UNIVERSIDADE FEDERAL DE SANTA CATARINA - Departamento de Automação e Sistemas
PROGRAMAÇÃO DE SISTEMAS AUTOMATIZADOS - 2016/2 - P2

Nome:

---

1)(2 pontos) Em um programa orientado a objetos, a classe Aluno é subclasse da classe Cliente, que por sua vez é subclasse da classe Pessoa. O método buscaDados() da classe Cadastro espera como parâmetro um objeto do tipo Cliente.

É possível invocar este método passando para ele como parâmetro uma referência a objeto instância da classe Aluno? Justifique.

É possível invocar este método passando para ele como parâmetro uma referência a objeto instância da classe Pessoa? Justifique.

2)(2 pontos) Em um programa orientado a objetos a classe Veículo é superclasse das classes Carro e Caminhão. A classe Veículo possui o método void setOdometro(double).

É possível que as classes Carro e Caminhão possuam cada uma um método void setOdometro(double), porém com código diferente (fazendo coisas diferentes)? Justifique.

É possível que a classe Caminhão ainda adicione um método void setOdometro(int), além método void setOdometro(double), mesmo que a classe Veículo não possua tal método? Justifique.

3)(2 pontos) Crie uma classe Gerente que é subclasse de Funcionário. Ela possui um atributo departamento do tipo String e uma gratificação além do salário. O método aumento() agora aplica o fator de aumento na gratificação e no salário. Um construtor permite inicializar todos os atributos. Utilize “super” no novo construtor.

```
public class Funcionario {
String nome;
String cargo;
double salario;
public Funcionario( String n, String c, double s) {
        nome = n;
        cargo = c;
        salario = s;
}
public void print() {
        System.out.println(nome + “-“ + cargo + “-“ + salario);
}
public void aumento( double fator){
        salario = salario * fator;
}
}
```

4)(1 ponto) É possível criar uma classe que herda de duas superclasses ?

5)(1 ponto) O que acontece quando a classe não herda de nenhuma classe via “extends” ?

6)(1 ponto) Qual a diferença entre public, private e o default package ?

7)(1 ponto) Como fazer para a Classe Pessoa abaixo garantir que o atributo idade sempre assuma valores entre zero e 150 ?

```
class Pessoa {

        public int idade;
        ...
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA - Departamento de Automação e Sistemas
PROGRAMAÇÃO DE SISTEMAS AUTOMATIZADOS - 2016/2 - P3

Nome:

QUESTÕES 1, 2 e 3

```
interface Remuneravel {
     void remunera( double taxa);
}

abstract class Conta {
     double saldo = 0;

     void retira( double valor) {
          if( saldo >= valor )
               saldo -= valor;
     }

     void deposita( double valor) {
          saldo += valor;
     }

     double getSaldo() {
          return saldo;
     }

     abstract void cobraTaxa( double taxaFixa);
}


class ContaCorrente extends Conta {
     void cobraTaxa( double taxaFixa) {
          saldo -= taxaFixa;
     }
}


class ContaCorrenteEspecial extends ContaCorrente {

     void retira( double valor) {
          saldo -= valor;
     }
}
```

QUESTÕES 1, 2 e 3

1)(2 pontos) Crie a classe ContaCorrentePoupanca que é subclasse de ContaCorrente e implementa a interface Remuneravel. A lógica da remuneração pode ser qualquer.

2)(2 pontos) Crie uma nova interface "DadosImpostoRenda" com um metodo "ganhosAuferidos" e faça ContaCorrenteEspecial implementar esta nova interface.

3)(1 ponto) Crie uma subclasse da classe Conta que também é abstrata.

QUESTÕES 4, 5 e 6

4)(1 ponto) O que acontece se existirem vários objetos interessados em eventos gerados pelo "MonitorDeChuva" ?

5)(2 pontos) Crie uma classe "ChuvaAdapter" adaptadora para a interface "ChuvaListener".

6)(2 pontos) Crie uma classe "ControladorJanelas" que usa a classe "ChuvaAdapter" como classe interna, e no caso de iniciar chuva forte ele coloca mensagem na tela.

QUESTÕES 4, 5 e 6 – Primeira Parte

```
import java.util.EventObject;

class EventoChuva extends EventObject{
     private boolean chuvaForte;

     EventoChuva(Object source, boolean cf) {
          super(source);
          chuvaForte = cf;
     }

     boolean getChuvaForte() {
          return chuvaForte;
     }
}

interface ChuvaListener {
     void chuvaIniciou(EventoChuva e);
     void chuvaParou(EventoChuva e);
}

class ControladorToldo implements ChuvaListener{
     MonitorDeChuva meuMonitor;

     ControladorToldo( MonitorDeChuva monitor) {
          meuMonitor = monitor;
          meuMonitor.addChuvaListener( this );
     }

     public void chuvaIniciou(EventoChuva e) {
          System.out.println("Chuva " +
           (e.getChuvaForte()?"forte":"") + "iniciou");
     }

     public void chuvaParou(EventoChuva e) {
          System.out.println("Chuva parou");
     }
}
```

QUESTÕES 4, 5 e 6 – Segunda Parte

```
class MonitorDeChuva {
     private int ultimaMedidaChuva = 0;
     private ChuvaListener interessado = null;

     boolean addChuvaListener(ChuvaListener cl) {
          if( interessado == null ) {
               interessado = cl;
               return true;
          }
          else
               return false;
     }

     void geraEventoChuvaIniciou(EventoChuva e) {
          if (interessado != null)
               interessado.chuvaIniciou( e );
     }

     void geraEventoChuvaParou(EventoChuva e) {
          if (interessado != null)
               interessado.chuvaParou( e );
     }

     void medidaChuva(int volume){
          if( volume > 10  &&  ultimaMedidaChuva < 10 ) {
               boolean forte = volume > 50;
               EventoChuva ec = new EventoChuva( this, forte);
               geraEventoChuvaIniciou( ec );
          }
          else if( volume < 10  &&  ultimaMedidaChuva > 10 ) {
               EventoChuva ec = new EventoChuva( this, false);
               geraEventoChuvaParou( ec );
          }
     }

}
```