PROVA DE EXAME
de
Programação Orientada por Objetos

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época Normal – 3 de fevereiro de 2017

Notas importantes:

1. *A duração da prova é de 1 hora e 30 minutos.*
2. *Comece por preencher a zona reservada à sua identificação na folha de exame.*

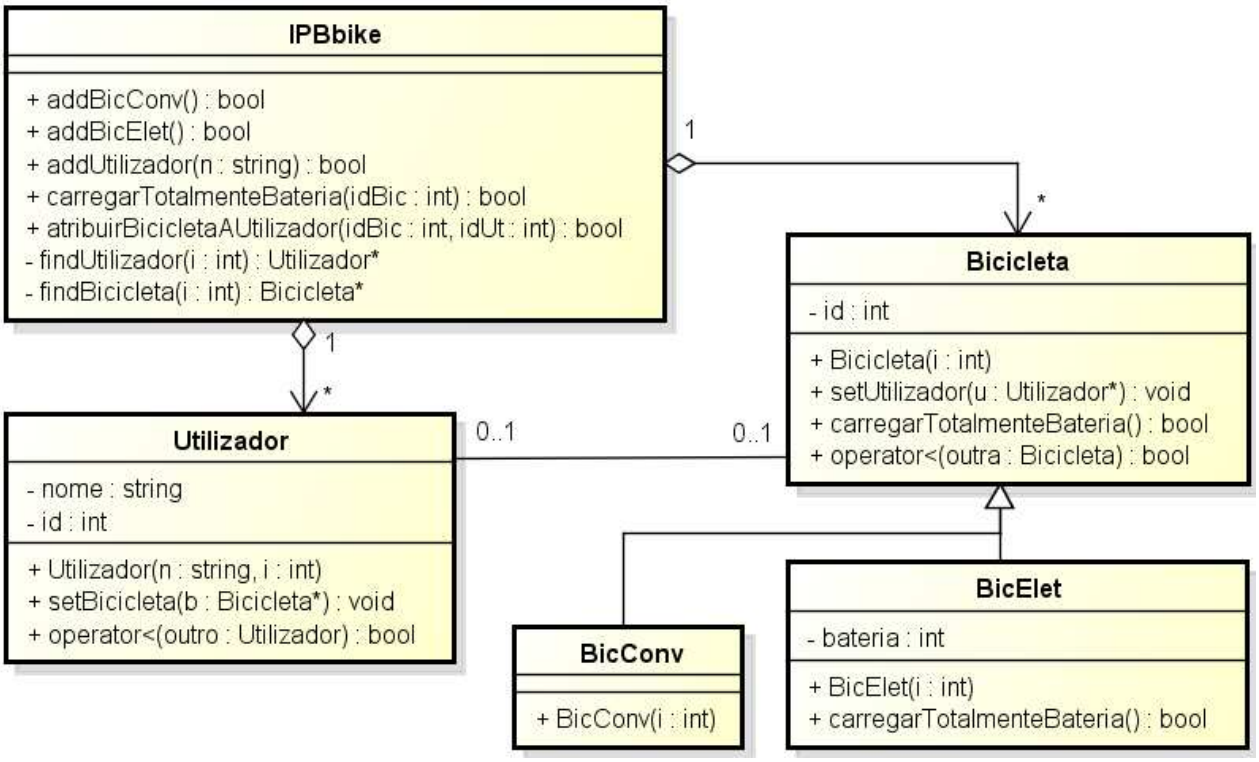
Grupo I

[15 valores]

O Projeto U-Bike Portugal é uma iniciativa de âmbito nacional (coordenada pelo IMT e enquadrando-se nos apoios do Portugal 2020), que visa incentivar a adoção de hábitos de mobilidade mais sustentáveis nas comunidades académicas das Instituições de Ensino Superior públicas, através da disponibilização de bicicletas elétricas e convencionais, estimando-se que, com essa medida, se venham a percorrer mais de 2 milhões de quilómetros a pedalar, o equivalente a mais de 60 voltas ao mundo.

O IPB viu a sua candidatura ser aprovada e, como tal, propomos-lhe que desenvolva uma pequena aplicação que venha no futuro a permitir gerir facilmente a utilização do parque de bicicletas que vierem a ser disponibilizadas ao IPB no âmbito desse projeto. Pretende-se, essencialmente, saber a cada momento que utilizador é que detém cada bicicleta que esteja a ser utilizada. O programa prevê a disponibilização de um conjunto considerável de bicicletas, quer elétricas, quer convencionais, todas elas identificadas por um código inteiro e, no caso das elétricas, também caracterizadas por possuírem um indicador que revele a todo o instante o estado da sua bateria.

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 defina quaisquer outros métodos ou atributos. [12 val.]

NOTA 1: Os ids, quer dos utilizadores, quer das bicicletas, devem ser atribuídos de forma automática pelo sistema, garantindo que ao 1º elemento é atribuído o id 1, ao 2º o id 2, e assim sucessivamente;

NOTA 2: As bicicletas elétricas são adicionadas ao sistema com a sua bateria completamente descarregada (0%);

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 na saída standard as bicicletas que estejam a ser utilizadas, mostrando para cada uma delas o respetivo id, se se trata de uma bicicleta elétrica ou convencional, e o nome de quem a está a utilizar. [3 val.]

Grupo II

[5 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) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2 val.]

b) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos da classe Base fosse virtual (incluindo o método destrutor). Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior [1.0 val.]

c) Por fim, para descomprimir, uma questão meramente teórica, que nada tem a ver com o pequeno programa tratado nesta secção.
Indique, no contexto da POO, um sinónimo para cada um dos seguintes conceitos: [2.0 val.]

(A) – Instância de uma classe;
(B) – Instância de uma template de classes;
(C) – Método virtual;
(D) – Método virtual puro;
(E) – Membro variável de uma classe;
(F) – Membro função de uma classe;
(G) – Superclasse;
(H) – Subclasse;
(I) – Derivação de classes;
(J) – Atributo estático;

ANEXO A

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

class Base{
public:
	Base() {cout << "Base()" << endl;}
	Base(const string &str) {cout << "Base("<<str<<")"<< endl; }
	virtual string toString() { return "classe generica "; }
	virtual ~Base() {cout << "~Base()" << endl; }
};

class Derivada: public Base {
public:
	Derivada(){ cout << "Derivada()" << endl; }
	Derivada(const string &str): Base(str){ cout<<"Derivada("<<str<<")"<< endl;}
	string toString() { return "classe especializada "; }
	~Derivada() { cout << "~Derivada()" << endl; }
};

void main() {
	Base *v[3];
	cout << "OUTPUT 0:" << endl;
	Base b;
	v[0] = &b;
	cout << "OUTPUT 1:" << endl;
	Derivada d("Ola");
	v[1] = &d;
	cout << "OUTPUT 2:" << endl;
	v[2] = new Derivada();
	cout << "OUTPUT 3:" << endl;
	cout << v[0]->toString() << v[1]->toString() << v[2]->toString() << endl;
	cout << "OUTPUT 4:" << endl;
	delete v[2];
	cout << "OUTPUT 5:" << endl;
}
```

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();
};
```