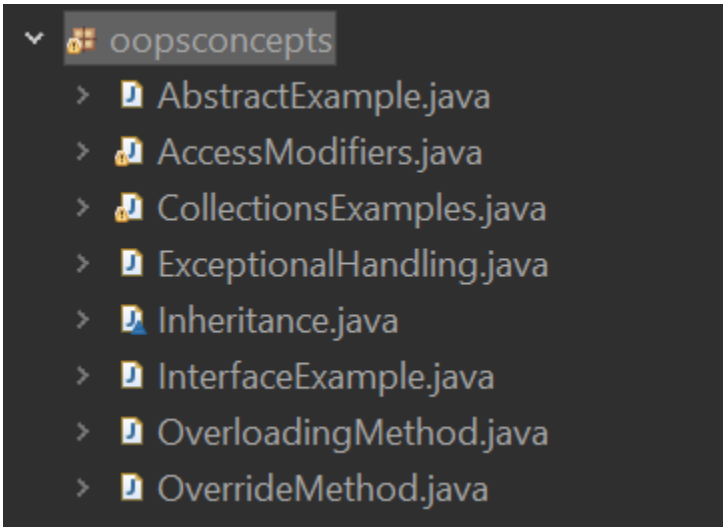
<u>Steps to JAVA Selenium Projects</u>

Automation Testing Stream

NAME- *KRITTIKA AGRAWAL*

CONTENTS- *Object Oriented Programming*

Step-1: Created Classes for Package- “oopsconcepts”.



Step-2: Code for Abstract Classes.

```java
package oopsconcepts;
abstract class Abc{
    abstract void m1();
    void m2() {
        System.out.println("M2 CODE");
    }
}
public class AbstractExample extends Abc {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        AbstractExample obj = new AbstractExample();
        obj.m1();
    }
    void m1() {
        System.out.println("M1 CODE");
        m2();
    }
}
```

Step-3: Code for Override Method.

```java
package oopsconcepts;
class Animal{
    void printing() {
        System.out.println(" This is the Grandfather Class!!");
}
}
class Animal1 extends Animal{
    void printing() {
        super.printing();
        System.out.println(" This is the Father Class!!");
}
}
public class OverrideMethod extends Animal1{
    void printing() {
        super.printing();
        System.out.println(" This is the Child Class!!");
    }
    public static void main(String[] args) {
        OverrideMethod obj = new OverrideMethod();
        obj.printing();
    }
}
```

Step-4: Code for Array List, Linked List and Vector List- collections.

```java
public class CollectionsExamples {
    public static void main(String[] args) {
        // ArrayList
        Collection values=new ArrayList();
        values.add("Krittika");
        values.add(1);
        values.add(33.22);
        Iterator i= values.iterator();
        while(i.hasNext())
        {
            System.out.println("Array List: "+ i.next());
        }
        values.remove(33.22);
        for(Object i1 : values)
        {
            System.out.println(i1);
        }
        values.add(2);
        for(Object i1 : values)
        {
            System.out.println(i1);
        }
        // LinkedList
        LinkedList<String> l1 = new LinkedList<String>();
        l1.add("Krittika");
        l1.add("Agrawal");
        l1.addLast("Ran");
        l1.addFirst("Anil");
        l1.add(2, "Harish");
        System.out.println("Linked list: "+l1);
        l1.remove("Anil");
        l1.remove(3);
        l1.removeFirst();
        l1.removeLast();
        System.out.println(l1);
        //VectorList
        Vector<Integer> v = new Vector<Integer>();
        for (int j = 1; j <= 5; j++)
            v.add(j);
        System.out.println("Vector list: "+v);
        v.remove(3);
        System.out.println(v);
        for (int k = 0; k < v.size(); k++)
            System.out.print(v.get(k) + " ");
```

Step-5: Code for Hash Set and Linked Hash List-collections.

```java
        //Hashset
        HashSet<String> set = new HashSet<String>();
        set.add("HI");
        set.add("To");
        set.add("ALL");
        set.add("For");
        set.add("READ");
        System.out.println("\nHashSet: " + set);
        set.remove("HI");
        System.out.println("HashSet after removing elements: " + set);
        //LinkedHashSet
        LinkedHashSet<String> linkedset = new LinkedHashSet<String>();
        linkedset.add("A");
        linkedset.add("B");
        linkedset.add("C");
        linkedset.add("D");
        linkedset.add("A");
        linkedset.add("E");
        System.out.println("Size of LinkedHashSet = "+ linkedset.size());
        System.out.println("Original LinkedHashSet:"+ linkedset);
        System.out.println("Removing D from LinkedHashSet:"+ linkedset.remove("D"));
        System.out.println("Trying to Remove Z which is not "+ "present: " + linkedset.remove("Z"));
        System.out.println("Checking if A is present="+ linkedset.contains("A"));
        System.out.println("Updated LinkedHashSet: "+ linkedset);


    }
}
```

Step-6: Code for Exceptional Handling.

```java
package oopsconcepts;
public class ExceptionalHandling  {

    public static void main(String[] args) {
        int a[]= new int[7];
        try {
            a[10] = 100/0;
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Index out of range");
        }
        catch(ArithmeticException e) {
            System.out.println("Arithmetic Exception occured");
        }
        catch(Exception e) {
            System.out.println("Exception occured");
        }
        finally {
            System.out.println("This is the finally block!!");
        }
    }
}
```

Step-7: Code for Inheritance- Multi-level.

```java
package oopsconcepts;
    class Grandfather{
        void m1()
        {
            System.out.println("This is grandfather class!");
        }
    }
    class Father extends Grandfather{
        void m2()
        {
            System.out.println("This is father class!");
            m1();
        }
    }
    class Inheritance extends Father {
        void m3()
        {
            System.out.println("This is child class!");
            m2();
        }
        public static void main(String[] args) {
            Inheritance obj = new Inheritance();
            obj.m3();
        }
}
```

Step-8: Code for Access Modifiers.

```java
package oopsconcepts;
class Parent{
private void msgprivate(){// private method to class Parent
    System.out.println("Hello this is the private access modifier of class Parent!");
}
protected void msgprotected(){//protected member to Class Parent
    System.out.println("Hello this is the protected access modifier of class Parent!");
}
void msgdefault(){ //default method
    System.out.println("Hello this is the default access modifier of class Parent!");}
}
public class AccessModifiers extends Parent {
    private void printing(int x ) { // private method to class AcessModifiers i.e. the derived class
        System.out.println("The value of x in Private is : "+ x);
    }
    public static void main(String[] args) {
        AccessModifiers obj = new AccessModifiers();
        obj.printing(100);
        obj.msgprotected();
        obj.msgdefault();
    }

}
```

Step-9: Code for Interfaces.

InterfaceExample.java × OverloadingMethod.java

```java
package oopsconcepts;
interface Interface1{
    void m1();
}
interface Interface2{
    void m2();
}
public class InterfaceExample implements Interface1,Interface2 {
    public static void main(String[] args) {
        InterfaceExample obj = new InterfaceExample();
        obj.m1();
        obj.m2();
    }
    public void m1() {
        System.out.println("M1 CODE");
    }
    public void m2() {
        System.out.println("M2 CODE");
    }
}
```

Step-10: Code for Overloading Method.

OverloadingMethod.java ×

```java
package oopsconcepts;
public class OverloadingMethod {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OverloadingMethod obj = new OverloadingMethod();
        int sum = obj.add(2, 3);
        System.out.println("Addition of two int numbers is: "+ sum);
        int sum1 = obj.add(2, 3, 5);
        System.out.println("Addition of three int numbers is: "+ sum1);
        float sum2= obj.add(10f, 11f);
        System.out.println("Addition of two float numbers is: "+ sum2);
    }
    int add(int a, int b)
    {
        return(a+b);
    }
    int add(int a, int b, int c)
    {
        return(a+b+c);
    }
    float add(float a, float b)
    {
        return(a+b);
    }
}
```

Step-11: Output for Overloading Method.

```
<terminated> OverloadingMethod [Java Applica
Addition of two int numbers is: 5
Addition of three int numbers is: 10
Addition of two float numbers is: 21.0
```

Step-12: Output for Overriding Method.

```
History  Synchronize  Git Staging  Git Reflog  Properties  Co
<terminated> OverrideMethod [Java Application] C:\Users\ei13087\.p2\pc
This is the Grandfather Class!!
This is the Father Class!!
This is the Child Class!!
```

Step-13: Output for Interfaces.

```
<terminated> InterfaceExample (1) [Java Appl
M1 CODE
M2 CODE
```

Step-14: Output for Inheritance- Multi-level.

```
<terminated> Inheritance [Java Application] C
This is child class!
This is father class!
This is grandfather class!
```

Step-15: Output for Exceptional Handling.

```
<terminated> ExceptionalHandling [Java Application] C
rithmetic Exception occured
his is the finally block!!
```

Step-16: Output for Collections.

```
<terminated> CollectionsExamples [Java Application] C:\Users\ei13087\.p2
Array List: Krittika
Array List: 1
Array List: 33.22
Krittika
1
Krittika
1
2
Linked list: [Anil, Krittika, Harish, Agrawal, Ran]
[Harish]
Vector list: [1, 2, 3, 4, 5]
[1, 2, 3, 5]
1 2 3 5
HashSet: [READ, HI, ALL, For, To]
HashSet after removing elements: [READ, ALL, For, To]
Size of LinkedHashSet = 5
Original LinkedHashSet:[A, B, C, D, E]
Removing D from LinkedHashSet:true
Trying to Remove Z which is not present: false
Checking if A is present=true
Updated LinkedHashSet: [A, B, C, E]
```

Step-17: Output for Abstract Classes.

```
<terminated> AbstractExample (1) [Java
M1 CODE
M2 CODE
```

Step-18: Output for Access Modifiers.

```
History  Synchronize  Git Staging  Git Reflog  Propert
<terminated> AccessModifiers [Java Application] C:\Users\ei1308
The value of x in Private is : 100
Hello this is the protected access modifier of class Parent!
Hello this is the default access modifier of class Parent!
```