Relatório Meta 2

Trabalho Prático

Licenciatura em Engenharia Informática

POO

2019/2020

Trabalho realizado por:

Filipe Silva 2016020567

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

    a. **Consola** – Obtido do professor
        i. Criar configurações de consola
    b. Autodromo
        i. Cordena o autodromo
    c. **Campeonato**
        i. Cordena o campeonato dos carros
    d. TextUI
        i. Obter comandos do utilizador
        ii. Configurar campeonato
        iii. Configurar carros
        iv. Configurar pilotos
    e. Campeonato
        i. Gere o campeonato e suas componentes
    f. carro
        i. Todos os dados que englobam o carro
    g. piloto
        i. Todos os dados que englobam o piloto

2. **Quais os conceitos/classe que identificou ao ler o enunciado?**
    a. Autodromo
    b. Carro
    c. Piloto

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.**

    DGV: Geria a parte logica, criação de objetos etc.

    TextUI: Interacao com o utilizador.

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

    a. A classe Piloto(base) e as suas variantes (crazt,rapido,surpresaa) possuem dados que são responsáveis pela execução e representação das ações dos pilotos, sendo que estas são as suas únicas responsabilidades.

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

    a. A classe Piloto(base) e as suas variantes (crazt,rapido,surpresaa) possuem dados que são responsáveis pela execução e representação das ações dos pilotos, sendo que estas são as suas únicas responsabilidades.

b. A classe Campeonato (Base) e engloba todas as classes usadas na logica da interação com o utilizador.

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:

   a. A **classe simulacao2** é responsável pela interpretação e a "ponte" entre o utilizador e a lógica do programa. Por exemplo, a interpretação de comandos e a representação gráfica das classes.
   b. A **classe consola** é outra classe responsável pela apresentação dos dados, mas maioritariamente para controlo e acesso ao "ecrã/consola".

   Responsabilidade da lógica da aplicação:

   a. A **classe Mundo** é o "contentor" dos restantes elementos do mundo, onde todos os elementos dependem do mundo para serem criados.
   b. A **classe Ninho** é o "contentor" das formigas, pois não haveram formigas de dada comunidade se não houver ninho.
   c. A **classe Formiga** é a ultima classe da hierarquia de contentores, sendo que este objeto não irá conter mais nada para além das suas regras e informações.
   d. A **classe Migalhas** serve somente ter as suas informações, sendo que as formigas e o mundo irão interagir com este elemento.
   e. A **classe Coordenadas** é responsável por armazenar as coordenadas cartesianas de dado objeto, a sua principal função será a comparação entre coordenadas com o operador ==.
   f. A **classe Regra e suas derivadas** são responsáveis pelas ações das formigas.
   g. A **classe Random** é responsável por gerar coordenadas aleatórias e números aleatórios para auxílio em algumas partes do programa.

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

   a. As ordens vindas da camada de interação com o utilizador são recebidas pela classe mundo, que irá por si interagir com as restantes classes do programa.

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. O mundo representa a envolvente de toda a lógica. Este utiliza a classe Ninho para criar e gerir as formigas, utiliza os vetores de ninhos e de migalhas para os manusear.

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.**

Existem várias classes de formigas que se distinguem em, pelo menos, no conjunto de regras que as integram. Desta forma, a classe formiga (a partir do método acao()) chama, por vez a execução de cada um das suas regras, a partir do mesmo método: executa().

```
void Formiga::acao()
{
    for(const auto & x : regras)
        if(x->executa_regra(this,mundo) == true)
            return;
}
class Regra {
    const string nome;

public:
    Regra(string s):nome(s){};

    string getNome() const {return nome;};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const {};
    virtual ~Regra();
};

class Reg_passeia : public Regra {

public:
    Reg_passeia():Regra("PASSEIA"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};

class Reg_come_migalha : public Regra {

public:
    Reg_come_migalha():Regra("COME"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};

class Reg_foge : public Regra {

public:
    Reg_foge():Regra("FOGE"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};

class Reg_persegue : public Regra {

public:
    Reg_persegue():Regra("PERSEGUE"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const {return false;}; //TODO: persegue;
};

class Reg_assalta : public Regra {

public:
```

```
    Reg_assalta():Regra("ASSALTA"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};

class Reg_protege : public Regra {

public:
    Reg_protege():Regra("PROTEGE"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};

class Reg_procura : public Regra {

public:
    Reg_procura():Regra("PROCURA"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const {return false;}; //TODO: isto;
};

class Reg_vai_p_ninho : public Regra {

public:
    Reg_vai_p_ninho():Regra("VAI_P_NINHO"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};

class Reg_invasao : public Regra{

public:
    Reg_invasao():Regra("INVASAO_NINHO"){};
    virtual bool executa_regra(Formiga * formiga, const Mundo *
mundo) const;
};
```

10. Apresente as principais classes da aplicação através da seguinte informação:

Classe: Mundo

Responsabilidades:

- Criar Ninhos
- Criar Migalhas
- Listar informações do mundo
- Verificar existência de uma comunidade
- Validar posições do mundo
- Iterar mundo
- Limpar ninhos, sem energia
- Destruir regras

Colaboração:

Coordenadas, Formiga, Migalha, Ninho

Classe: Formiga

Responsabilidades:

Servir de base para outras formigas (polimorfismo)
Devolve informação relativa a formiga (posição, energia, etc…)

Colaboração:

Coordenadas, Regra

Classe: Formiga Exploradora, Cuidadora, Vigilante, Surpresa e Assaltante

Responsabilidades:

Respeitar as suas regras e interagir com o mundo conforme.
Devolver informação textual relativa ao seu tipo.

Colaboração:

Coordenadas, Regra e suas classes derivadas, Mundo, Ninho

Classe: Ninho

Responsabilidades:

Conter vetor de formigas pertencentes à comunidade
Dar energia às formigas que se encontram no ninho
Cria formigas e atribui-lhes as suas regras
Limpa as formigas mortas do seu ninho

Colaboração:

Mundo, Coordenadas, Formiga e suas derivadas, Regras e suas derivadas

Classe: Migalha

Responsabilidades:

Dar energia às formigas na sua vizinhança, ou a alguma sobreposta.

Avisar mundo quando ficar sem energia.

Colaboração:

Formiga, Coordenadas

12. Funcionalidades implementadas

Componente do trabalho	Realizado	Realizado parcialmente	Não realizado
Leitura e validação de comandos	X		
Leitura de comandos em ficheiro	X		
Criação de mundo, ninho, formigas e migalhas	X		
Visualização do mundo e conteúdo	X		
Formigas interagem com o mundo	X		
Avanço de iteração	X		
Listagens (listamundo, listaninho, listaposicao)	X		
Comandos de configuração	X		
Comandos de simulação	X		
Regras das formigas	X		