Q. What will happen if we are not providing name to dataprovider method?
Ans: If we are not providing name to data provider method then we will get exception in Test method because for Test method dataProvider name is required but we can overcome this probelm by using name of dataprovider method in Test method

Parameter of Test method and Return type of DataProvider

DataProvider method can have following types of return type

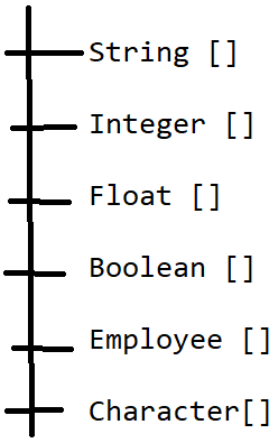
1) Object[] -> One dimentional

2) Object [][] -> Multi Dimential Array

3) Iterator<Object> -> For Collection type

4) Iterator<Object[]>

1) Object [] -> One dimentional Object Array



When we should go with Object [] and When we should go with particular type array?

Ans: If we know the data type of value then we can go with particular type array

If we are going to return only String type then use String [] , If we are going to return only Integer type value then use Integer []

How to define one dimentional String array and how to initialize one dimentional String array values?

String [] data=new String[size];

Example

String data []=new String[3];
data[0]="Abc";
data[1]="xyz";
data[2]="pqr";

```java
@DataProvider
public String[] loginData()
{
   String []data=new String[3];
   data[0]="Abc";
   data[1]="xyz";
   data[2]="pqr";
   return data;

}
```

Another Approach to define and initialize one dimentional Array

```java
String data[] =new String []{"Abc","Pqr","xyz"};

Or

String data[] =new String []{
                              "Abc",
                              "Pqr",
                              "xyz"};
```

```java
@DataProvider
public String [] loginData()
{
    String data[] =new String []{
                                  "Abc",
                                  "Pqr",
                                  "xyz"};
       .


}
```

How to use data in Test method which are suplied by DataProvider

```java
@Test
public void loginTest(String s)
{
    System.out.println(s);
}
```

When we should go with Object [] ?

Ans: If we don't know value type or we are passing mix kind of values to Test method then we should go with Object[]

```java
@DataProvider
public Object[] loginData()
{
   Object [] data=new Object[]{101,100.100,'C',"Abc"};
}
```

```java
@Test
public void loginTest(Object obj)
{
  System.out.println(obj);

}
```

2) Object [][] -> Multi Dimential Array

String[][]

Integer[][]

Float[][]

Double[][]

If we know type of value, then we should go with particular type array

If we don't know type of value or we are passing mixec type of values to Test method then we should go with Object array

How to define two dimentional/multi-dimentional array

```
String [][]data=new String[2][2];

data[0][0]="Abc";
data[0][1]="xyz";
data[1][0]="kkk";
data[1][1]="nnn"

or
String [][] data=new String[][]{
                    {"Admin","123"},
                    {"admin","Test"},
                    {"Test","test@123"}
                    };
```

```
@DataProvider
public String[][] loginData()
{
   String [][] data=new String[][]{
                    {"Admin","123"},
                    {"admin","Test"},
                    {"Test","test@123"}
                    };
        return data;

}
```

```
@Test
public void loginTest(String s1,String s2)
{
   System.out.println(s1+" ---"+s2);
}
```

Or

```
@Test
public void loginOrangeHrm(Object [] obj)
    {
            System.out.println(obj[0]+" "+obj[1]);
    }
```

If number of values passing to Test are not fixed then use below approach

```
@DataProvider()
public Object[][] loginData()
{
    Object [][] data=new Object[][] {
         {"abc",123,'B',100,100.90},
         {"xyz","Test"},
         {"admin",'C',500,true}
     };
     return data;
}
```

```
@Test
public void loginOrangeHrm(Object [] obj)
    {
            for (Object object : obj) {
                System.out.println(object);
            }
            System.out.println("==========");
    }
```

return type 3

Iterator<Object> -> For Collection type

String

Integer

Float

Double

```
@DataProvider
public Iterator<String> loginData()
{
   List<String> list=new ArrayList<>();
   list.add("abc");
   list.add("pqr");
  list.add("test");
  return list.iterator();
}
```

```
@Test
public void loginTest(String s)
{
   System.out.println(s);
}
```

If values are mix type in Collection

```
@DataProvider()
    public Iterator<Object> loginData()
    {
        List<Object> list=new ArrayList<>();
        list.add("abc");
        list.add("xyz");
        list.add(123);
        list.add(true);
         return list.iterator();
    }
```

```
@Test
public void loginOrangeHrm(Object obj)
{
        System.out.println(obj);
}
```

Type 4: Iterator<Object[]>

Iterator<Object[]>

```
@DataProvider()
    public Iterator<String[]> loginData()
    {
        List<String[]> list=new ArrayList<>();

        String arr2[]=new String[] {"mm","ggg","jj"};
        String arr3[]=new String[] {"pppp","aaa","qqqq"};

        list.add(new String[] {"123","xyz","KKK"});
        list.add(arr2);
        list.add(arr3);
        return list.iterator();
    }
```

```
@Test
public void loginOrangeHrm(String[] obj)
{
    for (String string : obj) {
            System.out.println(string);
        }
         System.out.println("====");
}
```

If we want to pass partial data set to Test method then use indices attribute in @DataProvider annotation

```java
@DataProvider(indices = {0,1})
    public Iterator<String[]> loginData()
    {
        List<String[]> list=new ArrayList<>();

        String arr2[]=new String[] {"mm","ggg","jj"};
        String arr3[]=new String[] {"pppp","aaa","qqqq"};

        list.add(new String[] {"123","xyz","KKK"});
        list.add(arr2);
        list.add(arr3);
        return list.iterator();
    }
```

```java
public void loginOrangeHrm(String[] obj)
    {
            for (String string : obj) {
                System.out.println(string);
            }
            System.out.println("====");
    }
```

While using indices concept there is no change required in Test method