CI062 – Técnicas alternativas de programação **Prova 1**
2º semestre 2017

Instruções para a prova

- A prova é sem consulta e dura dura 1 hora e 30 minutos;
- Será cobrado o uso de boas práticas de programação de orientação a objetos (OO).

Questão 1: (25 pontos)
Explique o que é sobrecarga e sobrescrita. Dê 2 exemplos de assinaturas de métodos para ilustrar cada conceito.

Questão 2: (20 pontos)

```
abstract class Veiculo {
   public String nome, cor;
}

class Carro extends Veiculo {
   public int ano;
   private peso;
   public void setPeso() {
     return peso;
   }
}
class Programa {
public static void main (String args[]){
  Carro c = new Carro();
  Veiculo v = new Veiculo ();
  c.nome = "Jetta";
  c.cor = "azul";
  v = c;
  Field[] fields =
     v.getClass().getDeclaredFields();
  for (int i =0; i < fields.length; i++)
     System.out.println(
           fields[i].getName());
}
}
```

Dado o programa acima:
a) O código deve ser corrigido, alterando apenas a classe *Programa*, sem exclusão ou inclusão de linhas. Explique o motivo do erro.
b) Mostre o resultado da impressão da execução do programa, após correção. O método *Field[] getDeclaredFields();* retorna a lista de todos os campos de uma classe.

Questão 3 : (30 pontos)
Crie um modelo de classes representando um sistema gerenciador de arquivos. Respeite as especificações abaixo:
- O sistema deverá possuir diretórios e arquivos, sendo que os diretórios podem conter arquivos e/ou diretórios.
- Os arquivos devem possuir nome, tamanho e tipo.
- Os diretórios devem possuir nome.

O sistema de arquivos deverá (1) imprimir o caminho completo (com todos os nomes) de um determinado arquivo. (2) Calcular o tamanho de um diretório, com a soma do tamanho de todos os arquivos filhos.

Observações
O modelo de classe pode ser feito usando uma linguagem de programação orientada a objetos (e.g., Java, C++, Ruby, etc.) ou um diagrama complementado com código. Não é necessário encapsular.
Caso use listas, pode ser usada a classe ArrayList, que já implementa os métodos *void add(Object obj);* para adicionar elementos, *void remove(Object obj);* para remover e *Object get(int n);* para buscar o **n**-èsimo elemento.

Questão 4 : (25 pontos)

Considere o código OO abaixo, que implementa um mecanismo de envio e recepção de mensagens simplificado :

```
class Emissor {
 public boolean mensagem (IReceptor f,
                          String mensagem) {
      (1)//completar
   }
}
interface IReceptor {
  (2)//completar
}
class Receptor implements IReceptor
(3)//completar
```

a) Complemente esta especificação nos pontos (1), (2) e (3) definidos no código, de forma que a mensagem seja impressa tanto no emissor quanto no receptor. O código deve ser extensível para o eventual uso de diferentes receptores.
b) Crie um programa que instancie 1 emissor, 1 receptor, e que simule uma troca de mensagens entre eles.