==========

Topic: **OOPS Concepts**

==========

1. Document prepared by: **Rajat Verma**
    a. https://www.linkedin.com/in/rajat-v-3b0685128/
    b. https://github.com/rajatt95
    c. https://rajatt95.github.io/

========== Testing - OOPS Concepts ==========

---

1. Object-Oriented -

1. OOPS (**Object-Oriented Programming System**)
    a. It is a methodology that we use for software development and testing using some techniques.
2. **Java is an Object-Oriented, Platform Independent Programming language.**

—----------

The basic concept of OOPs:

1. Inheritance
2. Polymorphism
3. Encapsulation
4. Abstraction

Examples:

Object	Flight
Data member/Properties/Variable	Date, number, origin, destination
Tasks/Methods/Behaviour/Functionality	bookFlight(), getPNR()
Abstraction	Show - Download Android app (.APK file), Windows software (.exe file) Hidden - Package, Methods
Method Overloading	add(), add(x,y) and add(x,y,z)
Method Overriding	drawPolygon() -> square, Rectangle, Triangle

---

Object:

1. Instance of class
2. Real-world entity
3. **The entity that has**
    a. **State and Behaviour**
    b. Properties and Functionality
4. It helps to make communication between classes(by calling methods)
5. Objects help to invoke the methods present in different classes.
6. Example:
    a. Object: **Human**
        i. Properties/State: Name, Color, Height
        ii. Behaviour/Functionality: work(), run(), read(), write()
    b. Object: **Student**
        i. Properties/State: Name, Roll no., DOB
        ii. Behaviour/Functionality: read(), write(), play()
    c. Object: **Software Engineer**
        i. Properties/State: Name, Skills, Specialization
        ii. Behaviour/Functionality: writeCode(), runCode(), debugCode()

Class:

1. The blueprint that an object follows
2. The class has
    a. Properties/Data members/Variables
    b. Tasks/Methods/Functionalities/Behaviour

Constructor:

1. Special method
2. Same name as class name
3. No return type
4. Executes when an object is created
5. Types:
    a. Default Constructor
    b. Parameterized Constructor

Abstraction:

1. Selective Ignorance
2. Show only essential parts; hide the implementation details
3. ***Hiding internal details and showing functionality is known as abstraction.***
4. Example:
    a. Android applications - .apk file
    b. Windows-based OS Software- .exe file
    c. Selenium -
        i. Locators:
            1. id, name, xpath, linkText, tagName
    d. Automation framework:
        i. WebDriver driver = new ChromeDriver();
            1. WebDriver - Interface
            2. ChromeDriver - Class
            3. new - keyword
            4. driver - reference name
        ii. 100% abstraction -> Interface
        iii. Partial abstraction -> Abstract classes

Encapsulation:

1. Binding variables and methods under a single Entity.
2. Binding (or wrapping) code and data together into a single unit are known as encapsulation
3. **Automation framework:**
    a. Design Pattern - Page Object Model
        i. Variables -private
        ii. Methods - public

```
@FindBy(id = "login1")
private WebElement emailTextBox;

public WebElement emailTextBox() {
    return emailTextBox;
}
```

1.

Inheritance:

1. Acquiring the properties of one class to another class
2. When one object acquires all the properties and behaviors of a parent object,
3. Code re-usability
4. Parent and Child | Super and Sub | Base and Derived
5. Types:
    a. Single, Multi-Level, Hierarchical

6. **Automation framework:**
    a. Single level: **TestScripts extends TestBase**
    b. Multi-level: TestScripts extends CommonUtilities, CommonUtilities extends TestBase

---

Polymorphism:

1. Many + Forms
2. Perform same task in different ways
3. Types:
    a. Compile-time Polymorphism
        i. Method Overloading
            1. Signature
                a. Number of Arguments
                b. Order of Arguments
                c. Type of Arguments
        ii. Multiple methods with the same name, but Different in Arguments/Parameters
        iii. Example:
            1. Assert.**assertEquals**(String actual, String expected)
            2. Assert.**assertEquals**(int actual, int expected)
            3. Assert.**assertEquals**(double actual, double expected)
            4. add(), add(x,y) and add(x,y,z)
    b. Run time Polymorphism
        i. Method Overriding
            1. A process where the method in the child class has the same name and the same parameters as that of the method in its base class.
        ii. Example:
            1. drawPolygon()
                a. Square
                b. Rectangle
                c. Triangle
            2. Association, Composition
            3. Selenium -
                a. get(), navigate()

==========

1. **To connect:**
    a. https://www.linkedin.com/in/rajat-v-3b0685128/
    b. https://github.com/rajatt95
    c. https://rajatt95.github.io/

==========