METHOD OVERLOADING AND OVERRIDING

Let's see first what method overloading and overriding are.

1. **Overloading** happens when 2 methods in the same class have the same name but different parameters.
2. **Overriding** means having 2 methods with the same name and same parameters, one being in a parent class and the other in a child class that inherits from the parent class.

See below an example for each.

Method overloading example

You want to have your own Listbox class to interact with dropdown lists.

The Listbox class is just a wrapper around the Select class.

You want it to have simpler method names for selecting list options:

```
1. public class Listbox {
2.
3.   Select list;
4.
5.   public Listbox(Select list) {
6.     this.list = list;
7.   }
8.
9.   public void select(int i) {
10.           this.list.selectByIndex(i);
11.         }
12.
13.         public void select(String text) {
14.           this.list.selectByVisibleText(text);
15.         }
16.
17.         public void deSelect(int i) {
18.           this.list.deselectByIndex(i);
19.         }
20.
21.         public void deSelect(String text) {
22.           this.list.deselectByVisibleText(text);
23.         }
24.
25.         //other methods
26.
27.       }
```

The class has 2 select methods, one with an integer parameter, the other with a string parameter.

The 2 methods have the same name and both have no return type.

This is an example of overloading.

The difference between the 2 methods is being made by the parameter type.

The same applies for the deSelect() methods.

Method overriding example

Method overriding happens when you have child classes inheriting from parent classes.

For example, you may have a base page class with methods that work for any page:

```
1. public abstract class BasePage {
2.
3.   String name;
4.   String url;
5.   String title;
6.   WebDriver driver;
7.
8.   public BasePage(String name, String title, String url, WebDriver drv) {
9.     this.name = name;
10.            this.title = title;
11.            this.url = url;
12.            this.driver = driver;
13.          }
14.
15.          protected String title() {
16.            return this.title;
17.          }
18.
19.          protected String url() {
20.            return this.url;
21.          }
22.
23.          protected String name() {
24.            return this.name;
25.          }
26.
27.          protected WebDriver driver() {
28.            return this.driver;
29.          }
30.
31.          protected boolean isDisplayed() {
32.            return this.title.equalsIgnoreCase(this.driver.getTitle()) &&
33.                   this.url.equalsIgnoreCase(this.driver.getCurrentUrl());
34.          }
35.        }
```

isDisplayed() method works by checking that the title and url of the page are equal with the title and url of the current page.

HomePage class inherits from BasePage class:

```
1. public class HomePage extends BasePage {
2.
3.   public HomePage(WebDriver driver) {
4.     super("HomePage",
5.           "Home Page Title",
6.           "http://www.abcdef.com/",
7.           driver);
8.   }
9.
10.          //other methods
11.
12.       }
```

When creating a HomePage object, isDisplayed() is the method from BasePage.java:

```
1. HomePage homePage = new HomePage(driver);
2. assertTrue(homePage.isDisplayed());
```

SlowPage class is used for pages that load slowly.

We need for these pages a different way of checking if they are displayed:

```
1. public class SlowPage extends BasePage {
2.
3.   public SlowPage(WebDriver driver) {
4.     super("SlowPage",
5.           "Slow Page Title",
6.           "http://somepage.com",
7.           driver);
8.   }
9.
10.          public boolean isDisplayed() {
11.            WebDriverWait wait = new WebDriverWait(super.driver(), 30);
12.
13.            return wait.until(ExpectedConditions.urlTobe(super.url())) &&
14.                   wait.until(ExpectedConditions.titleIs(super.title()));
15.          }
16.
17.          //other methods
18.
19.       }
```

By using an explicit wait and expected conditions, isDisplayed() method waits until the page title and url are correct.

When creating a SlowPage object, isDisplayed() is the method from SlowPage class and not from BasePage class:

```
1. SlowPage slowPage = new SlowPage(driver);
2. assertTrue(slowPage.isDisplayed());
```