**Enuntul problemei:**

**Problema 15**

**Cerințe:**

Optimizați aplicația client/server implementată la ***tema 2*** astfel încât să utilizați șablonul de proiectare comportamental ***Observer*** în dezvoltarea aplicației client. În plus, interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională (implicit limba română).

 În ***faza de analiză*** realizați diagrama cazurilor de utilizare și diagramele de activități corespunzătoare tuturor cazurilor de utilizare.

 În ***faza de proiectare*** realizați:

 diagrama de clase corespunzătoare aplicației ***server***;

 diagrama de clase corespunzătoare aplicației ***client***;

 diagrame de secvență corespunzătoare tuturor cazurilor de utilizare.

 În ***faza de implementare*** scrieți cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

 Finalizarea temei va consta în predarea unui director ce va cuprinde:

 Un fișier word care cuprinde numele studentului, grupa, enunțul problemei, instrumentele utilizate și detalii despre modul în care a fost utilizat șablonul de proiectare ***Observer***.

 Un fișier cu diagramele UML realizate;

 Directorul cu aplicația implementată.

**Observație:**

Pentru persistență se va utiliza o bază de date relațională (MySQL, SQL Server, Oracle, etc.). Se va utiliza baza de date doar în aplicațiile unde este necesară persistența.

**Limbaj de programare:** Java

**Mediu de dezvoltare:** Eclipse 2019-09, Java 8, JDK 12.0.2

**Librarii folosite:** Json Simple pentru salvarea fisierului JSON, JAXB pentru salvarea fisierului XML, JFreeChart pentru a vizualiza statisticile, Swing pentru interfata grafica, MySQL Connector pentru a face conexiunea cu baza de date MySQL, Java RMI pentru transferul de date(obiecte) intre aplicatia client si server, Java SocketServer pentru server si Java Socket pentru client, pentru schimbarea setarilor de limba am folosit clasa ResourceBundle

**Observer design pattern:** am creat doua interfete: interfata Observer, care are o singura metoda (update) si este implementata de fiecare view ( GeneralView, AngajatView, AdministratorView, CalatorView si LogInPanel); interfata Observable, care are trei metode ( notifyObservers, attach, detach) si este implemetata de clasa RBundle care are ca si functionalitate schimbarea setarilor de limba.

**Baza de date:** MySQL 8.0

**Credentiale pentru un angajat:**

Username: third

Password: pass3

**Credentiale pentru un administrator:**

Username: first

Password: pass1