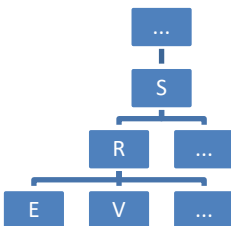


**Obiective**

- Testarea aplicațiilor la diferite niveluri de testare, e.g., testare unitară, testare de integrare.
- Utilizarea tool-urilor folosite în procesul de testare (TestLink, Jenkins, JUnit, Maven, Git, Mockito).

**Cerințe**

Să se realizeze următoarele task-uri:

Task, puncte	Descriere task
	Identificați la nivelul arhitecturii clase particulare, codificate aici prin E, R și S care împlinesc următoarele condiții: <i>E reprezintă o entitate din domeniul problemei. R este un repository cu elemente de tip E. S permite gestionarea repository-ului.</i>
[Unit Testing. JUnit. Mockito] 2 puncte	<p>Realizați testarea în izolare (unit testing) pentru clasele E, R și S, folosind framework-urile <b>JUnit</b> (vezi <a href="#">Tutorial JUnit</a> în Lab02) și <b>Mockito</b> (vezi <a href="#">Tutorial Mockito</a>). Pentru R și S se vor descrie teste care folosesc <b>mock</b> sau <b>spy</b>, a alegere. Pentru fiecare clasă testată se va descrie câte o clasă de test separată cu <b>minimum 2 teste</b>.</p> <p><b>Nu se cer următoarele:</b></p> <ul style="list-style-type: none"> <li>• elaborarea de fișiere similare cu <a href="#">Lab02_BBT_TCs_Form.xls</a> și <a href="#">Lab03_WBT_TCs_Form.xls</a>;</li> <li>• crearea <i>suplimentară</i> în TestLink a suite-lor de teste pentru testarea unitară a modulelor E, R și S.</li> </ul>
[Integration Testing. Mockito] 2 puncte	<p>Considerăm în aplicația dezvoltată existența următoarei diagrame de dependență între module, unde E, R și S corespund claselor deja testate în izolare.</p>  <pre> graph TD     S[S] --- R[R]     S --- D1[...]     R --- E[E]     R --- V[V]     R --- D2[...]     D1 --- V     D1 --- D3[...]   </pre> <p>Realizați testarea de integrare utilizând strategia de <b>integrare incrementală top-down</b>. Se vor evidenția următoarele clase cu teste:</p> <p><b>[task-ul anterior] Step 1.</b> testare unitară pentru E, R și S;  <b>Step 2.</b> integrare R (se testează S cu R; pentru E se folosesc obiecte <i>mock</i>);  <b>Step 3.</b> integrare E (se testează S + R cu E);</p> <p>Pentru <b>Step 2.</b> și <b>Step 3.</b> fiecare clasă de test va avea <b>minimum 2 teste</b>.</p>
[TestLink] 2 puncte	<p>În cadrul proiectului <b>PrjAAA</b>, corespunzător userului <b>xyir1234</b> utilizat anterior pentru <a href="#">Lab03</a>, se vor realiza următoarele task-uri:</p> <ol style="list-style-type: none"> <li>1.1. definiți planul de testare <b>xyir1234_IntT_TP</b> în cadrul proiectului <b>PrjAAA</b> (secțiunea <i>Test Plan</i>);</li> <li>1.2. creați suita de teste <b>xyir1234_IntT</b> care va conține <b>3 cazuri de testare</b>, câte un caz de testare pentru fiecare pas de aplicare a strategiei de integrare <b>Top down</b> (secțiunea <i>Test Specification</i>);</li> <li>1.3. asociați cazurile de testare create la planul <b>xyir1234_IntT_TP</b>;</li> <li>1.4. asociați cazurile de testare create la cerințele create anterior, după caz, la <b>xyir1234_F01</b> sau <b>xyir1234_F02</b>;</li> <li>1.5. generați documentația aferentă din (secțiunea <i>Test Specification</i>, opțiunea <i>Generate Test Specification Document</i>) în format .docx.</li> </ol>
[Jenkins] 2 puncte	<p>Se va crea, configura și executa <b>câte un job</b> (vezi <a href="#">Tutorial Jenkins</a>) pentru fiecare plan de testare creat anterior, i.e., <b>xyir1234_BBT_TP</b>, <b>xyir1234_WBT_TP</b>, <b>xyir1234_IntT_TP</b>. Job-urile create vor fi denumite <b>xyir1234Job_BBT</b>, <b>xyir1234Job_WBT</b> și <b>xyir1234Job_IntT</b>.</p> <p><b>Observație:</b></p> <ul style="list-style-type: none"> <li>• pentru fiecare test creat în TestLink se va verifica (1) dacă este un test executat <i>automat</i> și (2) dacă are setat corect <i>numele clasei de test</i> și <i>numele metodei de test</i> (vezi <a href="#">Tutorial TestLink</a>, pagina 14).</li> </ul>
[TestLink+ Jenkins] 1 punct	Vizualizarea rezultatului execuției fiecărui job Jenkins în TestLink (vezi <a href="#">Tutorial Jenkins</a> ). Pentru fiecare test case executat este necesar să apară statusul <b>passed</b> .
[Git] 1 punct	<p>Se va actualiza conținutul repository-ului <b>Git</b> cu documentele elaborate în cadrul acestei teme:</p> <ul style="list-style-type: none"> <li>• în folderul <b>Docs/Lab04</b> fișierul cu documentația generată în TestLink;</li> <li>• pachetul/ele cu teste implementate în Java pentru <a href="#">Lab04</a>;</li> <li>• dacă este cazul, codul sursă modificat în urma depanării.</li> </ul>

Timp de lucru recomandat pentru rezolvarea temei de laborator ~ 4 ore/echipă.

#### Predarea temei de laborator

##### [Unit Testing. JUnit. Mockito]

- implementarea claselor cu teste pentru clasele E, R și S.

##### [Integration Testing. Mockito]

- implementarea claselor cu teste folosind strategia de integrare **Top Down**.

##### [TestLink+Jenkins]

- Câte un job pentru fiecare dintre cele trei planuri de testare create în TestLink: **xyir1234Job\_BBT**, **xyir1234Job\_WBT** și **xyir1234Job\_IntT**.
- Fiecare test din TestLink are statusul **passed**.

#### Termene de predare

Săptămâna	Tema de laborator			Primul termen de predare	Ultimul termen de predare
S07	L04.	Niveluri de testare	JUnit, TestLink, Jenkins, Git, Mockito	S09	S11*°
S08				S10	S12*°

\*) Temele restante se vor putea preda în limita timpului disponibil.

°) Se pot preda cel mult două teme de laborator.