**Test interfaces and default methods in Junit**

On interface default methods, JUnit allows you to specify *@Test, @RepeatedTest, @ParameterizedTest, @TestFactory, @TestTemplate, @BeforeEach,*and*@AfterEach*.

If the test interface or test class is annotated with *@TestInstance(Lifecycle.PER CLASS), @BeforeAll*and*@AfterAll* can be declared on static methods in a test interface or on interface default methods.

On a test interface, *@ExtendWith* and *@Tag* can be defined so that classes that implement the interface inherit the tags and extensions automatically.

Finally, you can implement these test interfaces in your test class to have them applied. Let us create a demo example to test an interface and default method using JUnit.

interface TestLifecycleLogger {

  static final Logger logger = Logger.getLogger(TestLifecycleLogger.class.getName());

  @BeforeAll

  default void beforeAllTests() {

      logger.info("Before all tests");

  }

  @AfterAll

  default void afterAllTests() {

      logger.info("After all tests");

  }

  @BeforeEach

  default void beforeEachTest(TestInfo testInfo) {

      logger.info(() -> String.format("About to execute [%s]",

          testInfo.getDisplayName()));

  }

  @AfterEach

  default void afterEachTest(TestInfo testInfo) {

      logger.info(() -> String.format("Finished executing [%s]",

          testInfo.getDisplayName()));

  }

}

interface TestInterfaceDynamicTestsDemo {

  @TestFactory

  default Stream<DynamicTest> dynamicTestsForPalindromes() {

      return Stream.of("racecar", "radar", "mom", "dad")

          .map(text -> dynamicTest(text, () -> assertTrue(isPalindrome(text))));

  }

}