INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão
Época Normal – 3 de fevereiro de 2017

Grupo I (15 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [12 val.]

```
class Utilizador{                          (5)                                  1
    string nome;
    int id;
    Bicicleta *bicileta;                                                        1
public:
    Utilizador(const string &n, int i): nome(n), id(i){bicileta=NULL;}          1
    void setBicicleta(Bicicleta *b) {bicileta=b;}                               1
    bool operator<(const Utilizador &outro)const {return id<outro.id;}         1
};

class Bicicleta{                           (7)                                  1
    int id;
    Utilizador *utilizador;                                                     1
public:
    Bicicleta(int i) { id = i; utilizador = NULL; }                             1
    void setUtilizador(Utilizador *u) { utilizador=u;}                          1
    virtual bool carregarTotalmenteBateria() { return false; }                  2
    bool operator<(const Bicicleta &outra)const{return id<outra.id;}            1
};

class BicConv: public Bicicleta{                   (3)                          2
public:
    BicConv(int i): Bicicleta(i){}                                              1
};

class BicElet: public Bicicleta{                   (6)                          2
    int bateria;
public:
    BicElet(int i): Bicicleta(i){bateria=0;}                                    2
    bool carregarTotalmenteBateria() { bateria = 100; return true; }            2
};

class IPBbike{                (19)
     Coleccao<Utilizador> utilizadores;                                         2
     ColeccaoHibrida<Bicicleta*> bicicletas;                                    3
public:
     bool addBicConv(){                                                         3
        int id = bicicletas.size() + 1;
        BicConv *b = new BicConv(id);
        return bicicletas.insert(b);
     }
```

```
    bool addBicElet(){
        int id = bicicletas.size() + 1;
        BicElet *b = new BicElet(id);                                    1
        return bicicletas.insert(b);
    }

    bool addUtilizador(const string &n){                                 2
        int id = utilizadores.size() + 1;
        Utilizador u(n,id);
        return utilizadores.insert(u);
    }

    bool carregarTotalmenteBateria(int idBic){                           2
        Bicicleta *pb=findBicicleta(idBic);
        if(pb==NULL){
            cout<<"Bicicleta inexistente!";
            return false;
        }else return pb->carregarTotalmenteBateria();
    }

    bool atribuirBicicletaAUtilizador(int idBic, int idUt){              3
        Bicicleta *pb=findBicicleta(idBic);
        if(pb==NULL) {cout<<"Bicicleta inexistente!\n"; return false;}
        else{
            Utilizador *pu=findUtilizador(idUt);
            if(pu==NULL) {cout<<"Utilizador inexistente!\n"; return false;}
            else{
                pb->setUtilizador(pu);
                pu->setBicicleta(pb);
                return true;
            }
        }
    }

private:
    Utilizador *findUtilizador(int i){                                   1
        Utilizador u("",i);
        return utilizadores.find(u);
    }
    Bicicleta *findBicicleta(int i){                                     2
        Bicicleta b(i);
        return bicicletas.find(&b);
    }
};
```

b) Acrescente ao problema os métodos que permitam mostrar no écran 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.]

```
void IPBbike::listarBicicletasEmUso()const{                                    3
    ColeccaoHibrida<Bicicleta*>::iterator it;
    for(it=bicicletas.begin(); it!=bicicletas.end(); it++)
        if((*it)->getUtilizador()!=NULL)
            (*it)->print();
}

Utilizador *Bicicleta::getUtilizador() const{return utilizador;}               1

string Utilizador::getNome() const{return nome;}                               1

virtual void Bicicleta::print()const {                                         2
    cout << " " << id << " conduzida por " << utilizador->getNome()<<endl;
}

void BicConv::print()const {                                                   2
    cout << "Bicicleta Convencional";
    Bicicleta::print();
}


void BicElet::print()const {                                                   1
    cout << "Bicicleta Eletrica";
    Bicicleta::print();
}
```

Grupo II
(5 valores)

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

```
OUTPUT 0:
Base()
OUTPUT 1:
Base(Ola)
Derivada(Ola)
OUTPUT 2:
Base()
Derivada()
OUTPUT 3:
classe generica classe especializada classe especializada
OUTPUT 4:
~Derivada()
~Base()
OUTPUT 5:
~Derivada()
~Base()
~Base()
```

b) Apresente o resultado que seria visualizado na saída standard se nenhum dos métodos. [1.0 val.]

```
OUTPUT 3:
```

```
classe generica classe generica classe generica
OUTPUT 4:
~Derivada()
~Base()
```

f) Indique, no contexto da POO, um sinónimo para cada um dos seguintes conceitos: [2.0 val.]

(A) – Instância de uma classe;	objeto
(B) – Instância de uma template de classes;	classe template
(C) – Método virtual;	método polimórfico
(D) – Método virtual puro;	método abstrato
(E) – Membro variável de uma classe;	atributo
(F) – Membro função de uma classe;	método
(G) – Superclasse;	classe base
(H) – Subclasse;	classe derivada
(I) – Derivação de classes;	herança
(J) – Atributo estático;	atributo de classe