DOCUMENTAȚIE TEHNICĂ - PROIECT FIXITNOW

Student: Stan Ioan Alexandru

Grupa: 323AA

Proiect: FixItNow

1. ARHITECTURA APLICAȚIEI

Aplicația **FixItNow** este un sistem software orientat pe obiecte (OOP) destinat simulării și administrării activității unui service de electrocasnice. Arhitectura este modulară, centralizată în jurul clasei Service, care acționează ca un "Manager" al întregului sistem.

Proiectul respectă principiile fundamentale OOP:

- **Encapsulare:** Datele sensibile (ex: salariul, starea cererii) sunt protejate și modificate doar prin metode publice.
- **Moștenire:** Ierarhiile de clase Angajat și Electrocasnic permit extensibilitate ușoară.
- **Polimorfism:** Metodele virtuale (ex: calculeazaSalariu, afisare) permit tratarea uniformă a obiectelor, indiferent de tipul lor concret.
- **Single Responsibility:** Fiecare clasă gestionează o singură responsabilitate (ex: Tehnician știe doar să repare, nu să aloce cereri).

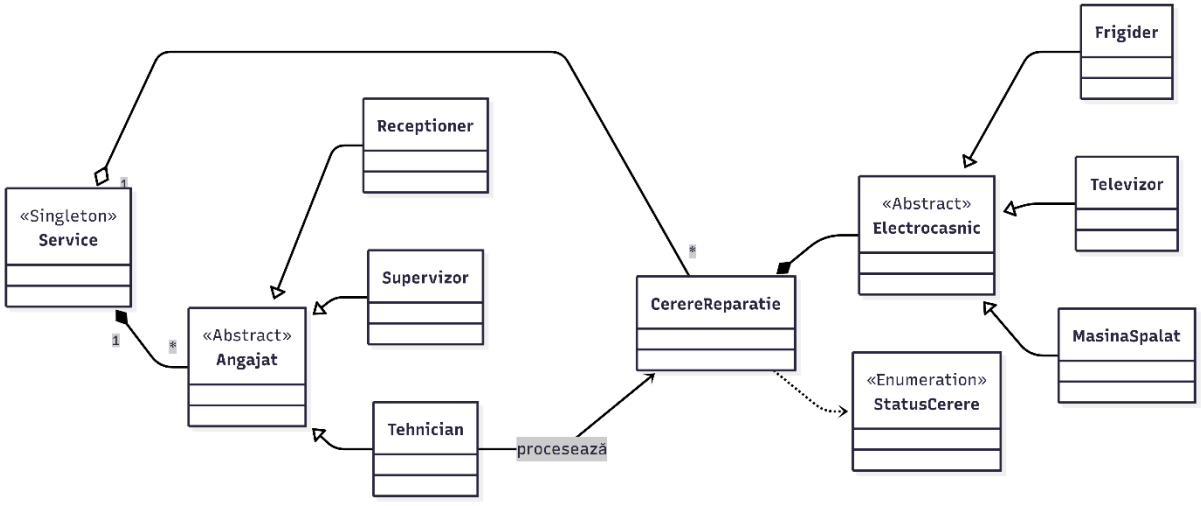
2. DIAGRAMA CLASELOR ȘI RELAȚIILE DINTRE ELE

Sistemul este compus din următoarele entități principale:

A. Clasa Service (Singleton)

Este "creierul" aplicației. Implementează design pattern-ul **Singleton** pentru a garanta o instanță unică ce gestionează resursele partajate (lista de angajați, coada de cereri, catalogul de modele).

- **Relații:** Agregare (conține vectori de Angajat* și cozi de CerereReparatie*).



B. Ierarhia Angajat

- **Angajat (Clasă Abstractă):** Definește atributele comune (Nume, Prenume, CNP, Data Angajării).
- **Tehnician:** Moștenește Angajat. Are un vector de specializări (ce știe să repare) și o listă proprie de cereri active. Implementează metoda simuleazaTic() pentru a avansa progresul reparațiilor.
- **Receptioner / Supervizor:** Moștenesc Angajat. Implementează algoritmi specifici pentru calculul salarial (sporuri, bonusuri).

C. Ierarhia Electrocasnic

- **Electrocasnic (Clasă Abstractă):** Modelează aparatul defect.
- **Televizor / Frigider / MasinaSpalat:** Clase concrete ce adaugă atribute specifice (diagonala, congelator, capacitate).

D. CerereReparatie

Reprezintă unitatea de lucru. Leagă un Electrocasnic de un timestamp, o complexitate și un status. Calculează automat durata estimată (vechime * complexitate) și costul.

---

3. MODUL DE FUNCȚIONARE

Fluxul aplicației este secvențial și interactiv, fiind gestionat de funcția main. Interacțiunea cu utilizatorul se realizează printr-un meniu principal recurent (implementat într-o buclă while), care permite efectuarea continuă de operațiuni până la comanda de ieșire.

Controlul execuției este asigurat de o structură decizională de tip **switch**, care mapează opțiunea introdusă de la tastatură la metodele specifice ale instanței unice Service. De exemplu, selectarea unei opțiuni declanșează încărcarea datelor, o alta pornește motorul de simulare, iar alta generează rapoartele, asigurând o separare clară între interfață și logica de business.

Pasul 1: Inițializare și Validare

Utilizatorul încarcă datele inițiale din fișiere CSV (modele.txt, angajati.txt, specializari.txt, cereri.txt).

- La citire, sistemul validează integritatea datelor (CNP valid, vârsta minimă 16 ani, existența modelului în catalog).
- Liniile invalide sunt respinse, iar erorile sunt afișate detaliat (linia și cauza), fără a opri execuția programului.

Pasul 2: Simularea (Motorul de Alocare)

Nucleul aplicației este funcția ruleazaSimulare(), care avansează timpul în unități discrete. La fiecare pas se întâmplă două acțiuni majore:

1. **Procesarea:** Fiecare Tehnician parcurge lista sa de cereri active și scade durata rămasă. Dacă o cerere ajunge la 0, este marcată ca FINALIZATA și mutată în istoric.
2. **Alocarea (Dispatcher):** Clasa Service verifică coada de așteptare (cereriAsteptare). Pentru fiecare cerere, caută un tehnician care îndeplinește simultan condițiile:
    - Este specializat pe Tipul și Marca aparatului.
    - Are disponibilitate (mai puțin de 3 cereri active).
    - Dacă se găsește, cererea este mutată din Așteptare în lista tehnicianului.

Pasul 3: Raportarea

Pe baza datelor acumulate în simulare, sistemul poate genera rapoarte CSV complexe (top 3 salarii, cea mai lungă reparație, situația cozii de așteptare), folosind algoritmi de sortare și filtrare, prin intermediul metodei clasei Service **genereazaRapoarte**()

---

4. ELEMENTE TEHNICE ȘI IMPLEMENTARE

În dezvoltarea proiectului au fost utilizate concepte avansate de C++ și biblioteci standard (STL) pentru a asigura o simulare robustă și realistă:

A. Parsare și Procesare Date

Pentru citirea eficientă a fișierelor de configurare (CSV), s-a implementat funcția readCSVLine care utilizează clasa **std::stringstream**. Aceasta permite descompunerea fiecărei linii citite în tokeni (câmpuri individuale) pe baza delimitatorului virgulă, facilitând conversia ulterioară a datelor în tipuri specifice (int, double, string).

B. Gestionarea Timpului și Simularea

Aspectul temporal al simulării a fost gestionat prin două mecanisme distincte:

1. **Timestamp-uri:** S-au utilizat bibliotecile <ctime> și structura tm pentru a genera și stoca momentul exact al creării cererilor și angajării personalului, asigurând trasabilitatea acțiunilor.
2. **Temporizare:** Pentru a vizualiza progresul reparațiilor în timp real, bucla principală a simulării integrează funcția **std::this_thread::sleep_for** (din biblioteca <thread> și <chrono>). Aceasta introduce o pauză controlată (ex: 500ms) între iterări, simulând trecerea unităților de timp fără a bloca resursele procesorului.

C. Provocare Tehnică: Compatibilitatea Cross-Platform

O provocare majoră a fost gestionarea diferențelor de formatare a fișierelor text între Windows (CRLF - \r\n) și Linux/WSL (LF - \n).

- **Simptom:** Citirea string-urilor eșua la comparații (ex: "Samsung" != "Samsung\r"), iar afișarea în consolă era decalată.
- **Soluție:** S-a implementat o logică de curățare (trimming) direct în funcția readCSVLine. Aceasta elimină automat caracterele de control (\r) după extragerea prin stringstream, asigurând funcționarea corectă a aplicației indiferent de sistemul de operare pe care au fost create fișierele de test.

---

5. DESCRIEREA TESTELOR

Testarea aplicației s-a realizat prin scenarii predefinite, încărcate din fișiere text, menite să verifice atât funcționarea corectă ("Happy Path"), cât și gestionarea erorilor ("Edge Cases").

A. Fișierul modele.txt

Definește catalogul service-ului. Testează capacitatea sistemului de a gestiona multiple mărci și modele. Orice cerere pentru un aparat care nu se regăsește aici este automat respinsă și contorizada în statistica de refuzuri.

B. Fișierul angajati.txt

Testează validările constructorului clasei Angajat. Scenariile includ:

- **Angajați Valizi:** Tehnicieni, Recepționeri și Supervizori cu date corecte.
- **Date Invalide:**
    - CNP cu lungime incorectă.
    - CNP valid structural, dar care indică o vârstă sub 16 ani (interzis la angajare).
    - Tip de angajat necunoscut (ex: caracterul 'X'). Aceste teste demonstrează robustețea sistemului, care raportează eroarea și linia, dar continuă să încarce restul angajaților valizi.

C. Fișierul specializari.txt

Realizează legătura competențelor. Testele verifică dacă sistemul permite alocarea specializărilor doar către Tehnicieni existenți. Încercarea de a adăuga o specializare unui CNP inexistent sau unui Receptioner este blocată și logată.

D. Fișierul cereri.txt (Scenariul de Simulare)

Acesta este cel mai complex test, conceput pentru a verifica algoritmul de alocare și coada de așteptare:

1. **Sarcini Complexe:** Primele cereri sunt pentru aparate foarte vechi (ani 2010-2015) cu complexitate maximă. Durata reparației (vechime * complexitate) este mare, ceea ce blochează rapid tehnicienii disponibili.
2. **Supraîncărcare:** Fișierul conține mai multe cereri valide decât capacitatea totală a tehnicienilor (3 tehnicieni x 3 sloturi = 9 cereri simultane).
3. **Efectul de Coadă:** Se observă cum cererile excedentare rămân în starea ASTEPTARE. Pe măsură ce simularea avansează și tehnicienii finalizează primele sarcini grele, aceștia preiau automat cererile din coadă.

Concluzie

Proiectul îndeplinește toate cerințele specificate, oferind o simulare funcțională, gestiune completă a datelor și rapoarte detaliate, fiind protejat împotriva datelor de intrare eronate.