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 – 8 de fevereiro de 2014

Notas importantes:

1. *O Grupo III é de resolução facultativa e destina-se a substituir as notas do trabalho prático e miniteste*
2. *Duração da prova: 1h30 (Grupos I e II) + 1h (Grupo III)*
3. *Comece por preencher a zona reservada à sua identificação na folha de exame;*
4. *Resolva os três grupos em folhas de exame separadas.*

Grupo I

(8 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) Identifique o tipo de relação existente entre *Instituto* e *Escola* e entre *Escola* e *ESTiG*. [0.6 val.]

b) Diga quais são os métodos construtores que estarão presentes em cada uma das classes do problema. [1.0 val.]

c) Nas classes do problema existem métodos que deveriam ser definidos como constantes? Se sim, diga quais. [0.6 val.]

d) Apresente o resultado que será visualizado na saída standard com a execução do programa. [3.6 val.]

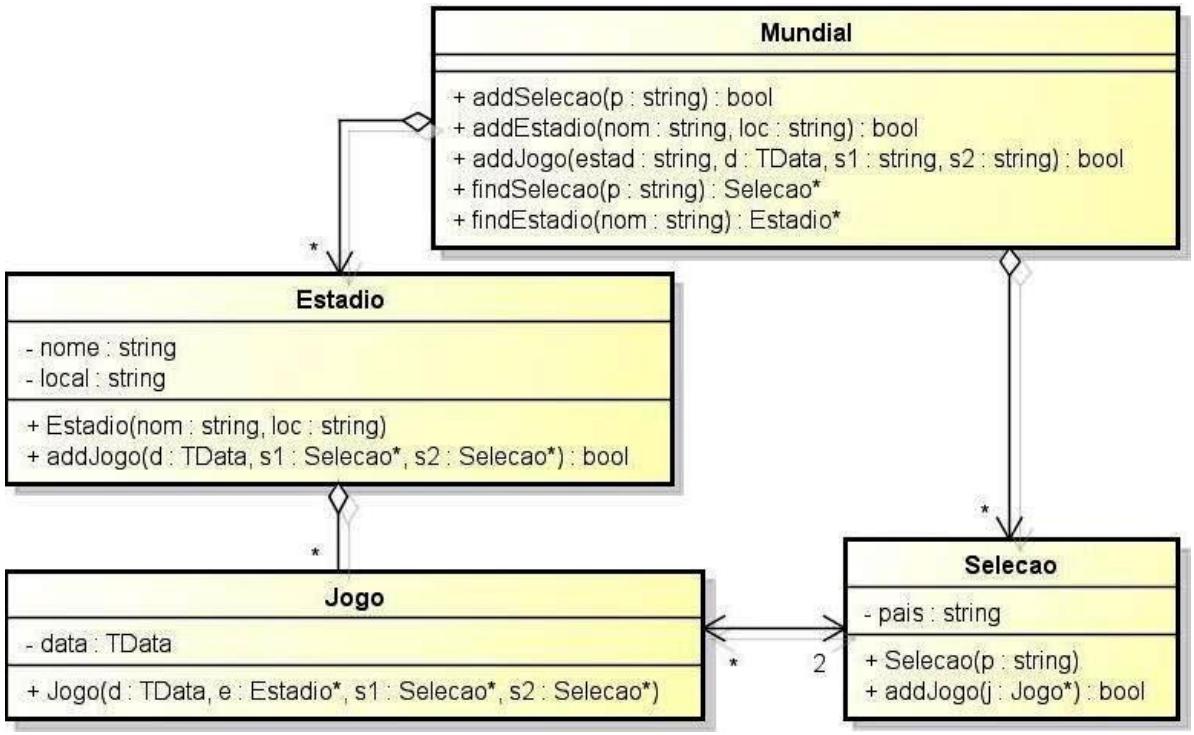
e) O que seria visualizado se o método `print` da classe `Escola` não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [1.0 val.]

f) E se fosse o método destrutor da classe `Escola` a não ser virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea d). [0.6 val.]

g) Diga se as declarações `ESTiG V1[5]; ESA *V2[5];` são ou não válidas no problema considerado. Caso não sejam, introduza as alterações necessárias nas classes do problema para que as declarações passem a ser válidas. Diga, por fim, quantos construtores serão invocados em cada uma das declarações. [0.6 val.]

Grupo II
(12 valores)

Estamos já a escassos meses do Mundial 2014, Campeonato do Mundo de futebol que mais uma vez contará com a presença de Portugal e que este ano terá o Brasil como país anfitrião. Suponha que o incumbiram a si de desenvolver uma pequena aplicação que permita gerir informaticamente a distribuição dos jogos da competição pelos diferentes estádios disponibilizados para o evento. Segue-se o diagrama de classes UML que descreve de forma precisa a solução que se pretende para a aplicação descrita.



a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos; apenas acrescente, nas classes em que se justifique, o operador que permita que os objetos sejam colecionáveis. [8.5 val]

NOTA 1: Admita que num mesmo estádio não podem ocorrer dois jogos no mesmo dia;

NOTA 2: Para o tipo data é usada a classe TData, estudada nas aulas da UC, e da qual se apresentam, no Anexo B deste enunciado, o protótipo de alguns métodos e operadores;

NOTA 3: 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;

NOTA 4: 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;

NOTA 5: Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.

b) Acrescente ao problema os métodos que permitam mostrar todas os jogos realizados num dado estádio, com a indicação da data e das duas seleções opositoras. [2.5 val.]

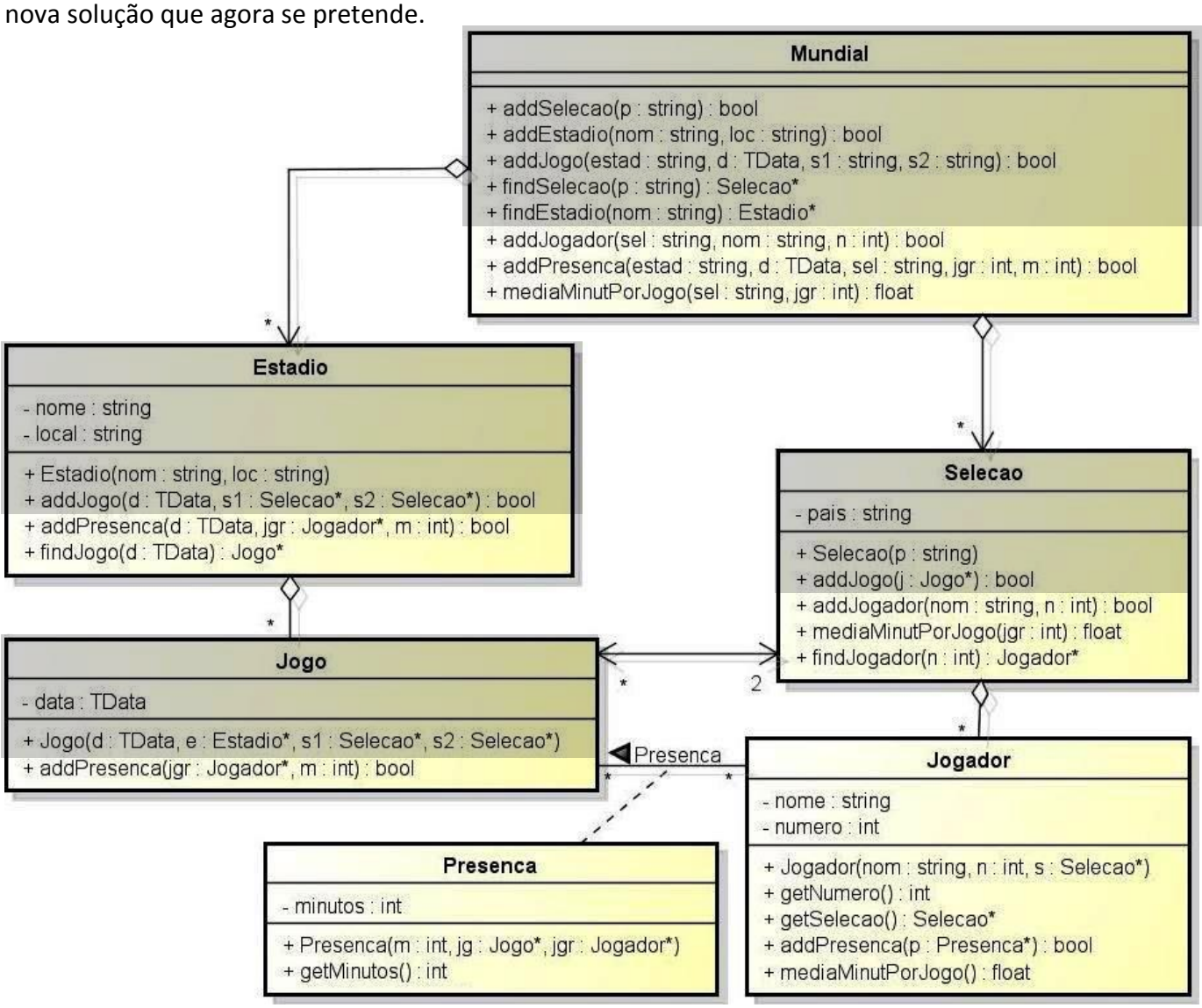
c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.0 val.]

Grupo III

(10 valores)

*NOTA: O problema que se segue é um exercício suplementar e facultativo que se destina a substituir a classificação obtida durante o período letivo (miniteste+trabalho). Com a sua resolução, ou tentativa de resolução, essa componente de avaliação deixará de contar para a época de recurso, e a sua nota final será: (GI+GII+GIII)*2/3.*

A ideia agora é acrescentar à aplicação a capacidade de registar, para cada jogador, o número de minutos que esteve em campo em cada um dos jogos em que participou. Como nova funcionalidade, a aplicação deve passar a permitir calcular, para um dado jogador, o valor médio do número de minutos que esteve em campo por jogo. Segue-se o diagrama de classes UML que descreve de forma precisa a nova solução que agora se pretende.



Este exercício destina-se a acrescentar novas funcionalidades ao problema que começou a implementar no grupo de questões anterior. Por isso, nas suas repostas deve ter sempre em conta o código que já implementou anteriormente para o problema.

a) Defina em C++ as novas classes Jogador e Presenca, e acrescente às já existentes as novas funcionalidade do problema, respeitando integralmente os métodos e atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos; apenas acrescente, nas classes em que se justifique, o operador que permita que os objetos sejam colecionáveis. [9.0 val]

b) Implemente um pequeno main que faça uso das novas funcionalidades da aplicação. [1.0 val.]

ANEXO A

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


class Escola{
      string nome;
public:
      Escola(){cout<<"Criada escola sem nome!"<<endl;}
      Escola(const string &n):nome(n){cout<<"Criada a "<<nome<<". ";}
      virtual void print(){cout<<"A escola e' mesmo fixe!"<<endl;}
      virtual ~Escola(){cout<<nome<<" encerrada!"<<endl;}
};


class ESTiG: public Escola{
public:
      ESTiG():Escola("ESTiG"){cout<<"A melhor escola do pais!"<<endl;}
      void print(){cout<<"A ESTiG e' mesmo fixe!"<<endl;}
      ~ESTiG(){cout<<"Encerrada a melhor escola do pais!"<<endl;}
};


class ESA: public Escola{
public:
      ESA():Escola("ESA"){cout<<"A 2a melhor escola do pais!"<<endl;}
      void print(){cout<<"A ESA e' mesmo fixe!"<<endl;}
      ~ESA(){cout<<"Encerrada a 2a melhor escola do pais!"<<endl;}
};


class Instituto{
      Escola *esc;
public:
      Instituto(){cout<<"Novo instituto sem escolas"<<endl; esc=NULL;}
      Instituto(Escola *e){esc=e;}
      void print(){esc->print();}
      ~Instituto(){cout<<"Instituto encerrado"<<endl;}
};


void main(){
      ESTiG estig;
      Escola *esa=new ESA;
      Instituto i1(&estig), i2(esa);
      i1.print();
      i2.print();
      delete esa;
}
```

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();
};
```

Classe TData

```
class TData{
public:
	TData();
	TData(int d, int m, int a);// cria o objeto com a data d/m/a
	TData(const string & dat); // cria o objeto com a data expressa na string dat
				//(string com uma data no formato "x/x/xxxx" ou "x-x-xx")
	TData(const char *dat); //cria o objeto c/ a data expressa na string tipo C dat
	string data_str() const;// devolve uma string com a data no formato "x/Mes/xx"
	bool operator<(const TData &outra) const;
	static TData hoje(); //devolve data corrente
	...
};

ostream &operator<<(ostream &os, const TData &dat);
istream &operator>>(istream &is, TData &dat);
```