Introdução à Programação Orientada por Objetos

Projeto – 2ª Fase

SimFire

Sistema de Simulação de Combate a Incêndios

Ano Letivo: 2017/2018

Época Normal e de Recurso

Índice

Introdução

O objetivo deste projeto é desenvolver uma aplicação, utilizando a linguagem Java e a Programação Orientada por Objetos (POO), com o intuito de simular a atividade de gestão de meios no combate a incêndios florestais.

No problema a modelar, destacam-se as seguintes entidades:

- Os bombeiros que combatem o fogo,
- Os veículos de combate a incêndios que transportam uma quantidade limitada de água e que possuem uma equipa de bombeiros para os operar,
- O quartel que tem vários bombeiros e veículos de combate a incêndios. Os quartéis estão localizados em vários pontos do país,
- Os incêndios que se caraterizam pelo local de ocorrência (coordenadas), a área que está a arder e a hora de início e fim (quando aplicável).
- As condições atmosféricas que influenciam o incêndio, nomeadamente, intensidade do vento, temperatura e humidade.

Além das entidades da simulação será ainda necessário ter um simulador que irá integrar um mecanismo que permita instanciar as entidades descritas. Com isto será possível criar e simular um "Teatro de Operações".

Esta segunda fase dá continuidade à anterior, pretendendo-se a reestruturação de algumas partes da implementação, contemplando novos mecanismos/técnicas/padrões entretanto aprendidos na disciplina, bem como a criação de algumas funcionalidades adicionais. Estes serão descritos no ponto seguinte.

Detalhes de Implementação

Código e Estrutura de Classes

Nesta segunda fase é levantada a restrição imposta anteriormente de uso exclusivo de *arrays* para representação de coleções. Assim, **onde se justificar**, os *arrays* devem ser alterados para uma representação com recurso a uma das coleções do Java que foram aprendidas. É importante que o tipo de coleções escolhido seja adequado à coleção que representa. Na escolha dos tipos de coleção tenha em consideração as novas funcionalidades que são pedidas.

Não existem restrições à modificação de código entregue na primeira fase. Neste caso, pode aproveitar para melhorar a qualidade da estrutura de classes e do código de acordo com as boas práticas de escrita, nomeadamente: um acoplamento fraco, uma boa coesão e um desenho orientado por responsabilidades.

Documentação e testes

Deverá ser criada uma bateria de testes adequada às entidades criadas, recorrendo à *framework* **JUnit**, conforme abordado na disciplina. Deste modo, os testes deverão ser suficientemente extensivos para que toda a lógica fundamental do projeto seja validada.

Pretende-se igualmente a criação de documentação adequada às diversas entidades implementadas, neste caso com recurso a **JavaDoc**.

Novos Requisitos

Adicionalmente aos requisitos expostos na primeira fase, o sistema deverá ter em conta as seguintes necessidades:

Preparação de estatísticas

No final da simulação será necessário mostrar ao utilizador vários parâmetros estatísticos. Alguns recolhidos ao longo do processamento, enquanto outros são calculados no final da época de incêndios. Sinta-se livre para criar o código auxiliar que achar necessário para guardar os dados pretendidos. Sempre que possível, utilize o processamento funcional de coleções na obtenção dos dados estatísticos.

- Número total de focos de incêndio
- Número total de veículos envolvidos
- Número total de bombeiros que participaram
- Área ardida (ha)
- Tempo médio de combate dos fogos (dias/horas)
- Número médio de operacionais por foco de incêndio
- Total de água gasta na simulação
- Média de água gasta por incêndio
- Número de quilómetros percorridos em média pelos veículos
- Número médio de fogos em que cada bombeiro participou

Regras de Desenvolvimento e Entrega do Projeto

Entrega

A data de entrega do projeto é **31 de janeiro de 2018**.

Implementação e codificação

O programa deve ser desenvolvido utilizando a linguagem Java, colocando em prática os conceitos fundamentais do paradigma de Programação Orientada por Objetos.

Em relação às regras de codificação, siga as convenções adotadas normalmente para a linguagem Java:

- A notação *camelCase* para os nomes de métodos e dos seus parâmetros, de variáveis locais e de atributos;
- A notação *PascalCase* para os nomes das classes;
- Não utilize o símbolo '_' nos identificadores, nem abreviaturas.

É necessário que o projeto cumpra o que é pedido no seu enunciado, sendo deixado ao critério do programador qualquer pormenor de implementação que não seja referido, o qual deverá ser devidamente documentado.

Nas funcionalidades desenvolvidas, deverão ser incluídas todas as validações necessárias para impedir um comportamento incorreto do sistema.

Sempre que fizer sentido, os métodos deverão ser responsáveis por apresentar no ecrã mensagens de informação e/ou erro, indicando o processamento que foi feito ao objeto.

Nas situações em que for adequado, devem ser facultadas diferentes assinaturas para os métodos implementados.

Os nomes das classes, atributos e métodos deverão ser definidos em língua inglesa. As mensagens apresentadas pela aplicação podem ser apresentadas em português e/ou inglês.

Constituição de grupos

Cada projeto deverá ser elaborado em grupos de dois alunos, podendo ser desenvolvido individualmente em casos pontuais devidamente justificados. Não serão permitidos, **em nenhum caso**, grupos com mais do que dois alunos.

Os grupos já se encontram determinados através da metodologia de *pair programming* que está a ser utilizada nos laboratórios. Caso existam alunos que não têm o grupo escolhido, deverão contactar o respetivo docente de laboratório para regularizar a situação.

Entrega do projeto

- O projeto deverá ser entregue até à data limite especificada por **via exclusivamente eletrónica, utilizando a área dos trabalhos na plataforma Moodle**. Todos os ficheiros que compõem o projeto

deverão estar guardados num único ficheiro compactado em **formato ZIP**. Em caso de dificuldades no acesso à plataforma Moodle, o envio dos ficheiros poderá ser feito por correio eletrónico para o respetivo docente de laboratório, dentro do prazo acima indicado.

- **Não serão aceites quaisquer projetos entregues fora do prazo!**
- Todos os materiais do projeto devem ser devidamente identificados com nome, número e endereço de correio eletrónico dos alunos.

Os materiais do projeto deverão incluir:

- A documentação do programa, explicando de forma simples as classes criadas, juntamente com os seus atributos e métodos, bem como qualquer detalhe de implementação que necessite de explicações adicionais.
- O código fonte do programa na forma de projeto em BlueJ ou Netbeans, com todas as funcionalidades implementadas. **Não existe necessidade de implementação de uma interface com o utilizador**.
- Todos os ficheiros que compõem o projeto deverão estar guardados num único ficheiro compactado em formato ZIP cujo nome deverá ter a seguinte nomenclatura: ***<curso>_<numAluno1>_<numAluno2>*.zip (exemplo: EI_12345678_87654321)**. **O incumprimento das normas de entrega está sujeito a uma penalização de até 1 valor.**

Regras e Critérios de Avaliação do Projeto

Regras de Avaliação

A avaliação do projeto está sujeita às seguintes regras:

- Não serão aceites quaisquer projetos entregues fora do prazo.
- A classificação do programa terá em conta a qualidade da programação e a estrutura do código criado segundo os princípios da Programação Orientada por Objetos.
- Serão premiadas a imaginação e a criatividade.
- O projeto terá uma componente de **avaliação oral obrigatória** com classificação individual dos elementos do grupo.
- Os alunos que não comparecerem à discussão serão classificados com zero na fase respetiva. Nesta discussão será apurada a capacidade do aluno de produzir o código apresentado. Nos casos em que essa capacidade não for demonstrada, a nota atribuída será zero.
- A avaliação oral é realizada pelo respetivo professor de laboratório e irá ser feita uma marcação prévia para cada grupo de trabalho.
- Todos os projetos serão submetidos a um sistema automático de deteção de cópias. Os projetos que forem identificados como possíveis cópias, e verificando-se serem efetivamente cópias, serão anulados.

Critérios de Avaliação

Esta segunda fase será avaliada segundo os seguintes critérios:

Funcionalidades	**30%**
Estatísticas	15%
Qualidade e detalhe da estratégia de simulação	15%

Implementação	**60%**
Restruturação e estrutura do projeto (não inclui alterações das coleções usadas)	10%
Estrutura e funcionamento interno das classes	10%
Utilização de coleções (`ArrayList`, `HashSet` e `HashMap`)	10%
Bom estilo (identificadores, comentários, indentação)	10%
Testes	10%
Documentação (JavaDoc e relatório)	10%

Avaliação qualitativa	**10%**
Qualidade geral, detalhes de implementação e funcionalidades adicionais	10%

Bom trabalho!