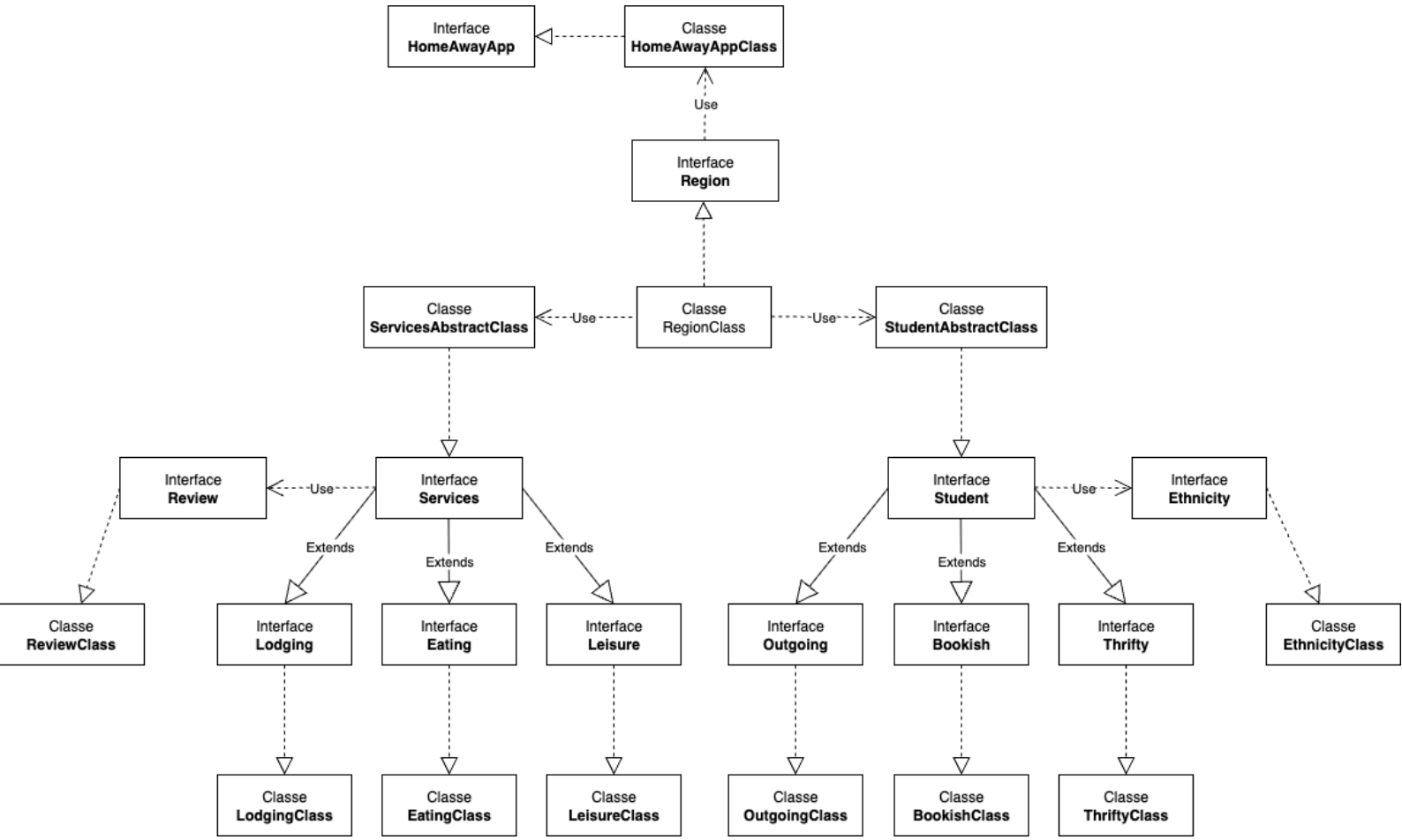
Relatório do primeiro projeto de AED

Guilherme Henriques Marques Martin, nº 71003, André Jordao Braz Pais Amante, nº70945



Relatório do Diagrama de Classes - 1ª Fase do Projeto "Homeaway From Home"

1. Entidades do Domínio e Justificativas de Estruturas de Dados

- As quatro entidades principais do domínio são **Região**, **Serviço**, **Estudante** e **Review (Star)**.
- **Região (RegionClass)**: Representa a área geográfica ativa e o seu *bounding box*.
    - **Atributos**: Deve armazenar os limites geográficos (quatro paramatros long para Latitude e Longitude) e coleções de todas as outras entidades.
    - **Estruturas**: Utiliza um **DoublyLinkedList** para gerir os **Estudantes**, mapeando o nome do estudante para o objeto, permitindo buscas rápidas (O(1) esperada)

para comandos como leave, go, e where. Utiliza tambem um DoublyLinkedList para os **Serviços**, pois o comando services exige tambem a listagem por ordem de registo (inserção).
    - **Funcionalidades**: Controlo de **bounds**, **save**, **load** da área, e métodos para adicionar serviços e estudantes.
- **Serviço (ServiceAbstractClass / Service)**: Entidade central com diferentes tipos (eating, lodging, leisure).
    - **Atributos**: Nome, localização (long latitude/longitude), e a média de **estrelas** (int arredondado).
    - **Estruturas**: Cada serviço contém um List de Review para manter o histórico por ordem de inserção, crucial para o cálculo da média e para as pesquisas por tag e ordenação por tempo.
    - **Funcionalidades**: Métodos para processar o comando **star** (avaliação) e para retornar as suas coordenadas (**where**).
- **Serviços Específicos (EatingClass, LodgingClass, LeisureClass)**: Especializações do Service.
    - **Atributos Específicos**: Armazenam preço e valor (long/int) (e.g., custo do menu, capacidade, desconto).
    - **Estruturas (Eating/Lodging)**: Utilizam um **List** de **Estudantes** para gerir e listar os ocupantes e verificar a lotação (ServiceFull error), suportando o comando **users** (listagem ordenada).
- **Estudante (StudentAbstractClass / Student)**: Representa os utilizadores da aplicação.
    - **Atributos**: Nome, País de origem, referência à **Localização Atual** (Service) e à **Casa** (LodgingClass).
    - **Funcionalidades**: Comandos de movimento (**go**, **move**) e métodos de busca por localização (**where**) e serviço relevante (**find**).
- **Estudantes Específicos (BookishClass, OutgoingClass, ThriftyClass)**: Especializações do Student.
    - **Estruturas (Bookish/Outgoing)**: Utilizam um **List** de **Serviços** (Service) para armazenar os locais visitados na ordem cronológica de visita, conforme exigido pelo comando visited.
    - **Atributos Específicos (Thrifty)**: Devem armazenar o **menor preço de comer/dormir** registado para impor as restrições de movimento e a regra do *distracted*.

2. Relações de Herança

- O *design* do sistema beneficia da herança (ou interfaces/classes abstratas) para modelar as variações de comportamento e dados:
- **Herança de Serviço**: As classes concretas **EatingClass**, **LodgingClass**, e **LeisureClass** herdam de uma classe abstrata **ServiceAbstractClass**. Isto permite que partilhem atributos comuns (nome, localização, avaliação), enquanto implementam a lógica específica de validação de preços/valores e o gerenciamento de lotação exigido para *Eating* e *Lodging*.
- **Herança de Estudante**: As classes concretas **BookishClass**, **OutgoingClass**, e **ThriftyClass** herdam de uma classe abstrata **StudentAbstractClass**. Esta relação é fundamental para implementar as regras distintas de cada tipo: o **armazenamento seletivo de visitas** (visited) e a **lógica de decisão** no comando find (melhor média vs. menor preço).