Hands-on Questions for Unit Testing Using TestNG

1. Data Types

- Write a TestNG test to validate default values of primitive data types.
- Test a method that converts primitive types to their wrapper class equivalents.
- Create a test case for verifying type casting between data types.

2. Operators

- Write a test to validate the behavior of arithmetic operators in a method.
- Create a unit test for logical operators used in conditional statements.
- Test a method implementing bitwise operators to ensure correct functionality.

3. Classes and Objects

- Write a TestNG test to verify object creation and initialization using constructors.
- Test a method that compares objects using the equals() method.
- Create a test case to validate the behavior of instance variables and methods.

4. Overloading

- Write unit tests for overloaded methods to ensure correct parameter handling.
- Create a test to validate the output of overloaded constructors.
- Test a method that uses overloaded versions for different operations.

5. Arrays

- Create a TestNG test to validate array initialization and boundary conditions.
- Write a test for methods that perform operations like sorting or searching in arrays.
- Test a method that merges two arrays and verify the resulting array content.

6. String and Wrapper Classes

- Write a test for methods manipulating strings, such as concatenation or substring extraction.
- Create tests to validate Integer.parseInt() and related wrapper class methods for type conversion.

- Test a method that uses String methods like replace() or split().

7. Inheritance

- Write a TestNG test to validate method overriding in a derived class.
- Test constructor chaining in a class hierarchy using inheritance.
- Create a test case for accessing parent class methods and variables from a child class.

8. Overriding

- Validate the behavior of overridden methods using TestNG assertions.
- Create a test case to ensure the superclass method is invoked using super.
- Test a method that uses polymorphism with overridden methods.

9. Object Class

- Write a test to validate the implementation of toString() in a custom class.
- Create a test for hashCode() and equals() to ensure consistent behavior.
- Test the clone() method for object duplication in a class.

10. Interface

- Write a TestNG test to validate the implementation of an interface.
- Test a method that uses multiple interfaces implemented by a class.
- Create a test case for default and static methods in an interface.

11. Exceptions

- Create a test case to handle checked and unchecked exceptions in a method.
- Write a TestNG test to validate exception hierarchy using try-catch blocks.
- Test a method that throws custom exceptions and verify the exception details.

12. Static

- Write a TestNG test to validate static variable and static method behavior.
- Test a method that uses static blocks for initialization.
- Create a test case for singleton classes relying on static variables.

13. Access Specifiers

- Write unit tests to verify accessibility of public, private, protected, and default methods.
- Test a method that invokes protected or default methods from another package.
- Create a test case for verifying encapsulation using private fields and getter/setter methods.

14. Collections

- Write a TestNG test to verify functionality of common collections like ArrayList, HashMap, and HashSet.
- Test operations such as adding, removing, and updating elements in a collection.
- Create a test case to validate sorting and searching within a collection, such as TreeMap or TreeSet.
- Write a test for iterating through a collection using various methods like for-each loop, Iterator, or Stream.
- Test a method that uses generics with collections to ensure type safety.