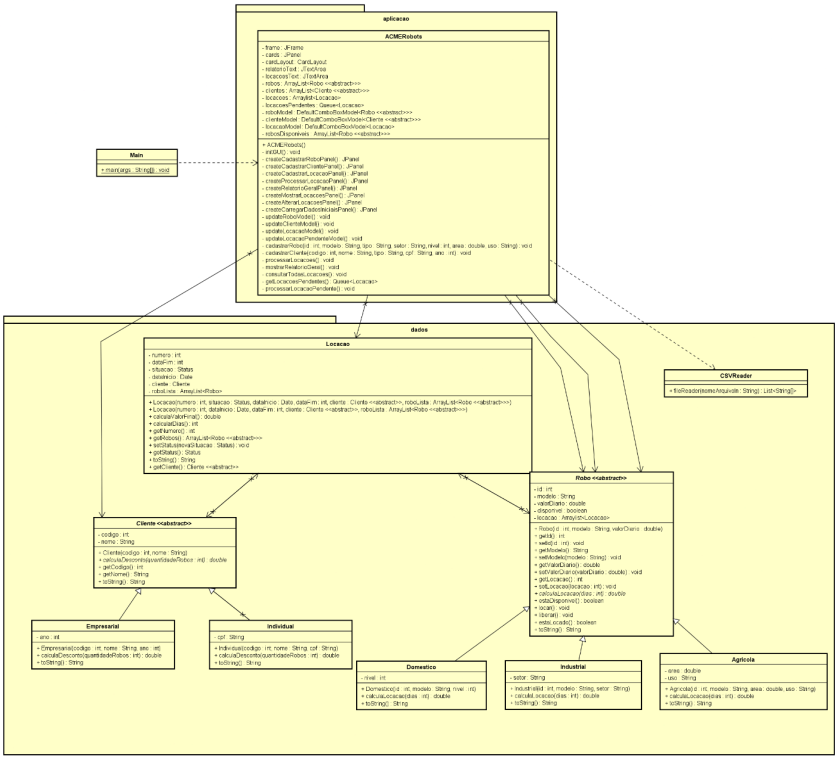
Relatório de Implementação ACMERobots

Pedro Tonial, Filipe Pereira

1. Diagrama de Classes

Diagrama de Classes pode ser encontrado no arquivo “ACMERobotsASTAH.asta” dentro da pasta .zip enviada no moodle



2. Coleções de dados

2.1. Descrição das coleções utilizadas

2.1.1. Listas (List)

Utilizadas para armazenar e manipular grupos de objetos em sequência. As listas são dinâmicas e permitem acesso, inserção e remoção de elementos por índice.

Exemplos de uso:

- Armazenamento de clientes em diferentes categorias (Empresarial, Doméstico, etc.).
- Armazenamento de robôs em diferentes aplicações (Industrial, Agrícola, etc.).

- Armazenamento em geral de toda a aplicação.
- Manipulação de todos os objetos armazenados após criação.

2.1.2. Arrays

Utilizados para armazenar múltiplos elementos do mesmo tipo em uma estrutura de dados de tamanho fixo. Arrays são úteis quando o tamanho da coleção é conhecido e não muda.

Exemplos de uso:

- Armazenamento temporário de dados onde o tamanho é conhecido e fixo.
- Implementação de algoritmos que requerem acesso rápido aos elementos por índice.
- Implementação da leitura de arquivos CSV.
- Elementos de bibliotecas nativas de java como JComboBox que faz uso de arrays de String.

3. Armazenamento (persistência) de dados

3.1 Serialização

Para a persistência dos dados, optamos pela serialização dos objetos Java em arquivos com extensão `.ser`. A serialização é o processo de converter um objeto em um formato que pode ser facilmente armazenado e reconstruído posteriormente. Em Java, a serialização é realizada utilizando a interface `Serializable`.

3.2. Serialização de Objetos para Arquivos .ser:

Para serializar um objeto em Java, a classe do objeto deve implementar a interface `Serializable`. Isso permite que o objeto seja convertido em uma sequência de bytes e armazenado em um arquivo.

3.3. Desserialização de Objetos de Arquivos .ser

A desserialização é o processo inverso, onde os bytes armazenados são convertidos de volta em um objeto Java. Isso é feito utilizando a classe `ObjectInputStream`.

3.4. Benefícios da Serialização:

1. Persistência:
    a. Facilita o armazenamento de objetos complexos, mantendo seu estado e estrutura.
2. Facilidade de uso:

    a. Simplicidade na implementação e uso das classes `ObjectOutputStream` e `ObjectInputStream`.
3. Compatibilidade:
    a. Os arquivos `.ser` podem ser usados para salvar o estado dos objetos entre execuções do programa, garantindo que dados importantes não sejam perdidos.

4. Leitura de Arquivos CSV

4.1. Desenvolvimento da Leitura de Arquivos por CSV

Para a leitura de arquivos CSV, foi desenvolvida uma abordagem que permite a fácil importação de dados estruturados em arquivos CSV para objetos Java. Esta funcionalidade é importante para a manipulação de grandes volumes de dados de forma eficiente e confiável.

4.2. Vantagens da Leitura de CSV

- Eficiência: A leitura direta de arquivos CSV para objetos Java permite a manipulação eficiente de grandes conjuntos de dados.
- Flexibilidade: O uso de mapeamento baseado em cabeçalhos facilita a adaptação a diferentes formatos de CSV, desde que os cabeçalhos correspondam aos nomes dos atributos dos objetos.
- Compatibilidade: Arquivos CSV são amplamente compatíveis com diversas ferramentas e sistemas, facilitando a importação e exportação de dados.