Verificação e Validação de Software I

Introdução ao Teste Unitário

Roteiro 01: Introdução ao uso do JUnit

O objetivo deste roteiro é familiarizar o estudante com o uso do JUnit no contexto do ambiente de trabalho proposto para este semestre.

Os projetos que serão criados na IDE "Visual Studio Code" e gerenciados usando o software "Maven". Siga este roteiro e faça os exercícios propostos a fim de dominar o uso das ferramentas. Esclareça suas dúvidas com o professor.

Parte 1: criando seu primeiro projeto

Para criar seu primeiro projeto siga os passos que seguem:

1) Abra o VSCode.
2) No menu principal selecione "View|Command Pallet" e em seguida "Java: Create Java Project".
3) Selecione o tipo de projeto a ser criado: "Maven".
4) Selecione "No Archetype".
5) Responda as questões como segue:
    a. Indique o identificador da sua organização: exemplo -> com.vev
    b. Indique o nome do artefato que será construído: exemplo -> exer01
    c. Indique a pasta onde o projeto deverá ser criado
6) O Maven irá criar então uma pasta com a estrutura do projeto no diretório indicado no passo 5.
7) Ajuste o arquivo "pom" conforme as indicações do "sample oficial" do JUnit para Maven. O arquivo deverá ser semelhante ao exemplo que segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.vev</groupId>
    <artifactId>exer01</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

       <dependencyManagement>
              <dependencies>
                     <dependency>
                            <groupId>org.junit</groupId>
                            <artifactId>junit-bom</artifactId>
                            <version>5.10.2</version>
                            <type>pom</type>
                            <scope>import</scope>
                     </dependency>
```

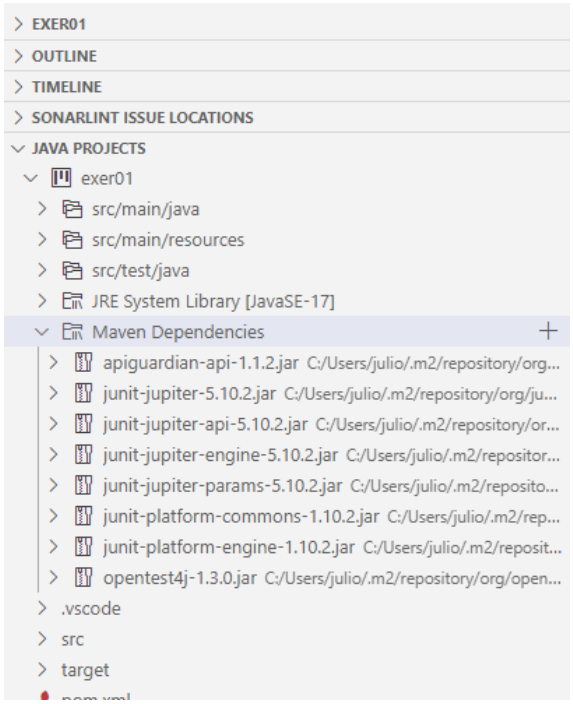
```
		</dependencies>
	</dependencyManagement>

	<dependencies>
		<dependency>
			<groupId>org.junit.jupiter</groupId>
			<artifactId>junit-jupiter</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<artifactId>maven-surefire-plugin</artifactId>
				<version>3.2.5</version>
			</plugin>
		</plugins>
	</build>

</project>
```

8) Antes de prosseguir, garanta que o projeto foi “atualizado/ressincronizado” com as alterações presente no arquivo “pom.xml” para que as dependências sejam “baixadas” para o diretório local.



9) Crie o arquivo “NumFinder.java” na pasta “src/main/java/...” (“...” significa os subdiretórios de acordo com o nome do pacote configurado no projeto, neste exemplo seria “/com/vev/) e copie o conteúdo que segue para ele:

```
public class NumFinder {
    private int smallest = Integer.MAX_VALUE;
    private int largest = Integer.MIN_VALUE;

    public void find(int[] nums) {
        for(int n : nums) {
            if(n < smallest)smallest = n;
            else if (n > largest) largest = n;
        }
```

```
    }

    public int getSmallest () { return smallest; }

    public int getLargest () { return largest; }
}
```

10) Crie o arquivo “NumFinderTest.java” na pasta “src/test/java” e copie o conteúdo que segue para ele:

```
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import com.vev.NumFinder;

class NumFinderTest {
          private NumFinder nf = null;

          @BeforeEach
          void setUp() {
              nf = new NumFinder();
          }

          @Test
          void testeMaiorQQOrdem() {
              nf.find(new int[] { 10, 8, 2, 14, 7 });
              int maior = nf.getLargest();
              assertEquals(14, maior);
          }

          @Test
          void testeMenorQQOrdem() {
              nf.find(new int[] { 10, 8, 2, 14, 7 });
              int menor = nf.getSmallest();
              assertEquals(2, menor);
          }

          @Test
          void testeMaiorOrdemCrescente() {
              nf.find(new int[] { 1, 2, 3, 4, 5 });
              int maior = nf.getLargest();
              assertEquals(5, maior);
          }

          @Test
          void testeMenorOrdemCrescente() {
              nf.find(new int[] { 1, 2, 3, 4, 5 });
              int menor = nf.getSmallest();
              assertEquals(1, menor);
          }

}
```

11) Abra um terminal na pasta raiz do projeto (aquela que tem o arquivo “pom.xml”).
12) Execute seus casos de teste com o comando “mvn test” e observe os resultados.
13) Altere um dos “asserts” de maneira que o teste vá falhar.
14) Execute novamente o conjunto de casos de teste.
15) Após os testes falharem, localize o relatório de falhas na pasta “target\surfire-reports”. Analise o relatório.