Crack Selenium Interview:

Locators:

Id, Name, Tag name, Class Name, linkText, Partial Link Text, CssSelector, Xpath

---

Syntax for finding element: [Also learn the syntax for each locator]

Driver.findElement(By.Locator("locator Value"));

Example: Driver.findElement(By.Id("idvalue"));

****

For Opening Browser:

fireFoxDriver driver = new fireFoxDriver();

ChromeDriver driver = new ChromeDriver();

InternetExplorerDriver driver = new InternetExplorerDriver();

Note: For browsers other than Firefox we have to set the browser driver (exe) file and declare where it is placed before instantiating the browsers.

Example:

For Chrome:

System.setProperty("Webdriver.chrome.driver", "D://Foldername//chromedriver.exe");

For IE:

System.setProperty("Webdriver.iedriver.driver", "D://Foldername//iedriver.exe");

****

To maximize the window:

driver.manage().window().maximize();

****

Implicit and Explicit Waits:

Interview Question: Difference between Implicit and Explicit waits

Implicit wait:

Selenium will check for the element and wait for the particular time before throwing an exception. This applies to every element (every line of code) throughout the entire script.

Driver.manage().timeouts().implicitlywait(10, timeUNIT.SECONDS);

Explicit Wait:

Sometimes we cannot judge the time taken for a button to get enabled for clicking. In such cases we can explicitly ask selenium to wait for a particular condition.

WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("someid")));

****

Xpath Syntax:

Absolute Xpath: (Starts from html: uses single slash: "/")

/html/tbody/table/td/tr

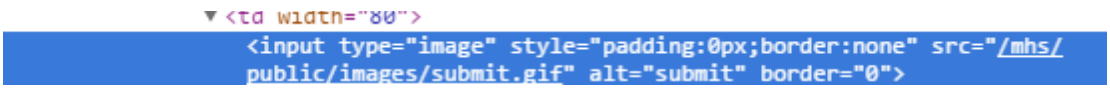
Relative Xpath: (uses double slashes "//")

//tagname[@attribute='attribute value'] or //*[@attribute='attribute value']

Interview Question: (Also you can add it as the challenge faced in automation)

When xpath keeps changing dynamically: go for partial xpath:

Example:



//input[contains(@src , 'submit')]
//input[ends-with(@src , 'submit.gif')]
//input[starts-with(@alt , 'submit')]

Interview Question: (In a web table (dynamic) find the particular word in a row and print its column values)

Refer this site:

http://www.toolsqa.com/selenium-webdriver/handling-tables-selenium-webdriver/

****

Read and Write Excel : Apache POI

Interview Question:

1. How will you handle Excel?: Ans: two jars available: JXL and Apache POI. Mention the advantage of Apache POI over JXL. Also make sure to know the latest version of Apache POI.

****

Get Methods:

getAttribute(); => This will return the value of any attribute in the source code of that element.

Example:

<div id="gs_lc0" style="position: relative;”>

Driver.findElement(By.id(“gs_lc0”)).getAttribute(“style”); => this will return “position: relative;”

driver.getTitle(); => To get the title of the website

driver.getCurrentURL(); => To get the URL of the website

driver.findElement(By.id(“idvalue”)).getText(); => get the text

Dimension dim = driver.findElement(By.id(“idvalue”)).getSize(); (=>getSize() will return a Dimension: store it in a variable)

int Height = dim.height; int Width = dim.width;

Point p = = driver.findElement(By.id(“idvalue”)).getLocation(); (=>getLocation() will return a Point: store it in a variable)

Intxposition = p.x; intyposition = p.y;

****

Handling Alert:(Alerts can not be inspected: If there is no Alert in the screen you will get “NoAlertPresentException”)

Alert a = driver.switchto().alert();

a.accept(); =>to click ok

a.dismiss(); =>to click cancel

a.sendKeys("text"); =>To enter text in the alert

a.getText(); =>To get the text present on the alert

Handling Frames: (If tag name starts with "Frame" or "Iframe" then it is a frame)

Syntax: driver.switchto().frame();

Frame can be found in 3 ways:
•driver.switchTo().frame(name_or_id)
If the frame tag has unique id or name, use it.
•driver.switchTo().frame(index)
This is the last option to choose, because using index is not stable enough
as you may know. If this is only iframe in the page,
trydriver.switchTo().frame(0)
•driver.switchTo().frame(driver.findElement(By.className("value")));
The most common one. Locate your frame/iframe like other
elements, then pass it into the method.

Once you are done with the frame: you have to move out of frame:

driver.switchTo().defaultContent();

****

Handling Windows:(refer material)

Driver.switchTo().window(winHandle);

Interview Question:

Difference between windowHandle and windowHandles:

Suppose your test script opens 3 new tabs/windows: Selenium webdriver assigns unique handle to each window. But it can get the Window Handle of the parent window alone using the "getWindowHandle()" method. To get the window Handle of other windows we have to use "getWindowHandles()" method which would return a "Set" of strings. (Set is a collection class in Java and the reason for using "Set" instead of "List" is window handles are unique values and Set allows only uniqiue entry.)

Then using "for each" loop iteration : we have to iterate through each window handle and complete testing. Finally if we want to move back to parent window: we can use its windowHandle which we stored initially.

****

Actions Class:(To perform keyboard and mouse actions. Actions should end with build().perform();)

(Mouse hover and Drag and Drop methods are enough)

//Drag and drop code

WebElement ele1 =driver.findElement(by.id("idval"));

WebElement ele2 =driver.findElement(by.id("idval1"));

Actions builder = new Actions(driver);

Builder.dragAndDrop (ele1,ele2).build().perform();

//Mouse hover code

Builder.moveToElement(ele1).build().perform();

****

Selecting Dropdown Value

Interview Question:

How will you select dropdown value?

Select s = new select(driver.findElement(By.id("idval")));

//We can select dropdown by 3 ways:

1.Select by Value:

s.SelectByValue("2"); => This will select the drop down option for which value is set as "2" in the source code

2. Select by index:

s.SelectByindex(5); => This will select the drop down option at the fourth postion. (Indexing starts with 0 hence 5-1 = 4)

3. select by Visible Text:

s.SelectByVisibleText("someText"); =>This will select the drop down option with the text value "someText".

Interview Question:

How will you select the last option in the dropdown?

s.getOptions(); => This will return me the total number of options. Then using the select by index we can select the desired option.

Interview Question:

Without using "select" how will you select drop down value? Ans: Using Actions class we can do that. Identify the element. Using the click method of Actions class click the dropdown and then click down arrows using the method sendKeys(keys.Down) and then click the desired element.

****

Grid:

Grid is used to achieve remote execution. One machine will act a Hub (Master) and it will check for the connected machines called Nodes and based on the Actual capabilities will send the requests to the nodes and execute on them.

Desired Capabilities: These are the user desired values like run my test script in Chrome and Windows platform.

DesiredCapabilities dc = new DesiredCapabilities();

dc.setBrowserName("chrome");

dc.setPlatform(platform.Windows);

RemoteWebdriver driver = new RemoteWebdriver(dc);

Actual Capabilities: If 2 machines (Nodes) are connected to my Hub: the Hub will check for the machine (Node) with Chrome Browser and Windows platform and send execute the script on that.

Code for starting a Hub:

In the command prompt type the following and hit enter:

java –jar jarfilenameincludingversion –role=hub –port=4444

(Here 4444 is a default port number and we can give any value:

example: java –jar Selenium-Standalone-Server-2.46.0.jar –role=hub –port=4444)

Code for starting a Node:

In the command prompt type the following and hit enter:

Java –jar jarfilenameincludingversion –role=node –hub http://ip address of Hub/localhost:4444/grid/register

Example: java –jar Selenium-Standalone-Server-2.46.0.jar –role=node –hub http://123.10.10.0/localhost:4444/grid/register

****

Framework:

Advantages of Framework:

1. Pre-Built and readily available.
2. Code reusability and no need to write again.

Types of Framework:

1. Modular : Follows an ordered folder structure
2. Data Driven : The scripts uses data from Data files like excel and validate input & output.
3. Keyword Driven : The scripts uses wrapper methods + excel read/write
4. Hybrid : Combination of all the above

Interview Question:

What is the type of framework you are using?

Ans: Hybrid with TestNG

Components in a framework:

1. Input: (This will contain my driver + scheduler to invoke the scripts)
2. Resources: (This will contain the scripts, jar files, driver files and other essential things required for execution).
3. Output: (This will contain the reporter (TestNG + Excel based reporter to store the results/screenshots).

Folder Structure of my Framework:

S.No	Folder	Contains
1	Data	Test Data
2	bin	All my .class files
3	library	Jar files (selenium, Apache etc)
4	Src	Scripts
5	Driver	Driver.exe of diff. Browsers
6	Result	Results+Reports+Snapshots
7	Config	configuration details
8	Help	Help/supporting doc

Working / Flow of Framework:

1. The driver file is called manually. Also Windows scheduler provides option to run the driver via batch file on a particular date and time.
   Note: The same can also be achieved by configuring Jenkins/Git/Maven. When a new Build is deployed in Jenkins, it will get reflected in Git (A cloud based tool) and it will trigger the Maven to run driver file.
2. The batch file by using the design will call the TestNG xml file using the code "java –cporg.testng.testNGXMLfilename.Xml".
3. The XML file using the design will invoke the TestNG classes (scripts) as RemoteWebdriver scripts and send them to the Hub.
4. The Hub checks the desired capabilities and based on the actual capabilities of the nodes will send the scripts to them and execute on them.
5. The results of execution will be stored in Results folder. We use both the TestNG default test results (HTML format) + our own customized excel based reporting.

Interview question:

Explain your framework? Ans: Tell everything from the above section.

****

TestNG:

TestNG is a framework similar to Junit.

Advantages of TestNG:

1. More and detailed annotations.
2. Paralllel execution.
3. Test Groups (Grouping scripts into groups like sanity/smoke/regression and executing group of tests).

Annotations and their Heirarchy inTestNG:

@BeforeSuite ->@BeforeTest -> @BeforeClass -> @BeforeMethod -> @Test -> @AfterMethod ->@AfterClass ->@AfterTest -> @AfterSuite

@Test Attributes:

Invocation Count: To run the script multiple times. (example: @Test (invocation count =2)

Enabled: To run/skip a script. (Example: @Test(enabled = true)) [By default the enabled will be set as true. If you set that as false that script will be skipped.]

Timeout: To set timeout for a script.

Groups: To group the scripts into a group. (example: @Test(Groups =”regression”)

Priority: To set priority to the execution order of @Test (Scripts) . (example: @Test (Priority =1) [Note: Priority takes the order of least value to a higher value. If two @Test have priority 2 and 5 respectively: @Test with priority 2 will be executed first).

Note: Also take a look at other attributes like DependsonGroups, DependsonMethods.

Interview Questions:

1. Explain the Hierarchy of TestNG Annotations.
2. Suppose if I have 3 @Test in my script what is their execution order. (Ans: The @Test scripts will be executed alphabetically)
3. If I want to run the 3rd @Test first, how to achieve that? (Ans: Use the @Test attribute "priority")
4. How many times @BeforeSuite, @BeforeTest and @BeforeMethod will be executed?
   Ans: @ BeforeSuite will be executed only once.
   @BeforeMethod will be executed before each method within @Test annotation.
   @BeforeTest will be executed before each "test tag" – (<test>) in the TestNG XML file.

StructureofTestNG XML File:

```
<Suite name="Default Suite">
<Test name ="Default Test">
<Classes>
<Class name="Test.TC001" />
<Class name="Test.TC002" />
</Classes>
</Test>
</Suite>
```

Parall Execution:

In the above code if we change the first line of code as:

<Suite name="Default Suite" parallel ="Classes">

The classes TC001 and TC002 will be executed parallel.

Challenges faced in Automation:

Important Interview question: Tell any 2

1. Screenshots taken in browsers other than “Firefox” will not cover the entire webpage. Only the area with the element of focus will be covered. Solution: Use Actions class to move (scroll) the webpage up/down and take screenshots.
2. Dynamically changing xpath: Solution: Used partial xpath to handle it.
3. Need to start scripting before the application is ready. That is we have mockup screen alone. Solution: Using Properties file concept and handled it. Once the application is ready we will simply change the code in the properties file rather than changing script.
4. Need to upload a document from my computer as part of sanity testing which would need windows application interactions. We can achieve it using tools like Sikuli or Autoit. But in client machines we should not install such tools. Solution: There is a method “Robot.awt” under “Java.AWT” package. [Concept: store the file path using the “setContents” method to store in clipboard. Then using sendKeys method send “Ctrl+V” and “Enter.”
5. Reading contents from a PDF file. Solution: Two jar files – “PDF Box” to read PDF and “Font box” to read/analyze” the contents in the PDF. (Limitation: It cannot read images/tables. Only text will be read.)
6. Run execution in the background or Hidden mode. Solution: HTML Unit driver is used to run scripts in the background. That is we don’t want any browser to run in the foreground. [Limitation of HTML unit driver: it cannot take screenshot. Mouse hover, Key board actions cannot be performed].

****

JAVA Topics

Basic OOPS:

1.Inheritance:

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.

It is achieved by using the keyword “extends”

```
class Employee{
 int salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

Output:

Programmer salary is:40000

Bonus of programmer is:10000

2.Polymorphism:

Polymorphism is the ability of an object to take on many forms. It is achieved by Method overriding and Method overloading concepts.

2.a.Method Overloading:

If a class have multiple methods by same name but different parameters (or argumenents), it is known as Method Overloading. Also we can say as when the method arguments are either different by type or count.

```
class Calculation{
  void sum(int a,int b){System.out.println(a+b);}
  void sum(int a,int b,int c){System.out.println(a+b+c);}
  void sum(float d, float e){System.out.println(d+e);}
```

Output: 30

40

16

```
  public static void main(String args[]){
 Calculation obj=new Calculation();
 obj.sum(10,10,10);
 obj.sum(20,20);
obj.sum(10.5, 5.5);
   }
}
```

2.b. Method overriding:

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding.**

```
class Vehicle{
void run(){System.out.println("Vehicle is running");}
}
class Bike2 extends Vehicle{
void run(){System.out.println("Bike is running safely");}

public static void main(String args[]){
Bike2 obj = new Bike2();
obj.run();
}
```

Output:

Bike is running safely

3.Constructor:

- Constructor is like a method.
- Name of the constructor should be the name of the class.
- no return type.
- constructor will be called whenever you create an instance of the class.
-Types: a.Default Constructor b.Parameterized Constructor

3.a.Default Constructor:
Constructor without parameter. Used to give default value to the object.

```
Class Bike{
Bike(){
System.out.println("Bike is created");
}
Public static void main (String[] args) {
Bike B = new Bike();
}}
```

Output:

Bike is created

3.b Parameterized Constructor:

A constructor that have parameters is known as parameterized constructor. Parameterized constructor is used to provide different values to the distinct objects.

```
class Student{
   int id;
   String name;

   Student(int i,String n){
   id = i;
   name = n;
   }
   void display(){System.out.println(id+" "+name);}

   public static void main(String args[]){
   Student s1 = new Student(111,"Karan");
   Student s2 = new Student(222,"Aryan");
   s1.display();
   s2.display();
  }
}
```

Output:

111 Karan

222 Aryan

4. Abstract Class:

A class that is declared as "abstract" is known as Abstract class.

- It needs to be extended.
- Its methods needs to be implemented.
- Abstract class can not be instantiated.

5.Abstract Methods:

–A method that is declared as abstract and does not have implementation is known as abstract method.
•A generic operation
•Part of an abstract class
–Must be implemented by a concrete subclass
•Each concrete subclass can implement the method differently
•Real usage:
- If you cannot define concrete implementation of method in superclass; then you can define it as abstract method. So that you force the subclass developer to implement the logic.
- e.g., break this can vary between types of car.. [sports car VS normal car]; just define the break as abstract method in super class; so the sub class developers implement the behavior.
- All the methods in sub class should definitely be implementing the behavior else the class becomes abstract again.

```
abstract class Shape{
abstract void draw();
//Note: Both class name and method name are declared as abstract
}

class Rectangle extends Shape{
```

```
void draw(){System.out.println("Drawing rectangle");}
}

class Circle extends Shape{
void draw(){System.out.println("Drawing circle");}
}

class TestAbstraction{
public static void main(String args[]){
Shape s1=new Circle();
//In real scenario, object is provided through method e.g. getShape() method
//Note: you can also use Circle s1 = new Circle(); instead of Shape S1 = new Circle();
Shape S2=new Rectangle();
S1.draw();
S2.draw();
}
}
```

Output:

Drawing circle

Drawing rectangle

6.Interface:

A big Disadvantage of using abstract classes is not able to use multiple inheritance. In the sense, when a class extends an abstract class, it can't extend any other class. So we go for Interfaces.

Interface is:
Abstract classes which are completely unimplemented.
- Every method is abstract.
- A class implement any number of interfaces.
•All the methods in interface should be implemented in subclass else it becomes abstract class.
•Cannot be instantiated

Defining an Interface in Java
•Use interface keyword instead of class
•All methods public abstract

```
public interface Steerable {
//Note: The methods should be public and abstract. (There is no definition of methods. Only declaration.)
public void turnLeft();
 public void turnRight();
}
```

Example code:

```
interface Steerable {
public void turnLeft(int deg);
public void turnRight(int deg);
interface Mode{
```

```
public void frontWheel();
public void backWheel();
}

public class Car implements Steerable, Mode{
        public void turnLeft(int deg){
                System.out.println("Turn left " +deg + " degrees");
        }
        public void turnRight(int deg){
                System.out.println("Turn right " +deg + " degrees");
        }
        public void frontWheel() {
                System.out.println("Front wheel drive");
        }
        public void backWheel(){
                System.out.println("Back wheel drive");
        }

        public static void main(String[] args) {
                Car c = new Car();
                c.backWheel();
                c.frontWheel();
                c.turnLeft(23);
                c.turnRight(40);
        }

}
```

Output:

Back wheel drive

Front wheel drive

Turn left 23 degrees

Turn right 40 degrees

Java Keywords

Static:

- Can be applied to variable / method / inner-class / blocks
- Memory efficient [Saves memory] e.g., common property for all objects [company name of employees]
- Java static property is shared to all objects. One time declaration is enough.

Example of static variable:

```
class Student{
  int rollno;
  String name;
  static String college ="ITS";  //String variable college is declared as static

  Student(int r,String n){
  rollno = r;
  name = n;
  }
 void details (){
System.out.println(rollno+" "+name+" "+college);
}
```

```
 public static void main(String args[]){
 Student s1 = new Student(111,"Karan");
 Student s2 = new Student(222,"Aryan");

 s1.details();
 s2.details();
 }
}
```

Output:

111 Karan ITS

222 Aryan ITS

Example of static method:

```
//Program of changing the common property of all objects.
class Student{
  int rollno;
  String name;
  static String college ="ITS";  //String variable college is declared as static

static void change(){
    college = "AUC";
    }

  Student(int r,String n){
  rollno = r;
  name = n;
  }
 void details (){
System.out.println(rollno+" "+name+" "+college);
}

 public static void main(String args[]){
 Student s1 = new Student(111,"Karan");
 Student s2 = new Student(222,"Aryan");

 s1.details();
 s2.details();
 }
}
```

Output:

111 Karan AUC

222 Aryan AUC

Final:

- **Applicable to class, variable, method. If you declare final then its value cannot be altered. Using final, Immutability can be achieved.**
- **If you want the class to final give ; so that no one can extend the class.. no subclass.**
- **for method you cant override.**
- **All final variables, class, method do not participate in inheritance.**
- **final method can be overloaded.**

```
Class testFinal{
final int a =5;
public static void main (string[] args){
System.out.println(a);
```

//If you give System.out.println(a++); - you will get compilation error as the value of a is 5 and cannot be changed since its declared as final.

Super:

The **super** keyword in java is a reference variable that is used to refer immediate parent class object.

Usage of java super Keyword

1.super is used to refer immediate parent class instance variable.
2.super() is used to invoke immediate parent class constructor.
3.super is used to invoke immediate parent class method.

```
1) super is used to refer immediate parent class instance variable.
class Vehicle{
  int speed=50;
}
class Bike extends Vehicle{
  int speed=100;
  void display(){
   System.out.println(speed);//will print speed of Bike
  }
void display1(){
   System.out.println(super.speed);//will print speed of Vehicle
  }
  public static void main(String args[]){
   Bike b=new Bike();
   b.display();
   b.display1();
}
}
```

Output:

100

50

```
2) super is used to invoke parent class constructor.
class Vehicle{
  Vehicle(){System.out.println("Vehicle is created");}
}
  class Bike extends Vehicle{
  Bike(){
   super();//will invoke parent class constructor
   System.out.println("Bike is created");
  }
  public static void main(String args[]){
   Bike b=new Bike();
    }
}
```

Output:

Vehicle is created

Bike is created

3) super can be used to invoke parent class method

```
class vehicle{
void run(){
System.out.println("vehicle is created");
}
}

class car extends vehicle{
void run(){
System.out.println("car is created");
}

void display(){
message();//will invoke current class message() method
super.message();//will invoke parent class message() method
}

public static void main(String args[]){
car c = new car();
c.display();
}
}
```

Output:

Car is created

Vehicle is created

This:

This keyword is a reference variable that refers to the current object.

Usage of java this keyword:

- **this keyword can be used to refer current class instance variable.**
- **this() can be used to invoke current class constructor.**
- **this keyword can be used to invoke current class method (implicitly)**
- **this can be passed as an argument in the method call.**
- **this can be passed as argument in the constructor call.**
- **this keyword can also be used to return the current class instance.**

In the above examples, if you mention this instead of super: the current class objects will be considered. (But it is not necessarily to mention this while calling current class methods because due to method overriding concept, the current class method will be implicitly called.)

****

Collections in Java

1.Array:

- Fixed length data structure.
- To get the size use "length()" method.

```
public class Array {
```

Output:

The array elemnts are :1
The array elemnts are :2
The array elemnts are :3
The array elemnts are :4

```
public static void main(String[] args) {
int a[] = {1,2,3,4,5};
for(int i=0; i<a.length; i++){
System.out.println("The array elemnts are :" +a[i]);
}
}
}
```

2.Array list:

- Variable length collection class.
- To get the size use “size()” method.

```
public class Arraylist {
public static void main(String[] args) {
ArrayList<String> al = new ArrayList<String>();
al.add("one");
al.add("two");
al.add("three");

for(String obj:al){
        System.out.println(obj);
}

//to get the index of an item - use indexof() method
System.out.println("The position of two in the array list is "+ al.indexOf("two"));
        }

}
```

Output:

```
one
two
three
The position of two in the array
list is 1
```

3.Hash Map:

- **A HashMap contains values based on the key. It implements the Map interface and extends AbstractMap class.**
- **It contains only unique elements.**
- **It may have one null key and multiple null values.**
- **It maintains no order.**

```
class Hashmap{
 public static void main(String args[]){
  HashMap<Integer,String> hm=new HashMap<Integer,String>();
   hm.put(100,"Amit");
  hm.put(101,"Vijay");
  hm.put(102,"Rahul");

  for(Map.Entry m:hm.entrySet()){
   System.out.println(m.getKey()+" "+m.getValue());
  }
 }
}
```

Output:

```
102 Rahul

100 Amit

101 Vijay
```

4.Hash Table:

- A Hashtable is an array of list.Each list is known as a bucket.The position of bucket is identified by calling the hashcode() method.A Hashtable contains values based on the key. It implements the Map interface and extends Dictionary class.
- It contains only unique elements.
- It may have not have any null key or value.
- It is synchronized.

```
public class TestHashtable{
public static void main(String[] args){

Hashtable<Integer, String> ht = new Hashtable<Integer, String>();
ht.put(1, "Rahul");
ht.put(2, "Amit");
ht.put(3, "Vijay");
System.out.println("items in the hashtable are :"+ht);
        }

}
```

Output:

items in the hashtable are :{3=Vijay, 2=Amit, 1=Rahul}

Note:

Also learn List and Set collection class.
Set will be used while handling windows (storing the window handles –driver.getWindowHandles() method).
List will be used while getting a collection of webelements. findElements(by) method.

****

Frequently Asked Java Programs

1. Factorial of a Number:

```
public class Factorial {
public static void main(String[] args) {
int num=5, fact=1;
for(int i=1; i<=num; i++){

fact=fact*i;
}
System.out.println("The Factorial of the number "+num +" is :" +fact);
}
}
```

2. Odd / Even Number:

```
public class OddorEven {
public static void main(String[] args) {
int num = 44;
if(num%2==0)
System.out.println("the given number is even");
else
System.out.println("the given number is odd");
}
}
```

3. Swap two variables without Third variable:

```
public class SwapNumbers {
public static void main(String[] args) {
int a =1, b=2;
System.out.println("Before Swapping: a = " +a +" and b = " +b);

                a=a+b;
                b=a-b;
                a=a-b;
System.out.println("After Swapping: a = " +a +" and b = " +b);
        }
}
```

4. Duplicates in an Array:

```
public class ArrayDuplicate {

public static void main(String[] args) {

String[] strArray = {"abc", "def", "mno", "xyz", "pqr", "xyz", "abc"};

      //1. Using Brute Force Method

      for (int i = 0; i < strArray.length-1; i++)
      {
         for (int j = i+1; j < strArray.length; j++)
         {
            if( (strArray[i].equals(strArray[j])) && (i != j) )
            {
               System.out.println("Brute Force Method : Duplicate Element is : "+strArray[i]);
            }
         }
      }

      //2. Using  Colections -HashSet

      HashSet<String> hs = new HashSet<String>();

      for (String arrayElement : strArray)
      {
         if(!hs.add(arrayElement))
         {
            System.out.println("HashSet :Duplicate Element is : "+arrayElement);
         }
      }
   }
}
```

5. Fibonacci Series:

```
public class FibonacciSeries {

public static void main(String[] args) {
int febCount = 15;
        int[] feb = new int[febCount];
        feb[0] = 0;
        feb[1] = 1;
        for(int i=2; i < febCount; i++){
           feb[i] = feb[i-1] + feb[i-2];
        }

        for(int i=0; i< febCount; i++){
              System.out.print(feb[i] + " ");
        }
}
}
```

6. Palindrome Number:

```
public class Palindromenumber {
public static void main(String[] args) {

int rem,sum=0,temp;
  int num=252;//It is the number variable to be checked for palindrome

  temp=num;
  while(num>0){
   rem=num%10;  //getting remainder
   sum=(sum*10)+rem;
   num=num/10;
  }
  if(temp==sum)
   System.out.println("palindrome number ");
  else
   System.out.println("not palindrome");
}
}
```

7. Palindrome String:

```
public class StringPalindrome {

public static void main(String[] args) {
String text = "madam";

/**************Using String buffer methods**************/

String rev = new StringBuffer(text).reverse().toString();

if (rev.equalsIgnoreCase(text))
System.out.println("Validated with String Buffer Method: The given string is palindrome");
else
System.out.println("Validated with String Buffer Method:The given string is not palindrome");

/************************************************************/


/**************Without Using String buffer methods**************/

String reverse="";
for(int i=text.length()-1; i>=0; i--){

reverse = reverse+text.charAt(i);

}


if (rev.equalsIgnoreCase(text))
System.out.println("Validated without String Buffer Method: The given string is palindrome");
else
System.out.println("Validated without String Buffer Method:The given string is not palindrome");


}

}
```

8. Prime Number:

```
public class PrimeNumber {
public static void main(String[] args) {
int num =21, i, flag=0;

//Prime number is one that is not divisible by the numbers starting from "2" to half of that number
//example: For 10: it is prime if it is not divisible by the numbers: 2,3,4,5 where 5 is half of 10

for(i=2; i<num/2; i++){

if(num%i==0){
System.out.println("The given number is not prime");
flag=1;
break;
}
}
if(flag==0)
System.out.println("The given number is prime");
}
}
```

****

********** All The Best...! **********