Programação Orientada aos Objetos (2º ano de LCC)
GeocachingPOO

Joao Ferreira a50193 | José Simões a68692 | Luis Azevedo a66704

6 de Junho de 2015

Figura 1: Grupo 8

1 Introdução

No âmbito da cadeira de programação orientada objetos do 2º ano semestre do segundo ano da licenciatura ciências da computação foi nos proposto desenvolver uma aplicação de geocaching, em que temos de gerir a aplicação , sendo os requisitos dados no enunciado do projeto.
Neste relatório vamos descrever todas as variáveis que utilizamos em cada classe bem como os métodos que implementamos.

Estrutura do Relatório

Cache:

Esta classe é abstrata e contém seguintes variáveis:

private String nome: Esta variável guarda o nome da cache.
private String dono: Esta variável guarda o nome do.
utilizador que criou a cache.
private String local: Esta variável guarda a localização da cache.
private GPS coordenadas: Esta variável guarda as coordenadas da cache.
private String descricao: Esta variável guarda uma breve descrição da cache.
private String terreno: Esta variável guarda o tipo de terreno da cache.
private String acessibilidade: Esta variável guarda a acessibilidade da cache.
private String perigo: Esta variável guarda os perigos da cache.
private String aptidoes: Esta variável guarda as aptidões necessárias para chegar à cache.
private String pistas: Esta variável guarda pistas para ajudar a encontrar a cache.
private String aptidoes: Esta variável guarda as aptidões necessárias para chegar à cache.
private GregorianCalendar dataCriacao: Esta variável guarda a data em que a cache foi criada.
private int tempoMedio: Esta variável guarda o tempo médio para chegar à cache.

Micro_Cache, Multi_Cache, Cache_Misterio, Cache_Evento, Cache_Virtual: São subclasses da classe Cache que representam os seus vários tipos. Todas estas classes têm em comum a variável:
private double pont: Esta variável contém a pontuação obtida.

Depois variam:
Micro_Cache:
private double tamanho: Esta variável contém o tamanho da cache.

Multi_Cache: private int checkPoints: Esta variável contém a quantidade

de pontos intermédios.

Cache_Misterio:
private int misterio: Esta variável contém a quantidade de mistérios.

Cache_Evento:
private int hora: Esta variável contém a hora de inicio da cache.
private int min: Esta variável contém o minuto de inicio da cache.

Cache_Virtual:
private String prova: Esta variável contém a prova que os utilizadores necessitam ter para provar que encontraram a cache.

Nesta classe usamos também os construtores usuais, temos os métodos get e set e também a implementação dos métodos toString(), clone() e equals(). Temos também as funções que calculam a dificuldade e a pontuação. Respetivamente calculaDificuldade e calculaPontuacao.

GPS:

Representa um ponto GPS e contém as variáveis:

private double lat: Esta variável contém a latitude.
private double longi: Esta variável contém a longitude.

Nesta classe usamos também os construtores usuais, temos os métodos get e set e também a implementação dos métodos toString(), clone() e equals().

Clima:

Representa os climas que poderemos ter no nosso programa , contém as seguintes variáveis:
private final ArrayList<String> escolheClima : É um array onde permite escolher um clima consoante a estação do ano.
private String clima : guarda o clima .
private Int Temperatura: guarda a temperatura.
private String Estação: guarda a estação.

Nesta classe usamos também os construtores usuais, temos método de calcular a estação do ano, que a partir da estação calculamos um clima e uma temperatura.

CacheAtiva:

Representa uma cache que o utilizador ativou quando chegou às coordenadas de uma cache, com esta classe temos como objetivo guardar o tempo que ele demora a procurar uma cache , já que quando ao criar uma cache, o seu criador dá um tempo médio para a encontrar.
De notar que fizemos isto para facilitar depois no calculo de simular um evento já que assim temos um bom parâmetro para classificar os participante . Contém as seguintes variáveis:
private GPS coordenadas : guarda as coordenadas de uma cache que são únicas.
private GregorianCalendar dataInicio : guarda a hora e dia em que iniciou a procura da cache.
private Clima clima : guarda o clima do dia em que a cache foi ativada.

Nesta classe como temos vindo a fazer temos os construtores usuais, get e set e também a implementação dos métodos toString(), clone() e equals().

CacheEncontrada :

Representa a cache encontrada pelo utilizador.Contém as seguintes variáveis:
private GPS coordenadas : guarda as coordenadas de uma cache que são únicas.
private double tempoDemorado : guarda o tempo demorado desde que a cache foi ativa até ser encontrada.
private Clima clima : guarda o clima que ele encontrou a cache.
private double pontos : É onde se guardo pontos que utilizador ganhou.

Nesta classe como temos vindo a fazer temos os construtores usuais,get e set e também a implementação dos métodos toString(), clone() e equals(), acrescentando o método calculaPontos que a partir do temperatura, tempo médio ,clima do dia e da pontuação base da cache calcula os seus pontos.

Utilizador :

Representa um geocacher. Contem as seguintes variáveis:

private String nome : guarda o nome do utilizador.
private String userName : guarda o userName que é unico.
private String password : guarda a passoword do utilizador.
private GregorianCalendar dataNasc : guarda a data de nascimento.
private int idade : guarda a idade que é calculada a partir da dataNasc.
private boolean parabens : guarda um boolean a dizer se o utilizador esta de parabéns ou não.
private ArrayList<String> amigos : guarda os Username dos amigos do utilizador.
private ArrayList<String> amigosIn : guarda os Username dos utilizadores que convidaram este utilizador.
private ArrayList<String> amigosOut : guarda os Username dos utilizadores que este utilizador convidou.
private ArrayList<GPS> cachesCriadas : guarda as coordenadas das caches que o utilizador criou.
private ArrayList<CacheAtiva> cachesAtivas : guarda as caches ativas pelo utilizador.
private ArrayList<CacheEncontrada> cachesEncontradas : guarda as caches que este utilizador encontrou.

private GPS coordenadasUtilizador : guarda as coordenadas do utilizador para facilitar a procura de cache perto dele.
private double pontosAno : guarda os pontos que o utilizador ganhou num ano.
private double pontosMes : guarda os pontos que o utilizador ganhou num mês.
private double pontosSempre : guarda os pontos que o utilizador ganhou desde que é um geocacher.
private TreeMap<GregorianCalendar, RegistoAtividade> regAtividades : guarda as atividades do utilizador.
private GregorianCalendar ultimoLogUtilizador : guarda a ultima vez em que o utilizador foi aplicação.

Nesta classe como temos vindo a fazer temos os construtores usuais,get e set e também a implementação dos métodos toString(), clone() e equals(),bem como o metodo de calcular idade , os métodos para dar limpar os pontos do mês e do ano e metodos de ordenação .

Administrador :

Representa um administrador da rede social. Contém as variáveis:

private String nome : guarda o nome do administrador.
private String userName : guarda o userName que é único.
private String password : guarda a password do administrador.
private boolean master : guarda se o administrador é ou não master. Se for master terá permissões para adicionar administradores e para aceder à lista de administradores.

Nesta classe como temos vindo a fazer temos os construtores usuais, get e set e também a implementação dos métodos toString(), clone() e equals().

Geocaching :

Representa a nossa base de dados .

private HashMap<String, Utilizador> utilizadores : Contém a lista de utilizadores. A key do HashMap é o userName de utilizador.

private HashMap<String, Administrador> administradores : Contém a lista de administradores. A key do HashMap é o userName de administrador.
private TreeMap<GPS, Cache> caches : Contém a lista de Caches.
private Clima clima : Contém o clima.
private GregorianCalendar ultimoLog : Contém a data da última vez que foi feito um login pelo utilizador.
private TreeMap<GregorianCalendar, String> reportAbuse : Lista com reports que tenham sido feitos tanto sobre caches como sobre utilizadores.

Nesta classe como temos vindo a fazer temos os construtores usuais, get e set e também a implementação dos métodos toString(), clone() e equals(). Temos também vários outros métodos para adicionar e remover Utilizadores, Administradores, Caches e Registo de Atividade, para calcular o clima, métodos de ordenação dos vários HashMaps e TreeMaps e métodos para gravar e ler as várias informações para ficheiro.

Registo de atividades :

Esta classe é abstrata e contém seguintes variáveis:
private Utilizador u : guarda um utilizador.

Nas classe registoamizade recebe outro utilizador, na classe registoencontrouc recebe uma cacheEncontrada e na classe registoProcuraC recebe uma cacheAtiva , sendo tudo isto registo da atividade de um utilizador.

Ler :

Na classe Ler temos algumas exceptions para controlo de erros que possam surgir ao inserir informações.

Menu :

Na classe Main, temos o menu e os métodos necessários para que o utilizador possa manipular as várias funcionalidades do programa.

Conclusão

Neste momento o nosso projeto tem implementado toda a parte basica que o enunciado pedia , faltando a gestao e simulação de eventos .
Com realização deste projecto facilitou nos a aprendizagem do paradigma orientado objetos.
Para trabalho futuro a implementação da gestão e simulação de eventos e melhoramento do menu para conter mais exception .