Instituto Federal de Santa Catarina - Câmpus Gaspar

Análise e Desenvolvimento de Sistemas

Programação Orientada a Objetos

Prof. Dr. Renato Simões Moreira

Avaliação Prática

GELASIO EBEL JUNIOR

OSMAR BARBOSA GOMES DA CONCEIÇÃO

APLICANDO ORIENTAÇÃO A OBJETOS EM UMA SIMPLES APLICAÇÃO DE FÓRMULA 1.

Desenvolvemos uma aplicação que simula uma competição de Fórmula 1 ou um campeonato de cinco etapas, utilizando o que aprendemos sobre orientação a objetos. Nossa aplicação cria automaticamente três equipes e seis pilotos, sendo dois carros por equipe, além de atribuir dois pilotos para cada equipe, cada um em seus respectivos carros.

Principais Entidades:

A classe Participante é uma entidade abstrata que serve como base para as filhas que compartilham atributos comuns, como nome, pais, pontos e vitórias. Esta classe é estendida por Piloto e Equipe, pois ambas são competidores, mas um piloto pode ser campeão mesmo a equipe não sendo, difícil, mas pode acontecer pela pontuação do colega de equipe. Nesse caso, vimos utilidade em aplicarmos o que aprendemos sobre herança.

Piloto e equipe e carro são objetos criados automaticamente na opção 1 disponível apenas na inicialização do programa. Esses três objetos implementam uma interface Exibir Detalhes que é comum a todos e chamamos por esse método na classe final Menu Interativo.

Equipe representa as equipes de Fórmula 1, e sempre tem dois pilotos e dois carros. A classe Equipe herda de participante, além disso, quando um objeto de equipe é criado, ele inicia uma reação em cadeia construindo seus carros, e chama também a criação de pilotos. Nosso principal objetivo foi demonstrar a complexidade e profundidade que a orientação a objeto oferece.

A classe Carro tem relação com uma Equipe e um Piloto. Cada objeto tem um modelo específico /A ou /B, pertence a uma equipe e possui um piloto.

Cadastramento automático criamos para automatizar a criação dos objetos, cuidados ao máximo para que as equipes tenham seus pilotos e carros corretamente

configurados antes do início das corridas. Ela simplifica a inicialização do sistema porque ninguém aguenta Scanners e ficar digitando nome o tempo todo.

Menu Interativo serve apenas para mostrar o menu do sistema e chamar os métodos que fazem a engrenagem do programa funcionar.

Ordenar ordena porque foi pedido no exercício, mas tivemos dificuldade em entender como fazer funcionar os imports que o IntelliJ queria e tivemos que pedir ajuda ao GPT para fazer funcionar, mas ficamos devendo nesse quesito de saber aplicar esses imports com Collections e Comparators.

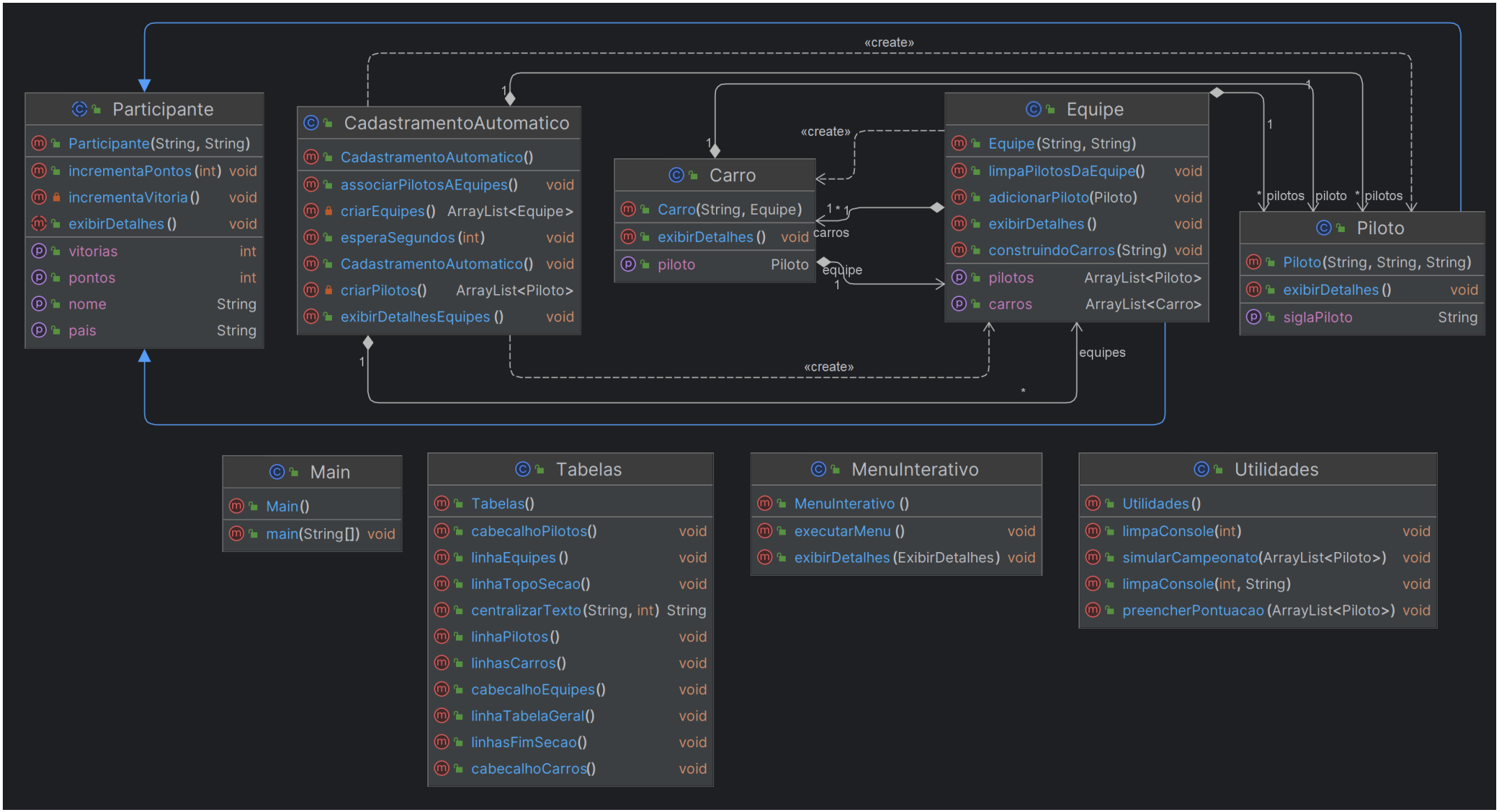
A classe Tabelas é utilizada para formatar e exibir dados de forma organizada no consolep ara diferenciar quando foi usado a Interface Exibir Detalhes. Deu bastante trabalho, mas ficou show de bola, tabulação e exibição de tabelas no console em ASCII (opcao 2) .

A classe Utilidades tem métodos auxiliares que facilitam a vida, como a pausa para não correr o texto muito rápido entre exibições e a centralização de texto. Ela pretende organizar o output e prevenir duplicações de códigos simples mas que se repetem entre os objetos.

Tentamos usar de tudo um pouco do que aprendemos na sala de aula, sem "forçar" a orientação a objeto, mas sim fazer uso quando era plausível sua utilidade, como no caso de Participante, ou de uma classe com métodos que se repetem e podem ser usados em múltiplas classes, no caso de Utilidades. E automatizamos o máximo que deu para testes e não ficar perdendo tempo digitando (nome, enter, sobrenome, enter) e focar no que precisa foi aprendido. Estamos bem felizes com o resultado, mas devido a ter sido feito em dupla e dois cara, que é difícil sobrar tempo, ficou apurado, dava para melhorar, por exemplo, queríamos incluir uma classe Circuito e GranPrix e Temporada, mas tivemos que cortar para poder finalizar com sucesso e funcionando.

Diagrama de Classes está na próxima pagina.

Tambem anexamos um PNG do diagrama de classes em alta resolução na pasta src no arquivo zipado enviado.


«create»
Participante
Participante(String, String)
incrementaPontos(int) void
incrementaVitoria() void
exibirDetalhes() void
vitorias int
pontos int
nome String
pais String
CadastramentoAutomatico
CadastramentoAutomatico()
associarPilotosAEquipes() void
criarEquipes() ArrayList<Equipe>
esperaSegundos(int) void
CadastramentoAutomatico() void
criarPilotos() ArrayList<Piloto>
exibirDetalhesEquipes() void
Carro
Carro(String, Equipe)
exibirDetalhes() void
piloto Piloto
Equipe
Equipe(String, String)
limpaPilotosDaEquipe() void
adicionarPiloto(Piloto) void
exibirDetalhes() void
construindoCarros(String) void
pilotos ArrayList<Piloto>
carros ArrayList<Carro>
Piloto
Piloto(String, String, String)
exibirDetalhes() void
siglaPiloto String
1 * 1
carros
equipe
1
pilotos
piloto
equipes
*
Main
Main()
main(String[]) void
Tabelas
Tabelas()
cabecalhoPilotos() void
linhaEquipes() void
linhaTopoSecao() void
centralizarTexto(String, int) String
linhaPilotos() void
linhasCarros() void
cabecalhoEquipes() void
linhaTabelaGeral() void
linhasFimSecao() void
cabecalhoCarros() void
MenuInterativo
MenuInterativo()
executarMenu() void
exibirDetalhes(ExibirDetalhes) void
Utilidades
Utilidades()
limpaConsole(int) void
simularCampeonato(ArrayList<Piloto>) void
limpaConsole(int, String) void
preencherPontuacao(ArrayList<Piloto>) void