<table>
<tr><th>Durata task
[Puncte]</th><th>Descriere task</th></tr>
<tr><td>2h</td><td>• Se va verifica modul de realizare a task-urilor primite în cadrului temei de laborator precedente, i.e., Lab 03. Testare Black-Box.</td></tr>
<tr><td>2h</td><td>• Se va exemplifica proiectarea cazurilor de testare folosind criteriul white-box (vezi MaxScoreQuizCounter_TCs_WBT.xlsx).
• Se va discuta proiectarea cazurilor de testare folosind criteriul white-box pentru metoda <code>cautaCarte</code> din proiectul Biblioteca, clasa CartiRepoMockB.

<pre>
/**
 * Construieste lista cartilor scrise de un autor dat
 * @param autor - autorul pentru care se cauta cartile scrise
 * @return lista cartilor scrise de autorul dat
 */
@Override
public List<Carte> cautaCarte(String autor) {
   List<Carte> carti = getCarti();
   List<Carte> cartiGasite = new ArrayList<Carte>();
   int i=0;
   while (i<carti.size()){
      boolean exista = false;
      List<String> lautor = carti.get(i).getCuvinteCheie();
      int j = 0;
      exista = hasAutor(lautor, autor);
      if(exista == true){
         cartiGasite.add(carti.get(i));
      }
      i++;
   }
   return cartiGasite;
}

private boolean hasAutor(List<String> autorList, String a) {
   boolean exista = false;
   for(String autor: autorList){
      if(autor.toLowerCase().contains(a.toLowerCase())){
         exista = true;
         break;
      }
   }
   return exista;
}
</pre>

• [Se va exersa aplicarea WBT (vezi TestWBT.ppsx)].</td></tr>
<tr><td>TEMĂ
[10 puncte]</td><td>Se consideră următoarea variantă* a metodei <code>modifyEmployeeFunction</code> din proiectul Angajaţi, clasa EmployeeMock.

<pre>
/**
 * Modifica atributul 'functia didactica' pentru un angajat dat
 * @param employee - angajatul eptnru care se modifica
atributul 'functia didactica'
 * @param newFunction - noua functie didactica (ASISTENT,
LECTOR, CONFERENTIAR, PROFESOR)
 */
@Override
public void modifyEmployeeFunction(Employee employee,
DidacticFunction newFunction) {

   if (employee!=null) {
</pre></td></tr>
</table>

```
        int i = 0;
        while (i < employeeList.size()) {
            if (employeeList.get(i).getId() == employee.getId())
                employeeList.get(i).setFunction(newFunction);
            i++;
        }
    }
}
```

) Dacă este necesar, se copiază codul de mai sus în proiectul **Angajaţi.*

Realizaţi următoarele task-uri pentru metoda precizată:

1. **[5 puncte] Se vor elabora:**
   - Control Flow Graph (CFG) **[1 punct]**;
   - calculul complexităţii ciclomatice (CC, 3 formule de calcul) **[1 punct]**;
   - drumurile independente **[1 punct]**;

2. **[2 puncte] Se vor proiecta cazuri de testare** pentru acoperirea criteriilor:
   1. acoperirea instrucţiunilor (statement coverage, **sc**);
   2. acoperirea deciziilor/condiţiilor (decision/condition coverage; **dc**, **cc**, **dcc**);
   3. acoperirea condiţiilor multiple (multiple condition coverage, **mcc**);
   4. acoperirea drumurilor (all path coverage, **apc**);
   5. acoperirea ciclurilor simple / imbricate (simple and nested loop coverage, **lc**).

3. **[3 puncte] Se vor implementa toate cazurile de testare** proiectate anterior, folosind JUnit.

4. **[2 puncte] Se va determina gradul/procentul de acoperire cu teste rulate pentru metoda testată**, i.e., metoda pentru care s-a elaborat CFG (vezi **Tutorial Coverage**). Valoarea obţinută se va completa în fişierul **Lab04_WBT_TCs_Form.xlsx** în secţiunea **Statistics**.

Pentru cerinţele 1 --> 4 se va folosi fişierul **Lab04_WBT_TCs_Form.xlsx**.

Observaţie

- În situaţia în care cazurile de testare proiectate şi implementate evidenţiază defecte (rezultatul aşteptat nu este identic cu rezultatul obţinut în urma rulării testelor), codul sursă se va depana şi procesul de testare se va relua pentru toate cazurile de testare. În situaţia modificării codului sursă, se reiau TOATE task-urile de testare.