2018/19

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

RELATÓRIO DO 1º TRABALHO

ALGORITMOS E ESTRUTURAS DE DADOS

Biblioteca de jogos

TURMA 6 – GRUPO 7

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

BRUNO FILIPE CARDOSO MICAELO	UP201706044
MIGUEL DELGADO PINTO	UP201706156
NUNO MIGUEL TEIXEIRA CARDOSO	UP201706162

Índice

Tema de trabalho

O objetivo geral do primeiro trabalho da unidade curricular era a criação de uma plataforma de dados e de gestão de jogos de vídeo, recorrendo ao paradigma da orientação por objetos e usar a linguagem C++ para implementar a solução.

O sistema de gestão implementado possui um conjunto de utilizadores, sendo que cada utilizador tem disponível uma biblioteca de jogos. Existem dois tipos principais de jogo: *Home* e *Online*.

Solução Implementada

O sistema de bibliotecas de jogo foi implementado através da criação da classe-mestre ***System***, que contém uma loja para a compra de jogos e guarda informação sobre todos os seus utilizadores.

A loja consiste num vetor de apontadores para objetos da classe ***Game*****.** Cada jogo tem um nome e um ID (número de identificação). Possui um preço base que poderá ser alterado através de descontos temporários ou mudança permanente de preço. Cada jogo pertence a pelo menos um género.

Existem dois grupos gerais de jogos: ***Home*** e ***Online***. Os jogos *Home* caracterizam-se por apenas ter modo *singleplayer* e *local co-op*, pelo que podem ser jogados sem qualquer conexão à internet. Como tal, possuem um vetor de *updates*, de modo a consertar possíveis bugs. Os jogos *Online* podem ser de dois tipos: ***VariableSubsc*** (subscrição variável) e ***FixedSubsc*** (subscrição fixa).

De modo a conseguir comprar jogos, e como se trata de uma biblioteca online, o utilizador, objeto da classe ***User***, possui pelo menos um cartão bancário, objeto da classe ***Card***. Esse cartão, para ser utilizável, deve ter um número válido. Como solução, foi implementado o **algoritmo de Luhn** para a validação de números de cartões.

O utilizador, quando decide começar uma sessão de jogos, instancia um objeto da classe ***PlaySession***. Este objeto guarda informação sobre a sessão, como o tempo jogado, o jogo jogado, a data e em que plataforma é que esta ocorreu (como por exemplo, Playsation 4, Nintendo ou Dreamcast).

Classes Implementadas

- ## System

Trata-se da "classe-mestre" do programa. Contém um vetor de apontadores para objetos da classe *Game* (corresponde à loja de jogos) e um vetor de apontadores para objetos da classe *User* (utilizadores do sistema).

Através desta classe é possível adicionar jogos e utilizadores, procurar jogos e utilizadores e ordenar os vetores referidos anteriormente através de diversos parâmetros. Estão também implementadas funções de restrição na procura de jogos, o que se revela uma grande ajuda quando se pretende procurar jogos em bibliotecas de grande dimensão.

- ## Game

A classe *Game* corresponde a um determinado jogo de vídeo. Especifica várias caraterísticas de cada jogo: nome, número de identificação (ID), número de utilizadores, preço base e preço atual (após desconto), data de lançamento, intervalos de idades recomendadas (sendo o limite inferior obrigatório para poder jogar) e a empresa que desenvolveu o jogo.

Possui também em vetores: conjunto de plataformas compatíveis para jogar, conjuntos de géneros aos quais pertence e historial de preço.

- ## Home

Classe derivada de *Game*. Corresponde a um jogo que apenas tem modo *singleplayer* ou *multiplayer* local. Possui um vetor de datas (elementos da classe *Date*), que simbolizam *updates* a instalar.

- ## Online

Classe derivada de *Game*. Corresponde a um jogo que pode ser jogado online. Contém o tempo total jogado por todos os utilizadores, para fins estatísticos.

- ## FixedSubsc

Classe derivada de *Online*. Corresponde a um jogo *Online* com subscrição fixa, paga em intervalos de tempo regulares.

- ## VariableSubsc

Classe derivada de *Online*. Corresponde a um jogo *Online* com subscrição variável: o valor pago depende do tempo total jogado.

- ## User

Corresponde a um utilizador do sistema. Cada utilizador é caracterizado pelo nome, email, idade e morada.

Guardados em vetores, os utilizadores podem ter vários cartões bancários , possuem uma coleção de jogos (vetor de apontadores para objetos *Game*) e um registo de sessões de jogo (vetor de apontadores para objetos *PlaySession*).

- ## PlaySession

Corresponde a uma sessão de jogo, caracterizada pela data em que ocorreu, a duração em horas, a plataforma em que foi realizada, o utilizador que a realizou e o jogo jogado.

- ## Card

Corresponde ao cartão bancário de cada utilizado. Possui um número, que apenas é valido se for aprovado pelo algoritmo de Luhn. Cada cartão tem um determinado balanço (que pode ser incrementado).

- ## Date

Classe com múltiplos propósitos, com o objetivo de representar uma data, no formato DD/MM/AAAA.

- ## Interval

Classe que representa um intervalo, usado para as idades recomendadas de cada jogo.

- ## Exception

Classe abstrata que inclui diversas classes derivada, com o objetivo de controlar determinados erros e exceções. As classes derivadas são: *InvalidDate*, *RepeatedGame*, *RepeatedCard*, *RepeatedUser*, *NonExistingGame*, *NonExistingUser*, *InvalidCard*, *InvalidComparer*, *InvalidRestrictor*, *InvalidArgument*, *UserTooYoung*, *InvalidCard, GameNotOwned, GameAlreadyOwned* e *NotEnoughFunds*.

Diagrama UML

Devido às grandes dimensões do diagrama UML do projeto, este será enviado em anexo, no formado pdf.

Casos de Utilização

Após a execução do programa, o utilizador depara-se com um menu de **três opções principais**, sendo as duas primeiras correspondentes à **criação e leitura de sistemas de bases de dados de bibliotecas de jogos**, respetivamente. Para **encerrar o programa**, recorre-se à terceira opção.

Na primeira opção, dá-se ao utilizador a possibilidade de criar e editar duas bibliotecas: a biblioteca de jogos (*store*) e a biblioteca de utilizadores (*user_library*). Para tal, é apresentado ao utilizador um novo menu de nove opções. Deste modo, o utilizador pode:

- Adicionar um novo jogo ao sistema, inserindo sucessivamente as especificações a ele subjacentes (tipo, título, preço, data de lançamento, faixa etária, plataformas disponíveis, géneros e criador), podendo, posteriormente, editar esta informação;
- Adicionar um novo utilizador ao sistema, inserindo sucessivamente as especificações a ele subjacentes (nome, email, idade e morada). Para cada utilizador criado será, seguidamente, possível adicionar cartões bancários à sua conta, comprar jogos para a sua biblioteca, instalar atualizações e adicionar sessões de jogo;
- Ordenar a biblioteca de jogos do sistema por ordem ascendente ou descendente segundo diferentes tipos de comparação de objetos (ID, título, preço, preço base, data de lançamento, faixa etária ou criador);
- Ordenar a biblioteca de utilizadores do sistema por ordem ascendente ou descendente segundo diferentes tipos de comparação de objetos (nome, idade);
- As restantes opções correspondem a pesquisas por jogo ou por utilizador, seja por modo direto ou através de *tag words*.

A **segunda opção** deixa o utilizador carregar um sistema de bibliotecas criado anteriormente, possibilitando a continuação da sua edição.

Principais dificuldades

Uma das maiores dificuldades sentidas pelo grupo foi, sem dúvida, a gestão do tempo.

Quanto aos detalhes técnicos relativos ao projeto e implementação de código, não foram sentidas nenhumas dificuldades relevantes. No entanto, a implementação da interface foi muito trabalhosa, tal como o input de ficheiros de texto.

Conceitos Utilizados

Foram implementados diversos conceitos discutidos ao longo das aulas teóricas e práticas:

- Classes
- Classes abstratas
- Membros *static*
- Herança e Polimorfismo
- Tratamento de exceções
- *Overload* de operadores
- Classes e funções *template*
- Algoritmos de procura e ordenação (criados pelo grupo)
- Estruturas lineares
- Tratamento de ficheiros de texto

Empenho dos participantes

É seguro afirmar que o trabalho esteve distribuído uniformemente pelos três membros do grupo. Todos os membros participaram na implementação e projeto de todas as classes.

Foram realizadas reuniões de grupo, com objetivo de orientar, planear e desenvolver o projeto. Houve, de igual modo, bastante trabalho individual, fundamento por comunicação *online*.

Uma ferramenta essencial para o desenvolvimento organizado deste projeto foi o *GitHub*. Permitiu o carregamento dos ficheiros do projeto, guardando e assinalando alterações, de modo a que todos os membros ficassem informados.

Link para o repositório do *GitHub*:

https://github.com/BrunoFCM/ProjetoAeda