Ficha Prática #05

5.1 Objectivos

1. Praticar a utilização de **Diagramas de Classe** e de **Diagramas de Sequência**;
2. Relacionar diagramas de Classe e de Sequência com a implementação que eles representam.
3. Desenvolver a capacidade de utilizar diagramas de Classe e de Sequência para conceber sistemas.

5.2 Exercícios

Resolvas os exercícios abaixo propostos criando os diagramas pedidos.

5.2.1 Sistema de Avaliação de Trabalhos

Considere o excerto de código Java apresentado no Anexo A, relativo a um subsistema de gestão de trabalhos práticos numa Universidade.

Relativamente ao código apresentado, resolva os seguintes exercícios:

1. Analise o código e apresente o correspondente **Diagrama de Classes**, procurando ser o mais exaustivo possível na identificação dos relacionamentos entre as classes. Considere que ao nível da *Facade*, todas as associações correspondem a composições.
2. Desenhe **Diagramas de Sequência** para os seguintes métodos:
   (a) `public Aluno getAluno(String codAluno)` — O método deverá devolver o aluno com o número indicado.

(b) `public int getNotaAluno(String codAluno)` — O método deverá calcular a nota de um aluno, sabendo que a nota teórica e prática valem 60% e 40% da nota final, respectivamente.

(c) `public void registaEntrega(Entrega e, String codGrupo)` — O método deverá registar uma entrega no grupo indicado, caso ainda não exista uma entrega para essa data.

(d) `public boolean validaAvaliadores()` — O método deverá verificar que nenhum aluno seja avaliador do seu próprio grupo.

5.2.2 Empresa de Transportes

Considere o diagrama de classes apresentado na Figura 5.1, que representa uma solução para uma empresa de transportes públicos, e responda às seguintes questões:

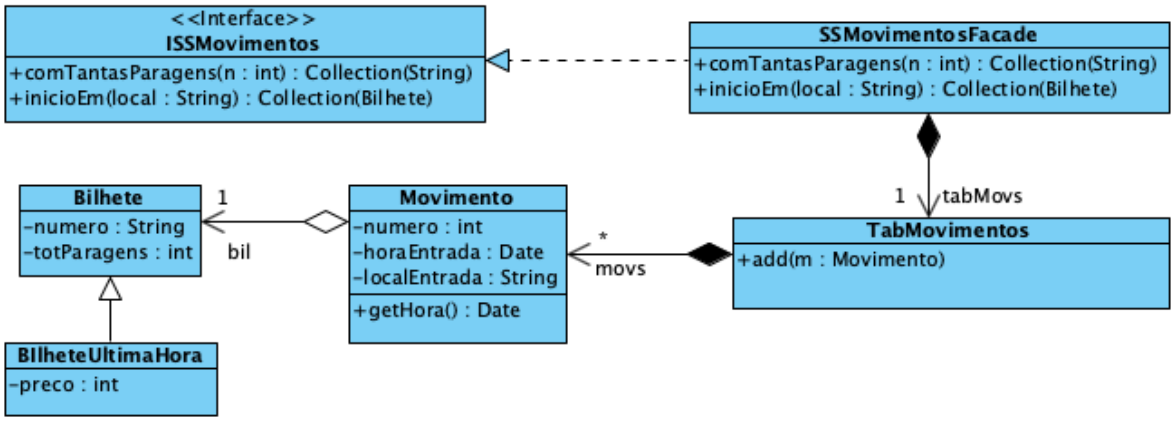

Figura 5.1: Diagrama de classes para o Exercício 5.2.2

1. Escreva o código Java de uma implementação possível para este diagrama.
2. Defina uma pré-condição para a operação `add(Movimento)` (ver `TabMovimentos`), de modo a garantir que a operação não viola a seguinte restrição:

   Não podem existir dois movimentos com o mesmo número numa instância de TabMovimentos.

3. Escreva um **Diagrama de Sequência** para a operação:

```
comTantasParagens(n: int): Collection(String)
```

da classe `SSMovimentosFacade`, que determina os códigos de todos os bilhetes que fizeram viagens (`Movimentos`) com um número de paragens igual ao valor `n` dado como parâmetro.

4. Escreva um **Diagrama de Sequência** para a operação:

```
inicioEm(local: String): Collection(Bilhete)
```

da classe `SSMovimentosFacade`, que determina todos os bilhetes que fizeram viagens com início em `local`.

5.2.3 Parques de Estacionamento

Considere que, no contexto do Sistema de Informação da Universidade, se pretende modelar um subsistema de administração de parques de estacionamento. Neste sistema os utentes, para que possam estacionar nos diversos parques, devem possuir um identificador (um dispositivo físico) que está associado a uma determinada viatura. Caso um cliente tenha mais do que uma viatura é necessário que adquira tantos identificadores quantas as viaturas que possui. Sempre que uma viatura utiliza um parque, é registada a hora e data de entrada e a hora e data de saída do parque (o identificador comunica o seu número ao passar nas cancelas de entrada/saída). Cada parque tem uma tabela de preços baseada no tipo de viatura. Essa informação é fornecida pela associação que existe entre o identificador e a viatura.

A equipa de análise e o cliente desenvolveram, em conjunto, o modelo de domínio apresentado na Figura 5.2 e, durante a análise de requisitos, ficou definido que deverá ser possível a um utente pesquisar, e listar, os seus movimentos nos diversos parques, bem assim como obter os extractos de conta mensais para um determinado identificador. Tendo, a partir dos Use Case, sido possível identificar a necessidade de implementar as operações referidas nas alíneas (4) a (6) abaixo e sabendo que:

- a direcção a utilizar nas associações deverá ser definida em função dos métodos pedidos nas alíneas (4) e (5);
- para simplificar, numa primeira abordagem, o subsistema de gestão dos parques tem como *Facade* a classe `GesParkFacade` que implementa a interface da Figura 5.3.

Responda às seguintes questões:

1. Complete, caso seja necessário, o Modelo de Domínio.

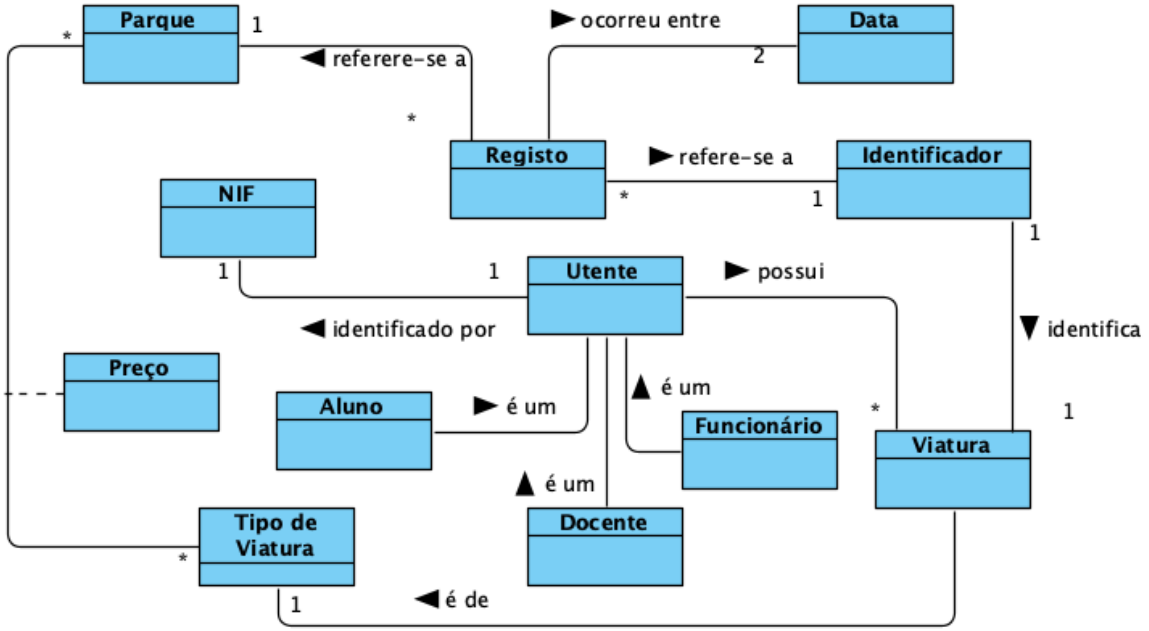

Figura 5.2: Modelo de Domínio para o Exercício 5.2.3

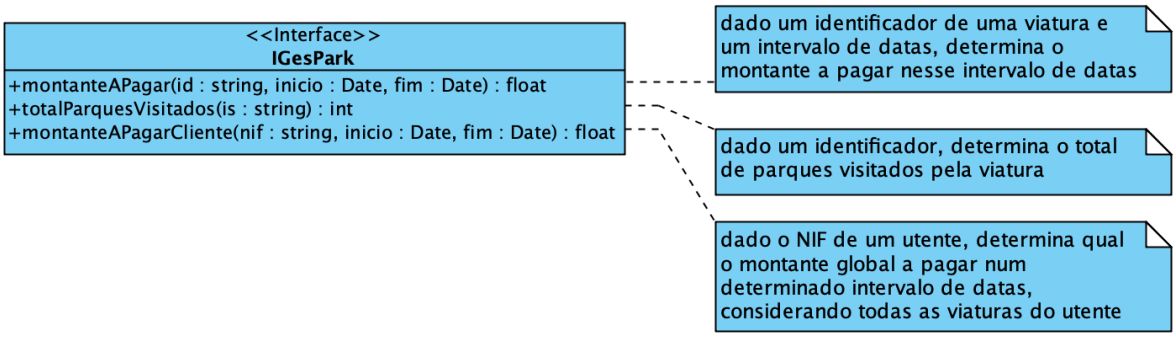

Figura 5.3: Interface do subsistema GesPark

2. Construa o **Diagrama de Classes** da arquitectura do subsistema de gestão dos parques, a partir da informação fornecida no Modelo de Domínio e sabendo que a classe `Utente` está definida no subsistema `SubSistemaUtentes`. Seja o mais completo possível na construção do mesmo, identificando as relações, nomeando-as e colocando as multiplicidades respectivas.

3. Acrescente ao diagrama da alínea anterior, caso seja necessário, o **OCL** relativo à seguinte restrição:

   > Todos os registos de um parque são relativos a viaturas registadas no sistema.

4. Construa o **Diagrama de Sequência** para a operação da classe `GesPark` que, dado um identificador de uma viatura e um intervalo de datas, determina o montante a pagar nesse intervalo de datas.

   A operação deverá ter como pré-condição a existência do identificador dado. Acrescente-a ao Diagrama de classes.

5. Construa o **Diagrama de Sequência** para a operação da classe `GesPark` que, dado um identificador, determina o total de parques visitados pela viatura a que este está associado.

   A operação deverá ter como pré-condição que o identificador existe e como pós-condição que o resultado é não negativo. Acrescente-as ao Diagrama de classes.

6. Considere agora que o subsistema possui informação sobre os utentes que se registaram nos parques. Acrescente essa informação à arquitectura e construa o **Diagrama de Sequência** da operação que permitirá calcular, para todas as viaturas de um utente, identificado pelo seu NIF, qual o montante global a pagar num determinado intervalo de datas.

   A operação deverá ter como pré-condição que o utente está registado no subsistema. Acrescente-a ao Diagrama de Classes.

5.2.4 Lumen

Relembre o Exercício 4.2.3. Com base na análise feita ao caso de uso "Instalar Painéis" (ver Figura 4.3, na página 22), desenvolva a arquitetura da lógica de negócio e os diagramas de sequência para as suas operações.

Anexo A

Código para o Exercício 5.2.1:

```
public interface IGestTurmas {
   public Aluno getAluno(String codAluno)
   float getNotaAluno(String codAluno);
   void registaEntrega(Entrega e, String codGrupo);
   boolean validaAvaliadores();
}

public interface Identificavel {
   String getID();
}

public abstract class Pessoa {
   protected String nome;
   public abstract void setNome(String n);
}

public class Aluno extends Pessoa implements Identificavel {
   private Grupo meuGrupo;
   private String numAluno;
   private int notaTeo;
   private int bounsPrat;
   public void regista(Grupo g) {...}
   public String getID() {...}
   public void setNome(String n) {...}
}

public class Grupo implements Identificavel {
   private String cod;
   private int nota;
   private List<Entrega> entregas;
   public void addEntrega(Entrega e) {...}
   public String getID() {...}
}
```

```
public class Entrega {
    private Date data;
    private int nota_docente;
    private Aluno avaliador;
    private int nota_avaliador;
    private String comentarios;
}

public class Docente extends Pessoa implements Identificavel{
    private String cod;
    public String getID() {...}
    public void setNome(String n) {...}
}

public class SSGesTurmasFacade implements IGestTurmas {
    private Docente responsavel;
    private List<Docente> docentes_praticas;
    private Map<String,Aluno> alunos;
    private List<Grupo> grupos;
    ...
}
```