Instituto Superior de Engenharia de

Coimbra



Departamento de Engenharia Informática e de Sistemas

Engenharia Informática

Programação Orientada a Objetos 2019/2020

Simulador Corridas de Carros

C++

Trabalho Prático – Checkpoint 2

Diogo Marques- 21240125

Miguel Abrantes – 21240373

1. **Quais foram as classes consideradas na primeira versão da aplicação que foi testada?**

A primeira versão tinha as seguintes classes:

* + - Carro
    - Piloto

Estas classes e funcionalidades foram testadas com uma simples função main.

1. **Quais os conceitos/classe que identificou ao ler o enunciado?**

Os conceitos identificados foram:

* Corrida
* Carro
* Piloto (O conceito de polimorfismo devido à existência de vários Pilotos – Não implementado para a meta 1)
* Autódromo
* Pista
* Garagem
* Campeonato
* DVG (Classe abstrata que contém todos os tipos de objetos do simulador).

1. **Relativamente a duas das principais classes da aplicação, identifique em que classes ou partes do programa são criados, armazenados e destruídos os seus objetos.**

Autódromo: contém os objetos Pista e Garagem que são criados localmente, assim como destruídos pela mesma.

Equipa: contém ponteiros para os Pilotos e Carros que estejam a agir como uma equipa, ou seja, que o piloto esteja no carro. Ambos os seus objetos são criados e eliminados na classe DVG, independentemente da classe Equipa.

1. **Indique um exemplo de uma responsabilidade atribuída a uma classe que esteja de acordo com a orientação dada acerca de *Encapsulamento*.**

A responsabilidade “criar equipa” está atribuída à classe DVG, porque tem a acesso a todos os pilotos e carros

1. **De entre as classes que fez, escolha duas e justifique por que considera que são classes com objetivo focado, coeso e sem dispersão.**

* Classe Autódromo: tem dados e responsabilidades relativos apenas as suas variáveis (pista garagem), como por exemplo adicionar carros na sua garagem e saber quais os carros que se encontram na pista.
* Classe Carro: tem dados e responsabilidades relativos apenas aos carros, como saber se o carro está em movimento ou não, a sua velocidade entre outros.

1. **Relativamente à aplicação entregue, quais as classes que considera com responsabilidades de interface com o utilizador e quais as que representam a lógica?**

* Responsabilidade de interface: Interface
* Responsabilidade da lógica da aplicação: Autodromo, Pista, Garagem, Carro, Piloto, Jogo e DVG.

1. **Identifique o primeiro objeto para além da camada de iteração com o utilizador que recebe e coordena uma funcionalidade de natureza lógica?**

* As ordens vindas da camada de interação com o utilizador são recebidas e processadas por um objeto da classe Jogo.

1. **A classe que representa a envolvente de toda a lógica executa em pormenor muitas funcionalidades, ou delega noutras classes? Indique um exemplo em que esta classe delega uma funcionalidade noutra classe.**

* A classe Jogo representa a envolvente de toda a lógica. É a classe que vai ser utilizada por exemplo para ler os ficheiros e também delega funcionalidades noutras classes, como por exemplo para obter a listagem de todas as entidades existentes, esta vai “pedir” à classe DVG.

1. **Apresente as principais classes da aplicação através da seguinte informação:**

Nota: Acrónimo CRUD corresponde a Create/Read/Use/Delete, ou seja, Criar/Ler/Usar/Eliminar.

**Classe:** Piloto

**Responsabilidades:**

* Verificar se existe repetição do nome na sua criação e ajusta conforme o enunciado.
* Obter e alterar informações relativas à equipa.
* Aceder aos controlos do carro que estiver a conduzir.

**Colaborações:** Carro

**Classe:** Carro

**Responsabilidades:**

* Atribuir ID automático a um carro novo
* Obter informações sobre o carro (Vel. máxima, cap. da bateria, etc.).
* Obter e alterar informações relativas à equipa.
* Alterar o seu estado (iniciar movimento, parar movimento, carregar bateria, etc.).

**Colaborações:**

**Classe:** Pista

**Responsabilidades:**

* Iniciar/Terminar corrida.
* Inserir/Remover Equipas da pista.
* Passar tempo para cada equipa na pista.

**Colaborações:** Equipa, Autódromo

**Classe:** Garagem

**Responsabilidades:**

* Armazenar as equipas que se encontram aptas para competir
* Carregar as baterias dos carros armazenados
* Retirar uma equipa apta para competir na pista

**Colaborações:** Equipa, Autódromo

**Classe:** Autódromo

**Responsabilidades:**

* Verificar a repetição de nomes na própria classe e corrigir de acordo com o enunciado.
* Retirar uma equipa da garagem e inserir na pista (e vice-versa).

**Colaborações:** Pista/Garagem/Equipa

**Classe:** Interface

**Responsabilidades:**

* Recebe e interpreta comandos do utilizador, transformando-os em ações especificas que a classe jogo possa utilizar.

**Colaborações:** Jogo

**Classe:** Campeonato

**Responsabilidades:**

* Armazenar os campeonatos em competição.
* Alterar o campeonato no qual irá a decorrer a corrida.
* Terminar o campeonato.

**Colaborações:** Autódromo

**Classe:** Jogo

**Responsabilidades:**

* Responsável por ler ficheiros de texto com informação a carregar.
* Gerir Campeonato (CRUD)
* Gerir Autódromos (CRUD)
* Gerir DVG (CRUD)
* Gerir Campeonato (CRUD)

**Colaborações:** DVG, Autódromo, Campeonato

**Classe:** DVG

**Responsabilidades:**

* Armazenar o carros, pilotos e equipas criadas.
* Gerir Equipas (CRUD).
* Gerir Carros (CRUD).
* Gerir Pilotos (CRUD).

**Colaborações:** Carro, Piloto, Equipa

**Classe:** Equipa

**Responsabilidades:**

* Armazenar o ponteiro para o respetivo carro e piloto
* Passar tempo para cada equipa, ou seja, permite que o carro e piloto ajam conforme as suas regras.

**Colaborações:** Carro, Piloto

Funcionalidades Implementadas

Comandos em funcionamento da aplicação

Antes de dar início à simulação em si, o jogo deve ser configurado. Sendo assim existem comandos específicos para o fazer e que serão descritos de seguida:

* Carregar elementos participantes na simulação por ficheiro.
  + carregaC
  + carregaP
  + carregaA

O comando carrega% serve para carregar todos os carros, pilotos e autódromos, respetivamente, presentes no jogo. Exemplo de uso: *carregaC carros.txt* (vai carregar todos os carros presentes no ficheiro, para o jogo).

* Criar elementos participantes na simulação por consola:
  + cria

O comando cria serve para criar carros, pilotos e autódromos pela consola. Exemplo de uso: *cria p crazy Diogo (*Vai criar um piloto do tipo crazy chamado Diogo).

* apaga

O comando apaga serve para apagar carros, pilotos e autódromos pela consola. Exemplo de uso: *apaga p Diogo* (Vai apagar o piloto com o nome Diogo).

* Entranocarro

O comando entranocarro serve para associar a um carro o respetivo piloto. Exemplo de uso: *entranocarro a Diogo* ().