🏢 Simulador de Elevador

Este projeto foi desenvolvido em parceria entre Vinícius e Ygor, utilizando a ferramenta Code With Me do IntelliJ IDEA, que permitiu a colaboração em tempo real na codificação do sistema. Este projeto é um simulador de elevadores em um prédio com múltiplos andares, onde diferentes algoritmos de decisão são aplicados para controlar os elevadores em situações simuladas.

🚀 Objetivo

Simular o comportamento de elevadores em um prédio com base em diferentes estratégias de despacho de chamadas:

- Normal: Prioriza o elevador com menos destinos ativos.
- Economia: Escolhe o elevador com menor consumo de energia para atender a chamada.
- Felicidade: Tenta reduzir o tempo de espera das pessoas.

🛠️ Tecnologias Utilizadas

- Java
- Estruturas de dados personalizadas (listas, filas, nós encadeados)
- Serialização de dados para simulação com arquivos .dat

📁 Estrutura do Projeto

```
Simulador_Elevador/
├── src/
│   ├── Main.java # Classe principal para execução
│   ├── classes/ # Lógica principal do sistema
│   │   ├── Andar.java
│   │   ├── CentralDeControle.java
│   │   ├── Elevador.java
│   │   ├── Painel.java
│   │   ├── Pessoa.java
│   │   ├── Predio.java
│   │   ├── Serializacao.java
│   │   └── Simulador.java
│   └── estruturas/ # Estruturas de dados implementadas
│   ├── FilaComum.java
│   ├── FilaPrioridade.java
│   ├── ListaDinamica.java
│   ├── ListaDupla.java
│   ├── ListaEstatica.java
```

```
│ └── Random.java
├── saves/ # Arquivos de simulação
│ └── predio.dat # Exemplo
└── README.md # Este arquivo
```

Modelagem do Sistema

Classes Principais:

- Andar
    - numero (int) → Número do andar (identificador do andar).
    - fila (FilaPrioridade) → Fila de pessoas esperando o elevador, organizada por prioridade.
    - pessoas (ListaDinamica)→ Lista de pessoas atualmente presentes no andar (não necessariamente esperando o elevador).
    - painel (Painel)→ Painel de chamada de elevador, responsável por registrar pedidos.

- CentralDeControle
    - elevadores (ListaEstatica<Elevador>) → Lista de elevadores que fazem parte do sistema.
    - predio (Predio)→ Referência ao prédio onde a simulação ocorre (estrutura dos andares).
    - estado (EstadoCentralDeControle) → Estado atual da simulação (ex.: executando, pausado, finalizado).
    - energiaGasta (int) → Total de energia consumida na simulação (paradas + deslocamentos).
    - tempoEsperaTotal (int) → Soma total dos tempos de espera de todas as pessoas atendidas.
    - maiorTempoEspera (int) → Maior tempo de espera registrado entre todas as pessoas na simulação.
    - chamadasAtendidas (int) → Número total de chamadas de elevador atendidas.
    - energiaDeslocamento (int) → Energia consumida exclusivamente com movimentação (subir/descer).
    - energiaParada (int) → Energia consumida com paradas (abrir/fechar portas, paradas sem movimentação).

- Elevador
    - numeroElevador (int) → Identificador único do elevador.
    - andarAtual (int) → Andar atual em que o elevador está.
    - destino (int) → Próximo andar de destino.
    - pessoasDentro (ListaDinamica<Pessoa>) → Pessoas que estão atualmente dentro do elevador.

    - pessoasSaida (ListaDinamica<Pessoa>) → Histórico de pessoas que já utilizaram o elevador.
    - destinos (ListaDinamica<Integer>) → Lista de andares que o elevador deve atender.
    - CAPACIDADE_MAXIMA (int) → Capacidade máxima de pessoas no elevador.
    - estado (EstadoElevador) → Estado atual do elevador (SUBINDO, DESCENDO, PARADO).

- Painel
    - botao (boolean) → Indica se o botão de chamada do elevador foi pressionado no andar.
- Pessoa
    - id (int) → Identificador único da pessoa (funciona como "nome").
    - prioridade (int) → Indica o nível de prioridade da pessoa (ex.: se é prioridade ou não para o elevador).
    - contadorDeDestinos (int) → Quantidade total de destinos que a pessoa tem na simulação (pode variar entre 2 e 4).
    - andarOrigem (int) → Andar onde a pessoa foi gerada ou iniciou sua trajetória (atualiza quando desembarca e gera um novo destino).
    - andarDestino (int) → Próximo andar de desembarque da pessoa.
    - destinos (ListaEstatica<Integer>) → Lista de todos os destinos que a pessoa precisa visitar durante a simulação.
    - andarAtual (int) → Andar atual onde a pessoa se encontra (seja dentro do elevador ou no prédio).
    - dentroElevador (boolean) → Indica se a pessoa está dentro de algum elevador (true) ou não (false).
    - tempoAndar (int) → Tempo que a pessoa permanece em um andar após desembarcar, antes de gerar uma nova chamada (simula atividades dentro do andar).
    - tempoEspera (int) → Tempo de espera acumulado enquanto a pessoa aguarda o elevador.

- Predio
    - andares (ListaEstatica<Andar>) → Lista de todos os andares do prédio.
    - centralDeControle (CentralDeControle) → Objeto responsável por gerenciar os elevadores e a lógica do sistema.
    - randomizacao (Random) → Gerador de números aleatórios utilizado para criar pessoas, destinos e outros comportamentos estocásticos.
    - tempoMovimentoElevador (int) → Tempo (em segundos) que um elevador leva para se deslocar de um andar para outro.
    - horarioPico (boolean) → Indica se a simulação está em horário de pico, onde a geração de pessoas é mais intensa.
    - andaresAleatorios (boolean) → Define se as pessoas são geradas em andares aleatórios (true) ou sempre no térreo (false).

- Simulador
    - segundosSimulados (int) → Tempo total que já foi simulado, em segundos.

- velocidadeMs (int) → Velocidade da simulação, em milissegundos (tempo de intervalo entre cada avanço do relógio da simulação).
- temporizador (transient Timer) → Objeto que controla a execução periódica da simulação. Marcado como transient porque não deve ser serializado.
- emExecucao (boolean) → Indica se a simulação está atualmente em execução (true) ou pausada (false).
- predio (Predio) → Instância do prédio que está sendo simulado, incluindo andares, elevadores e central de controle.
- duracaoSimulacao (int) → Duração total da simulação em segundos.
- tempoMovimentoElevador (int) → Tempo que cada elevador leva para se deslocar de um andar para outro, em segundos (pode ser usado para sincronizar com a velocidade da simulação).

Estruturas de Dados Utilizadas:

- ListaEstatica.java
  - Implementação própria de uma lista sequencial.
  - Usada para armazenar andares e elevadores.

- ListaDinamica.java
  - Implementação própria de uma lista dinâmica.
  - Usada para armazenar as pessoas no andar, fora da fila de espera, lista de pessoas dentro do elevador, lista de pessoas a sair do elevador e lista de destinos dos elevadores.

- FilaPrioridade.java
  - Implementação própria de fila com prioridade.
  - Usada para organizar as pessoas nas filas (considerando se são prioridade ou não).

- FilaSimples.java
  - Usada para gerenciar as filas presentes na FilaPrioridade.

Obs.: Não utilizamos estruturas prontas do Java (como ArrayList ou Queue) para cumprir os requisitos do projeto acadêmico.

▶️ Como Executar

1. Abra o projeto em uma IDE como IntelliJ ou Eclipse.
2. Compile todos os arquivos do diretório src/.
3. Insira os valores desejados na classe Main.java.
4. Execute a classe Main.java.

📊 Modos de Simulação

Modo Normal

- Seleciona o elevador com menos destinos (mais "livre").
- Menor complexidade computacional.

Modo Economia

- Tenta minimizar o consumo de energia.
- Avalia distância e quantidade de mudanças de rota.

Modo Felicidade

- Prioriza a satisfação dos usuários.
- Tenta atender o andar com o menor tempo de espera possível.

💾 Salvamento

A simulação utiliza serialização para carregar e salvar dados no diretório saves/.

✍️ Autor

Projeto acadêmico para fins educacionais. Comentários e melhorias são bem-vindos!