PROVA DE EXAME
de
Programação Orientada por Objetos

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época de Recurso – 13 de fevereiro de 2019

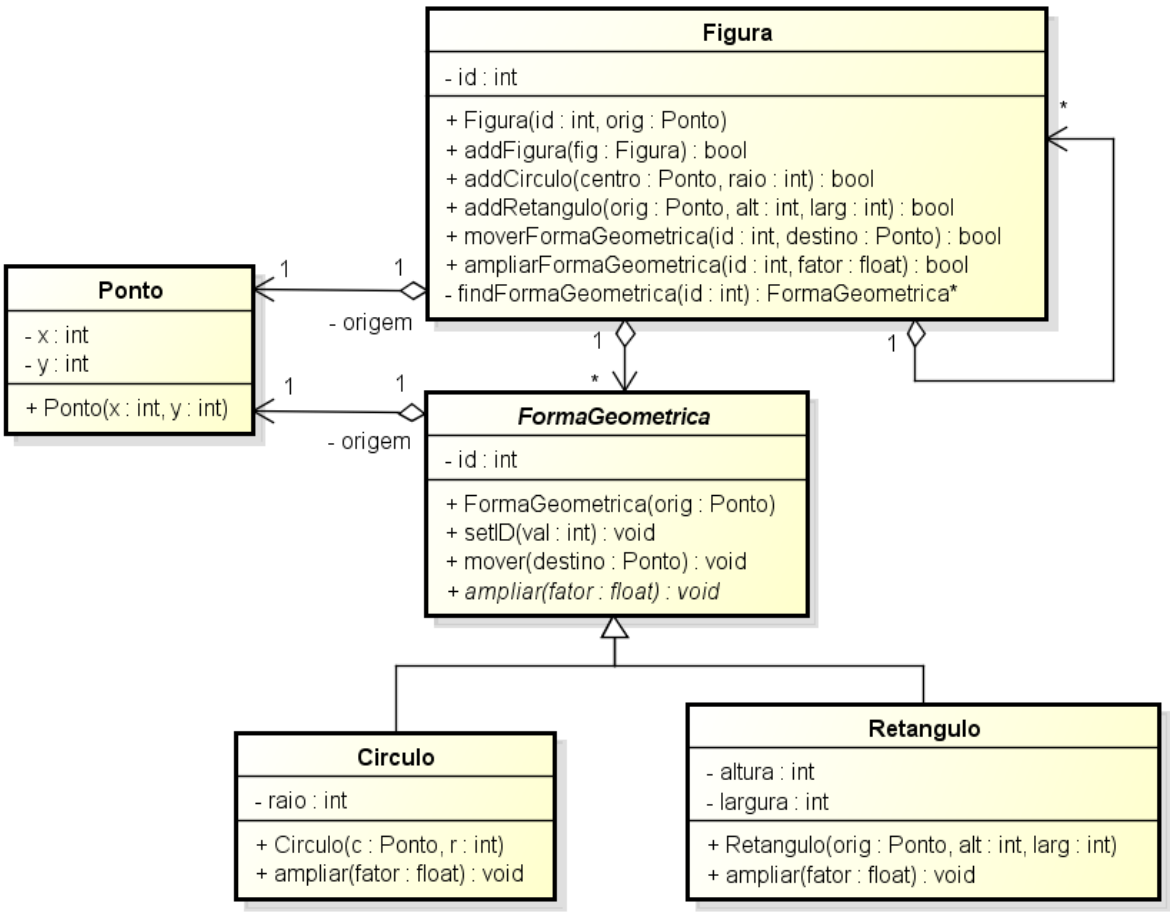
Notas importantes:

1. *A duração da prova é de 2 horas.*
2. *Comece por preencher a zona reservada à sua identificação na folha de exame.*

Grupo I

[16 valores]

Uma figura pode ser formada por formas geométricas simples ou, até mesmo, por outras figuras. Cada figura é também caraterizada, tal como as formas geométricas simples, pela posição que ocupa no plano (ponto origem) e por um código inteiro que a identifica. Para além das operações responsáveis pela criação dos elementos gráficos, deverá ser possível mover e ampliar as formas geométricas que integram uma qualquer figura. Segue-se o diagrama de classes UML que descreve de forma precisa a solução que se pretende para o problema descrito.



a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama, e tendo ainda em conta todas as seguintes considerações: [12 val.]

- *Não defina quaisquer outros métodos ou atributos, a não ser, nas classes em que se justifique, o operador que permita que os objetos sejam colecionáveis;*
- *Os ids das Formas Geométricas devem ser atribuídos de forma automática, garantindo que ao 1º elemento é atribuído o id 1, ao 2º o id 2, e assim sucessivamente;*
- *Tenha em atenção que algumas das entidades poderão ser abstratas (representadas a itálico no diagrama UML);*
- *As coleções deverão ser implementadas com base no template de classes Colecao ou ColecaoHibrida que utilizou nas aulas de POO. Para efeitos de consulta, disponibilizam-se os protótipos dos seus principais métodos no Anexo B deste enunciado;*
- *Defina os métodos no momento em que está a declarar as classes (não separe a declaração da implementação) e considere que todo o código reside num único ficheiro;*
- *Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.*

b) Acrescente ao problema os métodos que permitam apresentar na saída standard todos os elementos contidos numa figura, com indicação de que tipo de elemento se trata (Figura, Circulo, …) e de todos os atributos que o caraterizam. [3 val.]

c) Implemente um pequeno *main* que faça uso de todas as funcionalidades da classe Figura. [1 val.]

Grupo II

[4 valores]

No Anexo A do presente enunciado encontra-se o código de um pequeno programa em C++. Depois de o analisar com a devida atenção, responda às seguintes questões.

a) Diga quais são os métodos construtores (definidos explicitamente ou não) que estão presentes em cada uma das classes do problema. [0.8 val.]

b) Apresente o resultado que será visualizado na saída standard após a execução do programa. [1.8 val.]

c) Diga que consequências é que teria, naquilo que é visualizado, se os destrutores de ambas as classes fossem definidos como virtuais. [0.4 val.]

d) Faça as alterações que achar necessárias nas classes do problema para que a linha de código que se segue possa surgir na função *main*. [0.5 val.]

```
if (eq1 == eq2) cout << "Equipas em igualdade pontual" << endl;
```

e) Por fim, diga o que entende por *classe abstrata*, e se alguma das classes do problema pode ser classificada com tal. [0.5 val.]

ANEXO A

```
#include<iostream>
#include<string>
using namespace std;

class Equipa{
	string nome;
	int pontos;
public:
	Equipa(const string &n) : nome(n) {
		pontos = 0;
		cout << "Equipa: ";
		print();
	}
	string getNome() { return nome; }
	void operator+=(int val) { if (val>=1 && val<=3) pontos += val;}
	void print() { cout << nome << " (" << pontos << ")" <<endl; }
	~Equipa() {
		cout << "~Equipa: ";
		print();
	}
};

class Jogo{
	Equipa *anfitria;
	Equipa *visitante;
	int golos[2];
public:
	Jogo(Equipa *eq1, Equipa *eq2) {
		anfitria = eq1; visitante = eq2;
		golos[0] = golos[1] = 0;
		cout << "Inicio do jogo: ";
		print();
	}
	void setResultado(int ga, int gv) { golos[0] = ga; golos[1] = gv; }
	void print() {
		cout <<anfitria->getNome()<<" "<< golos[0]<<" - "<<golos[1] <<" "
		     <<visitante->getNome()<< endl;
	}
	~Jogo() {
		cout << "Fim do jogo: "; print();
		if (golos[0] > golos[1]) (*anfitria) += 3;
		else if (golos[0] < golos[1]) (*visitante) += 3;
		     else { (*anfitria) += 1; (*visitante) += 1; }
	}
};

void main() {
	cout << "-0-" << endl;
	Equipa eq1("City"), eq2("Chelsea");
	cout << "-1-" << endl;
	Jogo jogo1(&eq1, &eq2);
	cout << "-2-" << endl;
	Jogo jogo2(jogo1);
	cout << "-3-" << endl;
	jogo2.setResultado(6, 0);
}
```

ANEXO B

Template Colecao

```
#include<set>
using namespace std;

template<class K>
class Colecao: public set<K>{
public:
	bool insert(const K &c);
	K *find(const K &c);
	int size() const;
	void erase(const K &);
	//void clear();
	//bool empty() const;
	//iterator begin();
	//iterator end();
};
```

Template ColecaoHibrida

```
#include<set>
using namespace std;

template<class T>
class less_pointers{
public:
	bool operator()(const T &left, const T &right) const {
		return (*left < *right);
	}
};

template<class K>
class ColecaoHibrida: public set<K, less_pointers<K>>{
public:
	bool insert(const K &c);
	K find(const K &c);
	int size() const;
	void erase(const K &);
	//void clear();
	//bool empty() const;
	//iterator begin();
	//iterator end();
};
```