Linguagem de Programação Orientada a Objetos

Trabalho Prático

O objetivo deste trabalho é permitir a aplicação dos conceitos de orientação a objetos em um problema real. Para isso vocês devem escolher um problema de sua preferência e implementar telas de CRUDs (*Create*, *Read*, *Update* and *Delete*) utilizando interface gráfica em Java.

Pense em um cenário que contenha pelo menos 4 classes principais, de tal forma que exista pelo menos um relacionamento **1:1**, um relacionamento **1:n** e um relacionamento **n:n** entre elas. A imagem abaixo é um exemplo das classes utilizadas em um sistema dedicado à organização de corridas de rua.

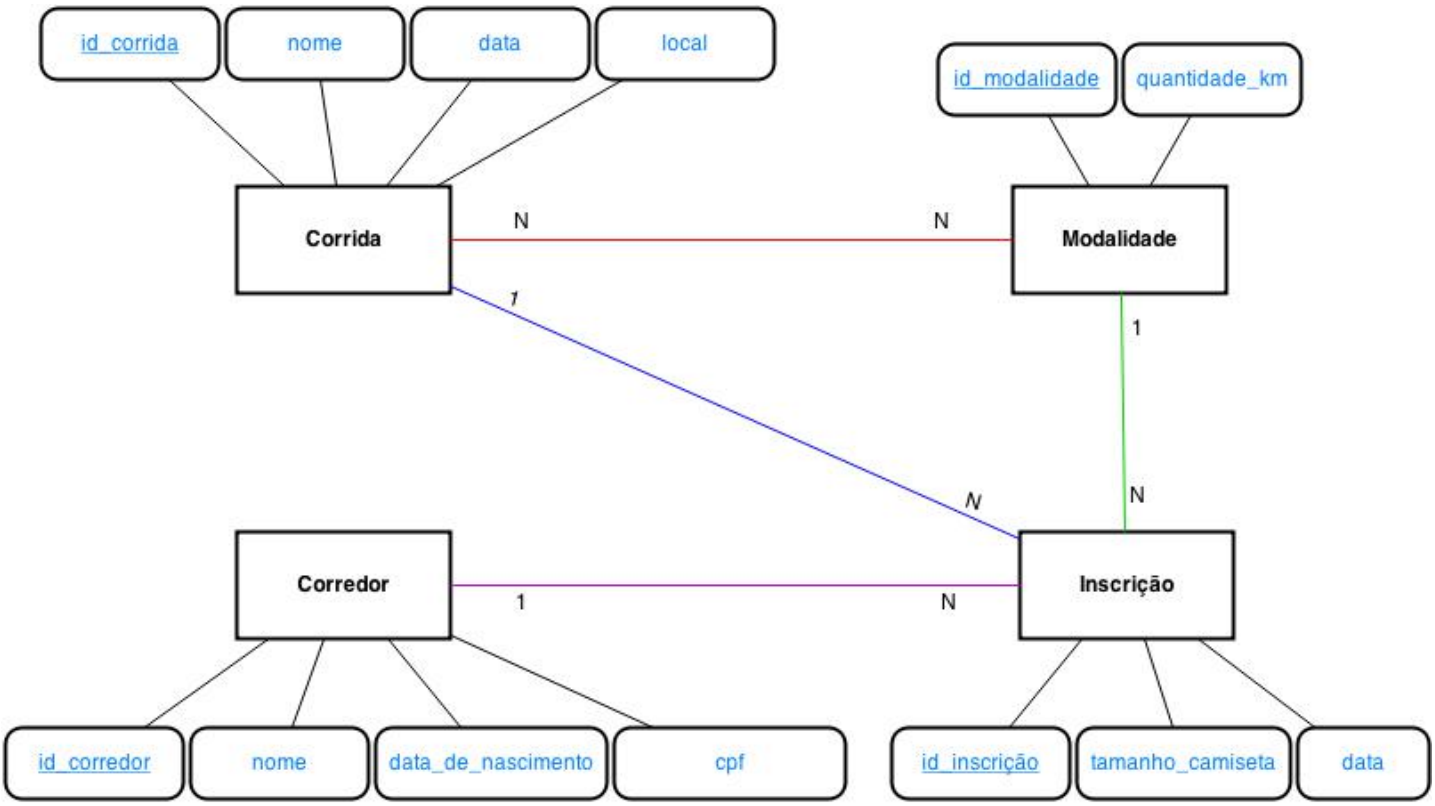

Figure 1: Exemplo de conjunto de classes.

Esse diagrama pode ser interpretado da seguinte maneira:

- Uma **Corrida** tem atributos *id_corrida* (que identifica unicamente uma corrida), *nome*, *data* e *local*;
- Uma **Modalidade** tem atributos *id_modalidade* (que identifica unicamente uma modalidade) e *quantidade_km*;
- Um **Corredor** tem atributos *id_corredor* (que identifica unicamente um corredor), *nome*, *data_de_nascimento* e *cpf*;
- Uma **Inscrição** tem atributos *id_inscrição* (que identifica unicamente uma inscrição), *tamanho_camiseta* (que pode ser *p*, *m* ou *g*) e *data*;
- Linha vermelha: Uma *Corrida* por ter várias *Modalidades*. Por exemplo, a volta das nações tem modalidades 21 Km, 10 Km e 7 Km. Além disso, uma mesma modalidade pode estar associada a diferentes corridas. Por exemplo, tanto a volta das nações quanto a corrida da UFMS têm a modalidade de 5 Km. Esse tipo de relacionamento é conhecido como *muitos-para-muitos*;

- Linha rosa: Um *Corredor* pode ter várias *Inscrições*, ou seja, pode participar de várias *Corridas* diferentes. Entretanto, uma determinada *Inscrição* só pode estar associada a um *Corredor*. Esse tipo de relacionamento é conhecido como *um-para-muitos*;
- Linha azul: uma *Inscrição* de um *Corredor* deve estar relacionada a uma *Corrida*, e uma *Corrida* pode ter muitas *Inscrições*;
- Linha verde: uma *Inscrição* de um *Corredor* deve estar relacionada a uma *Modalidade*, e uma *Modalidade* pode ter muitas *Inscrições*.

As classes projetadas devem ser utilizadas para construir um conjunto de telas que permitam a criação, listagem, atualização e remoção (CRUD) de objetos referentes a cada uma das classes. O uso de banco de dados é opcional. Seu sistema deve possuir um menu no topo da página, permitindo com que o usuário acesse todas as opções de gerenciamento das classes do sistema.

Por exemplo, a Figura 2 mostra a interface inicial do programa de gerenciamento de corridas de rua.

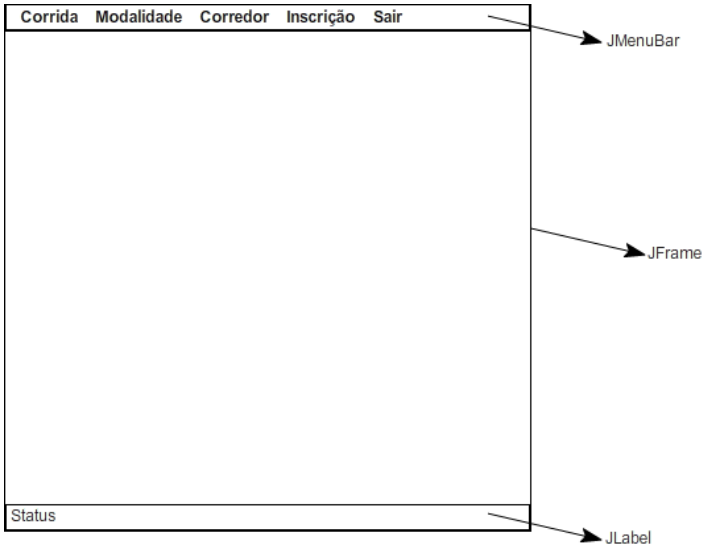

Figure 2: Tela inicial do sistema de gerenciamento de corridas de rua.

Nesse exemplo, a janela principal do sistema foi implementada utilizando a classe *JFrame*. Há um menu no topo da página, permitindo com que o usuário acesse todas as opções de gerenciamento de todas as entidades do sistema. Esse menu foi implementado utilizando a classe *JMenuBar*. Além disso, há um *JLabel* no rodapé da página, informando o status de todas as operações requisitadas no sistema.

Ao clicar no botão *Corrida*, as opções de *Criar uma corrida*, *Listas as corridas*, *Atualizar uma corrida* e *Remover uma corrida* são fornecidas para o usuário. Esse mesmo conjunto de opções também deve ser fornecido para os menus *Modalidade*, *Corredor* e *Inscrição*. A implementação de vocês deve seguir o mesmo padrão de interface gráfica exemplificado na Figura 2.

1 Entrega

O trabalho deve ser feito em grupos de no mínimo *1* e no máximo *2* alunos, e deve ser entregue via AVA impreterivelmente até às *23h55min* do dia *28/06/2022*. Entregue apenas um arquivo *zip* contendo todas as classes desenvolvidas, juntamente com um relatório (PDF) que explique o seu problema e as principais classes envolvidas.

2 Detalhes Adicionais

Vocês podem escolher qualquer editor de texto ou IDE para desenvolver o trabalho, desde que não sejam utilizados plugins de construção automática de tela. Trabalhos que utilizem esse tipo de plugin receberão

nota zero.

O problema escolhido deve ser diferente do problema apresentado na Figura 1.

3 Critérios de Correção

Os principais critérios de correção adotados no trabalho serão:

- Entrevista com os alunos;
- Completude do trabalho;
- Conceitos de orientação a objetos implementados corretamente;
- Organização e clareza do código.

Cada um dos critérios de correção acima conterão subcritérios a serem definidos pelo professor posteriormente.