Linguagens de Programação Orientada a Objetos - Trabalho Prático

Prof. Lucas Reis October 2, 2021

Informações Gerais

O objetivo deste trabalho é estimular a sedimentação dos conceitos adquiridos durante a disciplina, além de avaliar as capacidades do aluno de implementar um projeto em orientação a objetos.

Os trabalhos serão realizados **em dupla** e terão tema livre. Isso significa que cada dupla será responsável por sugerir o tema que achar mais interessante de desenvolver, contanto que possua um nível mínimo de dificuldade e cujo desenvolvimento esteja limitado ao escopo da disciplina (não escolham temas muito simples ou muito complexos).

Primeira fase e sugestões de temas

A primeira fase consiste no envio do tema e das duplas que realizarão o trabalho. Além de sugerir o próprio tema, a dupla também tem a opção de escolher entre uma das sugestões apresentadas abaixo:

I. Sistema de um banco (tipos de conta, cliente, gerente, operações como depósito, extração, rendimento,...)

II. Rede social de publicações em texto (usuários, postagens, amigos, mensagens privadas, feed de postagens, ...)

III. Sistema de mercado (produtos, descontos, funcionários,... Ações como registrar produtos, debitar de estoque, ...)

IV. Um sistema de RPG (jogadores, inimigos, classes, raças, itens, mapas, ... Ações como batalhar, explorar, obter recompensas, ...)

Considerem os temas sugeridos como guias de como estruturar o seu próprio tema, caso a dupla escolha realizar tema livre.

O envio das duplas e dos temas deverá ser feito por meio de um arquivo `txt` no AVA até o último minuto do segundo domingo de outubro (10/10 - 23:59).

Segunda fase e avaliação

A segunda fase consiste na implementação do projeto, utilizando dos conceitos vistos durante o período de aulas da disciplina.

A lista de conceitos de orientação a objetos que serão cobrados no trabalho será:

- Herança - Obrigatório (2 pontos)
- Polimorfismo dinâmico - Obrigatório (2 pontos)
- Abstração **ou** polimorfismo estático - Obrigatório (2 pontos)
- Encapsulamento das classes (uso correto de modificadores de acesso) - Obrigatório (2 pontos)
- Construtores, estrutura do código e execução (classe Main) - Obrigatório (1 ponto)
- Tratamento de exceções - Obrigatório (1 ponto)
- Interface Gráfica - Opcional (1 ponto)

A quantidade pontuada também é proporcional a complexidade do trabalho. Isso implica que trabalhos muitos simples terão baixa pontuação nas categorias de implementação de cada tópico. A atribuição livre dos temas tem propósito de estimular vocês a serem criativos para desenvolver os ambientes, e exercitar a capacidade de transformar conceitos abstratos em código usando o aprendizado em orientação a objetos. Portanto, evite desenvolver um trabalho que implemente apenas um curto conjunto de classes para cumprir cota!

O envio da segunda fase deve ser composto de um arquivo `zip` contendo o código fonte de todas as classes do projeto, incluindo um arquivo em PDF relatando o nome dos alunos da dupla e uma descrição de uso do projeto, incluindo um diagrama de entidade-relacionamento das classes do projeto, além de exemplos de casos de teste usados para verificar o funcionamento do código. Caso a interface gráfica tenha sido implementada, também devem ser enviado um link para um vídeo exemplificando o uso da interface e seus componentes[1].

O envio segunda etapa deverá ser feito no AVA até o último minuto do dia 24 de novembro (24/11 - 25:59).

Exemplo de diagrama Entidade-Relacionamento

A figura 1 representa um diagrama entidade-relacionamento do conjunto de classes envolvidas em um sistema dedicao à organização de corridas de rua. Neste exemplo, existem 4 entidades (classes) e 4 relacionamentos entre elas. Note que os atributos de cada classe também estão especificados no diagrama.

[1] Você pode usar o Google Meet com sua conta institucional para gravar uma apresentação, que ficará armazenada no seu Drive

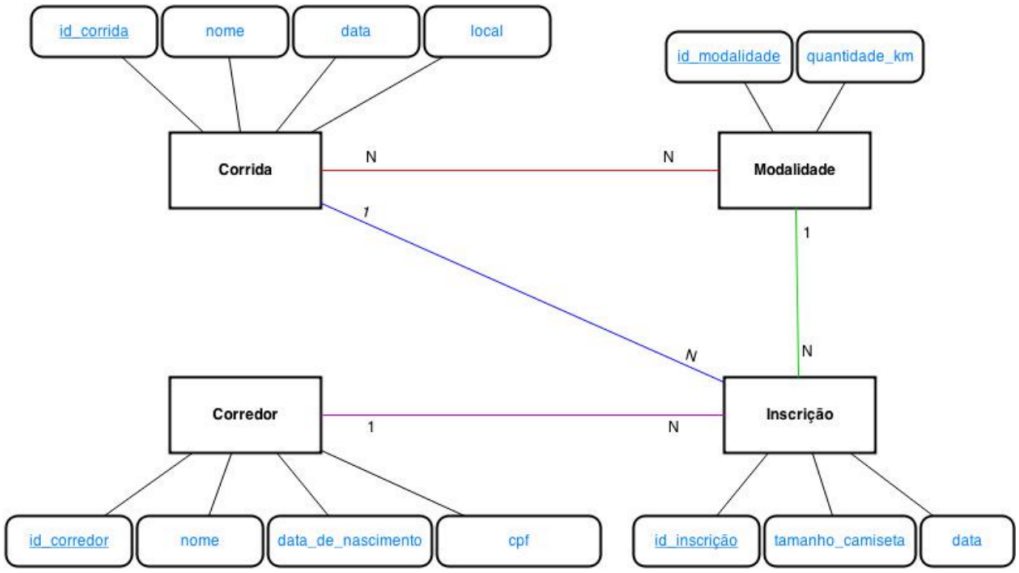

Figure 1: Exemplo de diagrama entidade-relacionamento

Neste exemplo, as quatro classes elencadas são:

- Corrida - tem atributos *id_corrida*, *nome*, *data* e *local*;
- Modalidade - tem atributos *id_modalidade* e *quantidade_km*;
- Corredor - tem atributos *id_corredor*, *nome*, *data_de_nascimento* e *cpf*;
- Inscrição - tem atributos *id_inscricao*, *tamanho_camiseta* e *data*.

Atente-se aos relacionamentos entre as classes:

- Linha vermelha: cada *corrida* pode ter `N` modalidades, e cada *modalidade* pode fazer parte de `N` *corridas* (uma mesma modalidade pode estar presente em diferentes corridas);
- Linha rosa: Cada *inscrição* só faz parte de um *corredor*, mas cada *corredor* pode realizar `N` *inscrições* (pode participar de várias corridas diferentes);
- Linha azul: De maneira similar, cada *corrida* é composta de `N` *inscrições* (as inscrições de cada corredor), mas cada *inscrição* faz parte de uma única *corrida*;
- Linha verde: Uma *inscrição* de *corredor* deve estar relacionada a uma *modalidade*, e cada *modalidade* pode ter várias *inscrições*.

Para herança, utilize uma seta (da classe pai para a filha). Para sinalizar abstração, utilize uma coloração específica (sinalizada no seu relatório). A figura 2 exemplifica um diagrama simples que utiliza conceitos de herança e abstração. Nesse exemplo, a cor amarela foi usada para sinalizar uma classe abstrata (*animal*).

Qualquer ferramenta de desenho de imagens vetoriais pode ser usada para projetar um diagrama entidade-relacionamento. Inkscape ou o Google Drawings são ótimos exemplos gratuitos.

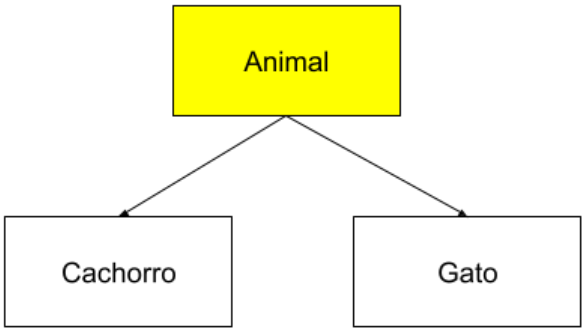

Figure 2: Uso de herança e abstração no diagrama

Plágio

Qualquer sinal de plágio entre duas duplas acarreta em **ZERO** para ambos os trabalhos, sem possibilidade de reconsideração. Trabalhos copiados da internet também implicam no mesmo resultado. Evitem ao máximo compartilhar seu código com outros alunos que não estejam presentes na sua dupla!