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 - P1 - 21240125/2014015208
Miguel Abrantes - P2 - 21240373/2014008589

Introdução

Este trabalho foi realizado no âmbito da unidade curricular de programação orientada a objetos e consiste na criação de um simulador de corridas de carros.

No jogo irão existir autódromos que irão contar uma pista e uma garagem respetivamente. É aqui que as corridas propriamente ditas se irão realizar sendo que os carros que a pista consegue comportar são movidos para lá aquando da corrida e os restantes ficam na garagem.

O simulador tem pistas de vários tamanhos, carros com diferentes capacidades de bateria, assim como diferentes pilotos. Dependendo do tipo de piloto que esteja a conduzir, a sua reação na pista irá ser também diferente.

No desenrolar do jogo os pilotos irão competir em vários autódromos, e ganhando pontos de acordo com a sua classificação. A simulação termina quando não existirem mais autódromos para competir, ou seja, o campeonato acabou.

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.

2. **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).

3. **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.

   DGV: contém armazenados todos os carros e pilotos antes de ser iniciada uma corrida. É possível fazer cópias de segurança, o que torna esta classe muito útil.

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

   A responsabilidade de carregar todas as baterias está entregue à classe *Garagem*, pois esta tem acesso a todos os carros inseridos na mesma.

5. **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.

6. **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: Autódromo, Pista, Garagem, Carro, Piloto, Jogo e DVG.

7. **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.

8. **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.

9. **Dê um exemplo de uma funcionalidade que varia conforme o tipo do objeto que a invoca. Indique em que classes e métodos está implementada esta funcionalidade.**
    - Inicialmente a simulação era apenas formada por um tipo de piloto. Tornou-se necessário considerar outros tipos de pilotos. A simulação deverá poder integrar pilotos do tipo "Rápido", "Crazy" e "Surpresa".
    - Cada tipo de piloto tem uma reação na pista diferente dos restantes.

```
class Piloto
{
	static std::vector<std::string> todosNomesPilotos;
	std::string nome;
	int pontuacao;
	char idCarro;
…

class PilotoRapido : public Piloto
{
	int tempoPassado;
…

class PilotoSurpresa : public Piloto
{
public:
…

class CrazyDriver : public Piloto
{
	int iniciaCorridaApos;
	int pos;
…
```

10. 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 informações relativas ao piloto.
- Aceder aos controlos do carro que estiver a conduzir.

Colaborações: Carro, Pista

Classe: Carro
Responsabilidades:
- Atribuir ID automático a um carro novo
- Obter informações sobre o carro (Vel. máxima, cap. da bateria, etc.).
- Alterar o seu estado (iniciar movimento, parar movimento, carregar bateria, etc.).

Colaborações: Piloto, Pista

Classe: Pista
Responsabilidades:
- Iniciar/Terminar corrida.
- Inserir/Remover carros da pista.
- Passar tempo para cada elemento na pista.
- Desenhar Pista na consola

Colaborações: Carro, Autódromo, Consola

Classe: Garagem
Responsabilidades:
- Armazenar os carros que se encontram aptos para competir
- Carregar as baterias dos carros armazenados
- Listar informações dos carros lá inseridos

Colaborações: Carro, Autódromo

Classe: Autódromo
Responsabilidades:
- Verificar a repetição de nomes na própria classe e corrigir de acordo com o enunciado.
- "Mandar" carros para a garagem e consequentemente, "mandar" retirar da garagem e inserir na pista, para a corrida.

Colaborações: Pista/Garagem

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 os carros e os pilotos.
- Gerir Carros (CRUD).
- Gerir Pilotos (CRUD).

Colaborações: Carro, Piloto

Funcionalidades Implementadas

Comandos em funcionamento da aplicação

Antes de dar início à simulação em si, o jogo deve ser configurado. Existem dois modos no jogo, o modo 1 que é de configuração e o modo 2 que é mais ligado à simulação da corrida em si. 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* (Vai fazer com que o piloto chamado Diogo entre no carro identificado pela letra a).

- Saidocarro

O comando 'saidocarro' serve para retirar um piloto de um carro. Exemplo de uso: *saidocarro a* (Vai fazer com que o piloto que esteja dentro do carro identificado pela letra a, saia do carro).

- Lista

O comando 'lista' serve para mostrar toda a informação relativa aos carros, pilotos e aos autódromos, bem como que equipas estão formadas (pelo comando '*entranocarro'*). Exemplo de uso: *lista* (Vai mostrar todas as informações descritas anteriormente).

- Savedgv

Este comando tem como objetivo fazer backup da dgv atual, antes de ser iniciado um campeonato, realizando uma duplicação de todos os dados. Deste modo são guardadas todas as informações relativas aos carros/pilotos caso haja algum problema, ou uma simulação destrua os mesmos. Podem ser feitas inúmeros backups, desde que todos tenham um nome diferente. Exemplo de uso: *savedgv coimbra* (Vai criar uma cópia do dgv atual com o nome "coimbra").

- Loaddgv

Permite utilizar os backups da dgv, realizados com o comando anterior. Não elimina apenas a dgv original, mas mantem o registo de todas os backups de de dgv's criados. Exemplo de uso: *loaddgv coimbra* (Vai alterar a dgv em utilização para a dgv com o nome "coimbra").

- Deldgv

Apaga a cópia do objeto dgv, que se encontra mantido em memória. Exemplo de uso: *deldgv coimbra* (Vai apagar a cópia da dgv com o nome “coimbra” armazenada em memória).

- Campeonato

O comando ‘campeonato’ é o responsável por iniciar o modo 2 do jogo. Este comando serve para iniciar um campeonato pelos variados autódromos do jogo. Exemplo de uso: *campeonato A1 A2 A3 A4* (Vai iniciar um campeonato, sendo a primeira corrida no Autódromo 1, seguindo-se o A2, A3 e por fim A4).

- Corrida

O comando ‘corrida’ é o responsável por iniciar uma nova corrida, seja ela o início de um novo campeonato, ou início de uma nova corrida num novo autódromo depois de um campeonato já estar iniciado. Exemplo de uso: *corrida* (Uma nova corrida vai começar).

- Listacarros

O comando ‘*listacarros’* é o responsável por listar todos os carros inseridos na garagem do autódromo em competição. Exemplo de uso: *listacarros* (Vai listar todos os carros, e respetivas informações, que estiverem inseridos na garagem do autódromo em competição).

- Carregabat

O comando ‘carregabat’ é o responsável por carregar a um valor da bateria de um carro à escolha. Exemplo de uso: *carregabat a 10* (Vai carregar em 10 mAh a bateria do carro a).

- Carregatudo

O comando ‘carregatudo’ é o responsável por carregar a bateria de todos os carros. Exemplo de uso: *carregatudo* (Vai carregar a bateria de todos os carros na garagem).

- Acidente

O comando ‘acidente ‘é o responsável por causar um dano irreparável num carro. Exemplo de uso: *acidente a* (Vai causar um dano irreparável no carro identificado pela letra a).

- Stop

O comando ‘stop’ é o responsável por fazer parar um piloto durante uma corrida. Exemplo de uso: *stop diogo* (Vai fazer com que o piloto Diogo ative o sinal de emergência e saia da corrida).

- Destroi

O comando ‘destrói ‘é responsável por eliminar um carro da simulação por completo. Exemplo de uso: *destrói a* (Vai eliminar o carro identificado pela letra a e caso este carro tenha algum piloto lá dentro, o piloto fica apeado).

- Pontos

O comando ‘pontos’ é o responsável por mostrar a pontuação de um campeonato existente. Exemplo de uso: *pontos* (É mostrada a classificação do campeonato em vigor).

- Passatempo

O comando ‘passatempo’ é o responsável por fazer avançar a simulação. Exemplo de uso: *passatempo 10* (Vai fazer avançar a simulação em 10 segundos).

- Log

O comando ‘log’ é o responsável por mostrar o log de comandos que o simulador tem registado. Exemplo de uso: *log* (Vai mostrar todos os comandos inseridos pelo utilizador).

Existem também comandos que são comuns aos dois modos de jogo. Nestes, encontram-se os seguintes:

- Help

O comando ‘help’ serve para dar indicações ao utilizador de todos os comandos que este pode utilizar. Exemplo de uso: *help* (Vai mostrar no ecrã todos os comandos possíveis de serem utilizados na simulação).

- Clear

O comando ‘clear’ serve para apagar todas as informações que estejam na consola. Exemplo de uso: *clear* (Vai apagar tudo o que estiver no ecrã).

- Fim

O comando ‘fim’, tal como o nome indica fecha a simulação. Exemplo de uso: *fim* (Vai terminar a simulação).