estNG is a testing framework inspired by JUnit and NUnit, designed for test configuration and management in Java applications. It provides a range of features that make it useful for unit testing, integration testing, and end-to-end testing, including:

1. **Annotations**: It uses annotations to define test methods and configuration, making the code more readable and manageable.
2. **Test Configuration**: You can set up configurations for test methods, groups, and classes, allowing for flexible test execution.
3. **Parallel Execution**: TestNG supports running tests in parallel, which can significantly reduce the time required for test execution.
4. **Data-driven Testing**: It allows for parameterized tests, enabling the same test method to run with different sets of data.
5. **Reporting**: TestNG generates detailed reports and logs, making it easier to analyze test results.
6. **Integration**: It integrates well with build tools like Maven and Gradle, as well as continuous integration systems like Jenkins.

When creating a new Java class, you typically have a few options for generating stubs or skeletons, depending on your IDE. Here are some common options you might consider:

1. **Basic Class Stub**: This includes the class declaration and a default constructor. It's a straightforward starting point.
2. **Fields and Getters/Setters**: If you want to define some properties for your class, you can create stubs for fields along with their corresponding getter and setter methods.
3. **Method Stubs**: If you have specific methods in mind, you can generate stubs for those methods with empty implementations.
4. **Override Methods**: If your class is extending another class or implementing an interface, you can generate stubs for the overridden methods automatically.
5. **JUnit Test Stub**: If you’re writing a class that needs unit tests, some IDEs allow you to create a test class with stubs for common test methods.
6. **import** org.openqa.selenium.WebDriver;
7. **import** org.openqa.selenium.chrome.ChromeDriver;
8. **public** **class** Selintroduction {
9. **public** **static** **void** main(String[] args) {


13. //Invoking Browser
14. //Chrome- Chrome Driver ->Methods close get
15. //firefox-FirefoxDriver ->methods close get
16. // Webdriver close get
17. // Webdriver methods + class methods
18. WebDriver driver= **new** ChromeDriver();

21. Why web driver not chrome driver
23. }
24. }
25. // chromedriver.exe-> Chrome browser(reason brower cannot be directly executed)( CHROME DRIVER FILE RESPONSIBLE FOR INVOKING THE CHROME BROWSER)
26. // step to invoke chrome browser//
27. // Selenium Manager
28. WebDriver driver= **new** ChromeDriver();
30. SECOND WAY(DIRECTLY GIVING THE LOCAL ADDRESS)
31. // chromedriver.exe-> Chrome browser
32. // step to invoke chrome browser//
33. System.setProperty("webdriver.chrome.driver", "path downloaded from web")
35. WebDriver driver= **new** ChromeDriver();