Raportul proiectului

Descrierea proiectului:

- Acest proiect este practic o simulare a birocratiei!
- Clientii trebuie sa obtina documente specifice (cum ar fi un pasaport sau permis de conducere).
- Documentele au dependente (de exemplu, ai nevoie de carte de identitate, cazier judiciar si certificat fiscal inainte de a putea obtine un pasaport).
- Institutiile emit documente si au mai multe ghisee.
- Ghiseele servesc clientii in ordine si au statusuri diferite (DISPONIBIL, IN PAUZA, INCHIS).
- Simularea foloseste multi-threading pentru a gestiona toate cererile clientilor si operatiunile institutiilor in acelasi timp. Este vorba despre demonstrarea sincronizarii thread-urilor, rezolvarii dependentelor si procesarii bazate pe grafuri.

Ce face aplicatia:

- Creeaza clienti care doresc diverse documente.
- Fiecare client primeste un document tinta (cum ar fi un pasaport).
- Sistemul calculeaza automat "calea de dependenta" – ce documente sunt necesare si in ce ordine.
- Clientii merg la institutiile potrivite, in ordinea corecta.
- La fiecare institutie, clientii asteapta la coada pana cand un ghiseu liber ii serveste.
- Ghiseele proceseaza clientii (simuland timpul real de lucru).
- Dupa ce obtin un document, clientii merg la urmatoarea institutie.
- Acest lucru continua pana cand toata lumea obtine documentul dorit!

Probleme de concurenta abordate:

- **Race Condition la accesarea documentelor:**
  - **Problema:** Mai multe thread-uri incercau sa modifice lista de documente in acelasi timp.
  - **Solutie:** Am folosit `ConcurrentHashMap.newKeySet()` si o

metoda `synchronized receiveDocument()`.

- **Deadlock la asteptarea documentelor:**
  - **Problema:** Un client putea ramane blocat daca o notificare era pierduta.
  - **Solutie:** Am folosit un pattern `wait/notify` cu o verificare intr-o bucla `while`.
- **Thread-safety pentru coada de clienti:**
  - **Problema:** Mai multe ghisee incercau sa acceseze aceeasi coada in acelasi timp.
  - **Solutie:** Am folosit `BlockingQueue<Client>` (cum ar fi `LinkedBlockingQueue`) pentru operatii thread-safe.
- **Probleme de vizibilitate cu statusul ghiseului:**
  - **Problema:** Alte thread-uri nu vedeau actualizarile de status imediat.
  - **Solutie:** Am folosit variabile `volatile` pentru status si flag-ul de rulare.
- **Race Condition la generarea ID-urilor:**
  - **Problema:** ID-urile erau duplicate atunci cand erau create simultan.
  - **Solutie:** Am folosit `AtomicInteger` pentru generarea atomica de ID-uri.
- **Gestionarea intreruperilor:**
  - **Problema:** Thread-urile nu se opreau corect atunci cand programul se inchidea.
  - **Solutie:** Am verificat flag-ul `running` si am gestionat corect `InterruptedException`.
- **Coordonarea inchiderii programului:**
  - **Problema:** Thread-urile nu se terminau intr-o maniera ordonata.
  - **Solutie:** Am folosit `join()` pentru clienti, apoi `shutdown()` si `interrupt()` pentru ghisee, cu un timeout.

Implicarea echipei:

- Constantin Cătălina-Viviana: 100%
- Hărdălău Elvis-Costinel: 100%
- Iuhasz Andrea: 100%
- Marogel Dragoș: 100%