Engenharia Informática/Informática de Gestão
Época Recurso – 14 de fevereiro de 2018

Grupo I (14 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [10 val.]

```
-50-
class Aluguer {   -10-                                                    1
   int id;
   TDataHora inicio;
   TDataHora fim;
   Cliente *cliente;                                                      1
   Viatura *viatura;                                                      1
public:
   Aluguer(int id, Cliente *c, Viatura *v) {                              1
      this->id = id;
      inicio = TDataHora::hoje_agora();                                   1
      fim = TDataHora(1, 1, 1900, 0, 0, 0);                               1
      cliente = c;
      viatura = v;
   }
   bool terminar() {
      if (ativo()) {                                                      1
         fim = TDataHora::hoje_agora();                                   1
         return true;
      }else return false;
   }
   bool ativo() const { return fim < inicio; }                            1
   bool operator<(const Aluguer &outro)const { return id < outro.id; }    1
};

class Viatura {   -14-                                                    1
   string matricula;
   Colecao<Aluguer> alugueres;                                            2
public:
   Viatura(string mat) :matricula(mat){}                                  1
   bool alugar(Cliente *c) {                                              5
      int id = alugueres.size() + 1;
      Aluguer a(id, c, this);
      alugueres.insert(a);
      return c->add(alugueres.find(a));
   }
   bool terminarAluguer(int id){                                          4
      Aluguer a(id, NULL, NULL);
      Aluguer *p = alugueres.find(a);
      if (p == NULL) return false; //aluguer inexistente!
      else return p->terminar();
   }                                                                      1
   bool operator<(const Viatura &outra)const{return matricula<outra.matricula;}
};
```

```
class Cliente{   -6-                                                    1
    string carta, nome;
    Colecao<Aluguer*> alugueres;                                        2
public:
    Cliente(string cart, string nom) :carta(cart), nome(nom){}          1
    bool add(Aluguer *a) { return alugueres.insert(a); }                1
    bool operator<(const Cliente &outro)const {return carta < outro.carta;}  1
};

class Autoaluga{   -20-
    Colecao<Viatura> viaturas;                                          2
    Colecao<Cliente> clientes;                                          2

    Cliente *findCliente(string cart) const{                            2
        Cliente c(cart, "");
        return clientes.find(c);
    }
    Viatura *findViatura(string mat) const{                             2
        Viatura v(mat);
        return viaturas.find(v);
    }
public:
    bool addCliente(string cart, string nom) {                          2
        Cliente c(cart, nom);
        return clientes.insert(c);
    }

    bool addViatura(string mat) {                                       2
        Viatura v(mat);
        return viaturas.insert(v);
    }

    bool alugarViatura(string cart, string mat) {                       5
        Cliente *c = findCliente(cart);
        if (c == NULL) return false; //Cliente inexistente!
        else {
            Viatura *v = findViatura(mat);
            if (v == NULL) return false; //Viatura inexistente!
            else return v->alugar(c);
        }
    }

    bool terminarAluguer(string mat, int id) {                          3
        Viatura *v = findViatura(mat);
        if (v == NULL) return false; //Viatura inexistente!
        else return v->terminarAluguer(id);
    }
};
```

b) Acrescente ao problema os métodos que permitam apresentar na saída… [4 val.]

```
   -20-
void Autoaluga::mostrarAlugueresCliente(string cart) const {                2
   Cliente *c = findCliente(cart);                                          1
   if (c != NULL)                                                           1
      c->mostrarAlugueres();                                                1
}

void Cliente::mostrarAlugueres() const {                                    2
   Colecao<Aluguer*>::iterator it;                                          1
   for (it = alugueres.begin(); it != alugueres.end(); it++)                2
      (*it)->print();                                                       2
}

void Aluguer::print()const {                                                2
   cout << "Aluguer " << id << " da viatura " << viatura->getMatricula();   1
   if (ativo())
      cout <<" iniciado a "<<inicio.toString()<<" e ainda decorrer"<< endl; 1
   else cout << " de " << inicio.toString()<< " a " <<fim.toString()<< endl;2
}

string Viatura::getMatricula()const { return matricula; }                   2
```

Grupo II (6 valores)

a) Alguma das classes é abstrata. Justifique. [0.6 val.]

A classe Figura, dado que possui um método virtual puro (abstrato).

b) Diga que método deveria ser constante e porquê. [0.6 val.]

O método print(), dado que não altera o estado do objeto (valor dos atributos).

c) Apresente o resultado que será visualizado na saída standard após a execução do programa. [2.8 val.]

```
Criada Figura
Criado Retangulo
Criada Figura
Criado Círculo
-1-
-2-
Quadrado de lado 3
Com origem em (1,2)
Círculo de raio 6
Com origem em (4,5)
Círculo de raio 12
Com origem em (4,5)
-3-
Xau Quadrado->Xau Figura
Xau Círculo->Xau Figura
-4-
Xau Círculo->Xau Figura
```

d) Apresente o resultado que seria visualizado na saída standard se o método `print()` da classe base não fosse virtual. [0.4 val.]

A única diferença é que deixava de aparecer a frase "Quadrado de lado 3".

e) Tendo em conta as classe definidas no problema, dê uma breve explicação do erro cometido em cada uma das instruções que se seguem: [1.6 val.]

(1) `Figura *fig = new Figura(10,20);`

Sendo Figura uma classe abstrata, a mesma não pode ser instanciada.

(2) `Circulo circulos[10];`

Como a classe Circulo não tem construtor por defeito, não é possível criar-se um array de instâncias dessa classe.

(3) `Quadrado q(3); delete q;`

O operador delete só pode ser usado para desalocar objetos criados de forma dinâmica.

Outra resposta válida: a classe Quadrado não dispõe do construtor de conversão de parâmetro inteiro; na sua instanciação deveria ser invocado o construtor de 3 parâmetros inteiros.

(4) `Circulo c1(1), c2(2); if(c1==c2) cout<< "sao iguais" << endl;`

A classe Circulo não dispõe do operador igualdade (==), nem o mesmo se encontra sobrecarregado como função global para esse tipo de operandos.

Outra resposta válida: a classe Circulo não dispõe do construtor de conversão de parâmetro inteiro; na sua instanciação deveria ser invocado o construtor de 3 parâmetros inteiros.