Class shape

{

Void draw()

Syso(“Drawing shape”)

}

Class circle extends Shape

{

Void draw()

{

Syso(“Drawing circle”)

}

}

Public static void main

{

Shape s;

s= new Circle();

s.Draw()

}

Output would be: Drawing circle

Here, the child class method is called which was decided during the run time so it is known as Run time or dynamic polymorphism.

Same holds true for the following statement:

WebDriver driver= new FireFoxDriver()

Driver.findElement(“….”)

Here, driver is reference variable of WebDriver interface and referring to firefoxDriver class which implments WebDriver interface.

Here, findElement element of child class i.e firefoxdriver will be called.

That’s the main reason we always use WebDriver driver= new FireFoxDriver() because, in our script there may be chances that we may need to run our scripts on chrome browser. So when the reference variable is assigned as driver= new ChromeDriver() class then the ‘driver’ reference variable will start using the ‘findElement’ of ChromeDriver class becauase of run time or dynamic polymorphism. Because, in this concept, the child’s class overridden method (in above case is ‘findelement’) will always be executed. So, oveririden methods of both classes i.e firefox and chromdriver will be decided and executed during run time.

Interface WebDriver

{

Void findElements()

}

Class firefox implements WebDriver

{

Void findElements()

{

Void findElements()

{

Syso(“Findelement method of firefox class”)

}

}

Class chromeDriver implements WebDriver

{

Void findElements()

{

Void findElements()

{

Syso(“Findelement method of chrome class”)

}

}

Public static void main

{

WebDriver driver;

Driver= new FireFoxDriver()

Driver.findElement()

Driver= new ChromeDriver()

Driver.findElement()

}

In the above example, findElement method will be called depending upon which the reference variable is pointing to and due to run time polymorphism.