main(String[] args)

{

String st1 = "This World is Very Nice";

String st2 = " And Beautiful.";

//Comparing two strings. Return true If both match else return false.

System.***out***.println("st1 equals to st2? -> "+st1.equals(st2));

//Concatenates st2 with st1.

System.***out***.println("Concatenation of st1 and st2 Is ->

"+st1.concat(st2));

//Retrieve the 9th Indexed character from string.

System.***out***.println("Character at Index 9 Is -> "+st1.charAt(9));

//Find the length of string.

System.***out***.println("Length Of St1 -> "+st1.length());

//Converting whole string In lower case.

System.***out***.println("String In Lowercase -> "+st1.toLowerCase());

//Converting whole string In upper case.

System.***out***.println("String In uppercase -> "+st1.toUpperCase());

//Retrieve the Index of first 'i' character.

System.***out***.println("Index of 1st charater i Is -> "+st1.indexOf('i'));

//Retrieve the index of 2nd most 'i' character.

System.***out***.println("Index of 2nd charater i Is -> "+st1.indexOf('i', 3));

//Retrieve the Index of word 'Very' from string.

System.***out***.println("Index of word Very Is -> "+st1.indexOf("Very"));

//Converting value Fromint to string.

**int**j = 75;

String val2 = String.*valueOf*(j);

System.***out***.println("Value Of string val2 Is -> "+val2);

//Converting string to integer.

String val1="50";

**int**i = Integer.*parseInt*(val1);

System.***out***.println("Value Of int i Is -> "+i);

//Print the String starting from 5th Index to 12th Index.

System.***out***.println("Retrieving sub string from string ->

"+st1.substring(5, 13));

//Split string.

String splt[] = st1.split("Very");

System.***out***.println("String Part 1 Is -> "+splt[0]);

System.***out***.println("String Part 2 Is -> "+splt[1]);

//Trim String.

System.***out***.println("Trimmed st2 -> "+st2.trim());

}

}

**Exception Handling in Java**

The **exception handling in java** is one of the powerful *mechanism to handle the runtime errors* so that normal flow of the application can be maintained.

### Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. Exception normally disrupts the normal flow of the application that is why we use exception handling. Let's take a scenario:

1. statement 1;
2. statement 2;
3. statement 3;
4. statement 4;
5. statement 5;/ /exception occurs
6. statement 6;
7. statement 7;
8. statement 8;
9. statement 9;
10. statement 10;

Suppose there is 10 statements in your program and there occurs an exception at statement 5, rest of the code will not be executed i.e. statement 6 to 10 will not run. If we perform exception handling, rest of the statement will be executed. That is why we use exception handling in java.

## Java Exception Handling Keywords

There are 5 keywords used in java exception handling.

1. try
2. catch
3. finally
4. throw
5. throws

## Problem without exception handling

Let's try to understand the problem if we don't use try-catch block.

1. public class Testtrycatch1{
2. public static void main(String args[]){
3. int data=50/0;//may throw exception
4. System.out.println("rest of the code...");
5. }
6. }

## Solution by exception handling

Let's see the solution of above problem by java try-catch block.

1. public class Testtrycatch2{
2. public static void main(String args[]){
3. try{
4. int data=50/0;
5. }catch(ArithmeticException e){System.out.println(e);}
6. System.out.println("rest of the code...");
7. }
8. }

## Difference between checked and unchecked exceptions

### 1) Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException, SQLException etc. Checked exceptions are checked at compile-time.

### 2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.