Centro Federal de EducaÃğÃčo TecnolÃşgica de Minas Gerais
Campus VII - Unidade TimÃşteo - Engenharia da ComputaÃğÃčo

Lista 01 - ProgramaÃğÃčo de Computadores II

Aluno: Egmon Pereira

Orientador: Prof. Douglas Nunes de Oliveira

TimÃşteo, Maio de 2016

1. Qual ÃĽ o padrÃčo de nomeaÃğÃčo de variÃąveis utilizado em praticamente todo o mundo?

   a) Esta pergunta deseja que vocÃł esteja apto a responder se o nome de uma variÃąvel ÃĽ todo maiÃžsculo ou minÃžsculo.

   R: *Primeiro define se a variÃąvel ÃĽ padrÃčo, private, protecte ou public, em seguida se retorno, ou seja, se inteira, String, char, etc., e o nome deve comeÃğar com letra minÃžscula e caso seja nome composto, a primeira letra do segundo nome deve ser maiÃžscula, ex: private int variavelComposta;*

2. Criar uma classe Conta com os atributos: nÃžmero da agencia, nÃžmero da conta e saldo.

```java

public class Conta {

//ATRIBUTOS
    String numAg;
    String numConta;
    String nome;
    String cpf;
    String nasc;
    String end;
    String tel;
    double saldo;
}
```

Lista01/ContaEx2.java

3. Criar um objeto de Conta e alterar os valores de seus atributos.

```java
    public static void main(String[] args) {

        Conta c = new Conta();
        c.numAg = "3131";
        c.numConta = "25.147";
        c.saldo = 15975.32;
    }
```

Lista01/ContaEx3.java

4. Uso do operador *new.Conta c = new Conta();*

   I. Qual a funÃğÃčo/relaÃğÃčo do new com a criaÃğÃčo de um objeto?

   R: *Quando declaramos uma variÃąvel para associar a um objeto, na verdade, essa variÃąvel nÃčo guarda o objeto, e sim uma maneira de acessÃą-lo, chamada de referÃłncia. A palavra* new *instancia uma Classe.*
   O new *, depois de alocar a memÃşria para esse objeto, devolve uma "*$flecha$*", isto Ãľ, um valor de referÃłncia. Quando vocÃł atribui isso a uma variÃąvel, essa variÃąvel passa a se referir para esse mesmo objeto.*

   II. O que ocorre na memÃşria?

   R: *Na memÃşria Ãľ separado um espaÃğo, uma posiÃğÃčo, para que Java possa acessÃą-lo quando necessÃąrio.*

   III. Onde estÃą o objeto?

   R: *Na memÃşria.*

   IV. A variÃąvel c nÃčo Ãľ o objeto, o que ela Ãľ entÃčo?

   R: c *Ãľ uma variÃąvel referÃłncia, apesar de, depois de um tempo, os programadores Java falarem "Tenho um objeto* c *do tipo Conta", mas apenas para encurtar a frase "Tenho uma referÃłncia* c *a um objeto do tipo Conta".*

   V. Como entÃčo eu altero o atributo saldo, do objeto criado, usando o cÃşdigo: $c.saldo = 40$?

   R: *Ao utilizarmos o "." para navegar, o Java vai acessar a Conta que se encontra naquela posiÃğÃčo de memÃşria, e nÃčo uma outra.*

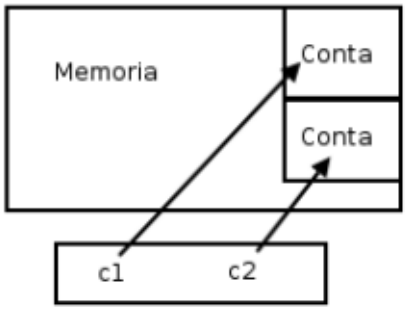
5. Tente escrever com suas palavras o que Ãľ uma classe e o que Ãľ um objeto no paradigma de desenvolvimento orientado a objetos.

R: **Classe** - Ãľ um conjunto de objetos com caracterÃŋsticas comuns.
Objeto - Ãľ um elemento da uma classe.

6. Quantos objetos e variÃąveis sÃčo criadas nos cÃ§digos abaixo; desenhe uma representaÃğÃčo de cada variÃąvel e cada objeto na memÃ§ria; especifique um endereÃğo qualquer para cada objeto:

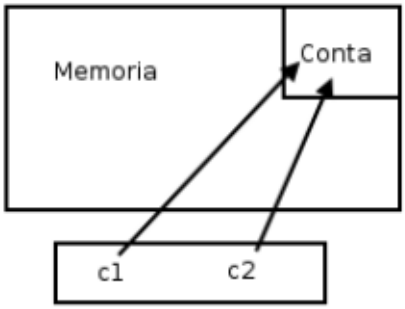
   I. {
   Conta c1 = new Conta();
   Conta c2 = new Conta();
   }

   R: *2 objetos e 1 variÃąvel*
   c1 = Conta@1540e19d e c2 = Conta@677327b6



   II. {
   Conta c1 = new Conta();
   Conta c2 = c1;
   }

   R: *1 objetos e 2 variÃąveis*
   c1 = c2 = Conta@677327b6



Pense em comportamentos que uma conta bancÃąria deve ter: sacar passando um valor, depositar passando algum valor, obter saldo. ÃĽ desta forma que se define os mÃľtodos.

7. Crie os mÃľtodos da classe Conta definida no exercÃņcio anterior.

```
1  public class Conta {
2  //ATRIBUTOS
3
4      String numAg;
5      String numConta;
6      String nome;
7      String cpf;
8      String nasc;
9      String end;
10     String tel;
11     double saldo;
12
13 //METODOS
14     public void saca() {
15
16     }
17
18     public void deposita() {
19
20     }
21
22     public double saldo() {
23         return saldo;
24     }
25 }
```

Lista01/ContaEx7.java

8. Crie um cÃ§digo para testar esta classe. Crie um objeto, imprima o seu saldo, faÃğa um deposito, imprima o seu saldo, faÃğa um saque e imprima o seu saldo.

```java
public class Conta {
//ATRIBUTOS

    String numAg;
    String numConta;
    String nome;
    String cpf;
    String nasc;
    String end;
    String tel;
    double saldo;

//METODOS
    public void saca() {
        saldo -= 6975.32;
    }

    public void deposita() {
        saldo += 1000;
    }

    public double saldo() {
        return saldo;
    }
}
```

Lista01/ContaEx8.java

```java
 public static void main(String[] args) {

        DecimalFormat df = new DecimalFormat("R$0.00");
        Conta c = new Conta();
        c.numAg = "3131";
        c.numConta = "25.147";
        c.saldo = 15975.32;
        System.out.println("Saldo: " + df.format(c.saldo));
        c.deposita();
        System.out.println("Saldo: " + df.format(c.saldo));
        c.saca();
        System.out.println("Saldo: " + df.format(c.saldo));
        System.out.println("");

    }
```

Lista01/ContaEx9.java

```
Saldo: R$15975,32
Saldo: R$16975,32
Saldo: R$10000,00

CONSTRUÃŊDO COM SUCESSO (tempo total: 0 segundos)
```

Lista01/ContaEx10.java

9. Afirmando que uma boa prÃątica Ãľ criar uma mÃľtodo de leitura e escrita para cada um dos atributos. E que os mÃľtodos de leitura de atributos comeÃğam com a palavra get e os mÃľtodos de escrita comeÃğam com a palavra set. Exemplo: getNome() e setNome(String nome). Crie um mÃľtodo de leitura e escrita para cada atributo da classe Conta. FaÃğa isto sem usar o gerador de cÃ§digo da IDE de desenvolvimento.

```
public class Conta {
//ATRIBUTOS

    String numAg;
    String numConta;
    double saldo;

//MÃĽTODOS
    public void saca() {
        saldo -= 6975.32;
    }

    public void deposita() {
        saldo += 1000;
    }

    public double saldo() {
        return saldo;
    }

//GETTERS
    public String getNumAg() {
        return numAg;
    }

    public String getNumConta() {
        return numConta;
    }

    public double getSaldo() {
        return saldo;
    }
```

```
33
34 //SETTERS
35     public void setNumAg(String numAg) {
36         this.numAg = numAg;
37     }
38
39     public void setNumConta(String numConta) {
40         this.numConta = numConta;
41     }
42
43     public void setSaldo(double saldo) {
44         this.saldo = saldo;
45     }
46 }
```

Lista01/ContaEx11.java

10. Toda vez que crio um mÃľtodo e declaro uma variÃąvel com o mesmo nome do atributo da classe ocorre o sombreamento:
public class Aluno{
String nome = "Douglas";
public void setNome(String nome) {
nome = nome;}
}
Caso uma classe Aluno possua o atributo "nome" e o cÃ³digo abaixo seja executado:
public class TesteMain{
public static void main(String [] args) {
Aluno a = new Aluno();
a.setNome("Nunes");
System.out.println(a.nome); }
}
O que serÃą impresso em tela?

R: *SerÃą impresso apenas o nome: Douglas*
A impressÃčo indica que nÃčo houve alteraÃğÃčo no atributo nome, porque?

R: *Porque o apontamento da memÃ³ria Ãĺ diferente, pois o nome Nunes foi colocado em outro endereÃğo de memÃ³ria instanciado pelo set*
Como resolver o problema encontrado?

R: No cÃşdigo:
public void setNome(String nome) {
nome = nome; }
deve ser acrescentado o "this", ficando:
public void setNome(String nome) {
this.nome = nome;
}
e tambÃľm o deve ser acrescentado o cÃşdigo:
public String getNome(){
return nome; }

Na apostila da Caelum, nos exercÃŋcios do capÃŋtulo 4, fazer os exercÃŋcios 1 ao 9.

11. Modele um funcionÃąrio. Ele deve ter o nome do funcionÃąrio, o departamento onde trabalha, seu salÃąrio (double), a data de entrada no banco (String) e seu RG (String).
VocÃł deve criar alguns mÃľtodos de acordo com sua necessidade. AlÃľm deles, crie um mÃľtodo recebeAumento que aumenta o salario do funcionÃąrio de acordo com o parÃćmetro passado como argumento. Crie tambÃľm um mÃľtodo calculaGanhoAnual , que nÃčo recebe parÃćmetro algum, devolvendo o valor do salÃąrio multiplicado por 12.
A ideia aqui ÃĽ apenas modelar, isto ÃĽ, sÃş identifique que informaÃğÃţes sÃčo importantes e o que um funcionÃąrio faz. Desenhe no papel tudo o que um Funcionario tem e tudo que ele faz.

R: **FuncionÃąrio**
- nome do funcionÃąrio String;
- departamento String;
- salÃąrio double;
- entrada String
- RG String

+ recebeAumento(); void
+ calculaGanhoAnual(); double

12. Transforme o modelo acima em uma classe Java. Teste-a, usando uma outra classe que tenha o main .
VocÃł deve criar a classe do funcionÃąrio com o nome Funcionario , mas pode nomear como quiser a classe de testes, contudo, ela deve possuir o mÃľtodo main.

```
package javaapplication21;

public class Funcionario {

    private String nomedofuncionario;
    private String departamento;
    private double salario;
    private String entrada;
    private String RG;

    public boolean recebeAumento(double salario) {
        if (salario > 0) {
            this.salario += salario;
            return true;
        }
        return false;
    }

    public double calculaGanhoAnual() {
        double calcula;
        calcula = salario * 12;
        return calcula;
    }

    public String getNomedofuncionario() {
        return nomedofuncionario;
    }

    public void setNomedofuncionario(String
        nomedofuncionario) {
        if (nomedofuncionario != null) {
            this.nomedofuncionario = nomedofuncionario;
        }
    }

    public String getDepartamento() {
        return departamento;
    }

    public void setDepartamento(String departamento) {
        if (departamento != null) {
            this.departamento = departamento;
```

```
42          }
43      }
44
45      public double getSalario() {
46          return salario;
47      }
48
49      public void setSalario(double salario) {
50          if (salario > 0) {
51              this.salario = salario;
52          }
53      }
54
55      public String getEntrada() {
56          return entrada;
57      }
58
59      public void setEntrada(String entrada) {
60          if (entrada != null) {
61              this.entrada = entrada;
62          }
63      }
64
65      public String getRG() {
66          return RG;
67      }
68
69      public void setRG(String RG) {
70          if (RG != null) {
71              this.RG = RG;
72          }
73      }
74
75  }
```

Lista01/Funcionario.java

```java
package javaapplication21;

import java.text.DecimalFormat;

public class Ex02 {

    public static void main(String[] args) {
        DecimalFormat df = new DecimalFormat("R$0,000.00");
        Funcionario f1 = new Funcionario();
        f1.setNomedofuncionario("LeoAndro");
        f1.setRG("MG 5.789.951");
        f1.setEntrada("21/09/1982");
        f1.setDepartamento("Faz Nada");
        f1.setSalario(1234.66);

        System.out.println("SalÃąrio Inicial: "+df.format(f1
            .getSalario()));
        System.out.println("");
        f1.recebeAumento(f1.getSalario());
        f1.calculaGanhoAnual();
        System.out.println("SalÃąrio Com Aumento: "+df.
            format(f1.getSalario()));
    }
}
```

Lista01/Ex02.java

TELA:
run:
SalÃąrio Inicial: R$1.234,66

SalÃąrio Com Aumento: R$2.469,32
CONSTRUÃŊDO COM SUCESSO (tempo total: 0 segundos)

13. Crie um mÃľtodo mostra(), que nÃčo recebe nem devolve parÃćmetro algum e simplesmente imprime todos os atributos do nosso funcionÃąrio. Dessa maneira, vocÃł nÃčo precisa ficar copiando e colando um monte de $System.out.println()$ para cada mudanÃğa e teste que fizer com cada um de seus funcionÃąrios, vocÃł simplesmente vai fazer:

R: *A SaÃŋda foi diferente, pois nÃčo foram comparados os atributos, mas sim os objetos, que sÃčo diferentes, pois possuem diferentes endereÃğos de memÃşria.*

14. Construa dois funcionÃąrios com o new e compare-os com o ==. E se eles tiverem os mesmos atributos? Para isso vocÃł vai precisar criar outra referÃłncia:

R: *A SaÃŋda foi igual, pois nÃčo foi criado um objeto diferente, mas usado o mesmo para f1 e f2.*

15. Crie duas referÃłncias para o mesmo funcionÃąrio, compare-os com o == . Tire suas conclusÃţes. Para criar duas referÃłncias pro mesmo funcionÃąrio;
O que acontece com o if do exercÃŋcio anterior?

R: *Por que na 3 criamos dois objetos diferentes. E na 4 ao invÃľs da variÃąvel receber um novo endereÃğo, recebeu um endereÃğo que jÃą existia.*

16. Modele e implemente uma classe chamada ParDeDados, composta por dois dados de seis lados e um mÃľtodo rolar. Crie uma classe TestaParDeDados com um mÃľtodo main que irÃą "lanÃğar" (rolar) um objeto ParDeDados mÃžltiplas vezes contando o nÃžmero de vezes que aparece o nÃžmero 6 em ambos os dados.

```
1  package testapardedados;
2
3  import java.util.Random;
4  import javax.swing.JOptionPane;
5
6  public class ParDeDados {
7
8      Random ram = new Random();
9
10 //ATRIBUTOS
11     int d = Integer.parseInt(JOptionPane.showInputDialog("
           Digite quantas vezes os Dados irao Rolar"));
12     int SIZE = d;
13     int[] dado1 = new int[SIZE];
14     int[] dado2 = new int[SIZE];
15     int cont = 0;
16
17 //METODO
18     public void rolar() {
19
20         for (int i = 0; i < d; i++) {
21             dado1[i] = ram.nextInt(6) + 1;
22             if (dado1[i] == 6) {
23                 cont++;
24             }
```

```
25          }
26
27          for (int i = 0; i < d; i++) {
28              dado2[i] = ram.nextInt(6) + 1;
29              if (dado2[i] == 6) {
30                  cont++;
31              }
32          }
33          System.out.println("Foram lancados " + cont + "
                numeros 6.");
34          int aux = Integer.parseInt(JOptionPane.
                showInputDialog("Deseja imprimir todos os
                numeros sorteados? \n0 - SIM \n1 - nÃčo"));
35
36          if (aux == 0) {
37              System.out.println("Dado 01");
38              for (int i = 0; i < d; i++) {
39                  System.out.print(dado1[i] + " ");
40              }
41              System.out.println("");
42              System.out.println("Dado 02");
43              for (int i = 0; i < d; i++) {
44                  System.out.print(dado2[i] + " ");
45              }
46
47          }
48              System.out.println("");
49      }
50
51  }
```

Lista01/ContaEx16.java

17. AtravÃľs da manipulaÃğÃčo genÃľtica, biÃłlogos criaram uma bactÃľria que come lixo. Esta consome o dobro de seu peso em lixo por hora. O peso da bactÃľria Ãľ constante (*cerca de 10 picogramas*), o lixo Ãľ totalmente metabolizado. De 3 em 3 horas cada bactÃľria se dividem em duas. Uma bactÃľria vive apenas 25 horas e depois morre.
    Modele a classe Bacteria. Ela deve representar adequadamente o estado de uma bactÃľria (*tempo de vida, lixo metabolizado, etc*). Deve ter mÃľtodos para retornar quanto lixo ela comeu, simular a passagem de uma hora (reduzindo o seu tempo de vida) e retornar o tempo de vida restante.
    Crie uma classe Colonia que representa uma colÃťnia de bactÃľrias (vetor). A classe Colonia deve ter mÃľtodos que: simula a passagem de horas (das bactÃľrias), retorna a quantidade de bactÃľrias existentes e o total de lixo consumido. FaÃğa um programa de teste que simula a passagem de N

horas.

```java
package bacteria;

import javax.swing.JOptionPane;

public class Bacteria {

    public static void main(String[] args) {
        Colonia l = new Colonia();
        l.setHora(Integer.parseInt(JOptionPane.
            showInputDialog("Digite quantas horas irÃą durar
             o processo!")));
        l.colonia();
    }

}
```

Lista01/Bacterias.java

```java
package bacteria;

import java.text.DecimalFormat;
import javax.swing.JOptionPane;

public class Colonia {

    DecimalFormat bf = new DecimalFormat("0,000");
//ATRIBUTOS
    private double lixo;
    private int bacterias, hora, cont, aux;

//METODOS
    public void setHora(int hora) {
        this.hora = hora;
    }

    public int getHora() {
        return hora;
    }

    public void colonia() {
        if (hora > 113) {
            JOptionPane.showMessageDialog(null, "VocÃł
                digitou " + hora + " Horas \nFavor digitar
                um valor abaixo de 114 horas!\nSeu programa
                serÃą encerrado!");
        } else {
            bacterias = 1;
```

```
27              for (int i = 0; i < hora; i++) {
28                  if ((i + 1) % 3 == 0) {
29                      cont++;
30                      bacterias = (int) Math.pow(2, cont);
31                  }
32                  if ((i + 1) % 3 == 0 && (i + 1) >= 24) {
33                      aux++;
34                      bacterias -= (int) Math.pow(2, aux);
35                  }
36                  lixo += bacterias * 10 * 2;
37              }
38              if (hora > 29) {
39                  System.out.println("Quantidade de BactÃľrias
                        apÃ§s: " + hora + " Horas = " + bf.
                        format(bacterias) + " BactÃľrias Vivas")
                        ;
40              } else {
41                  System.out.println("Quantidade de BactÃľrias
                        apÃ§s: " + hora + " Horas = " +
                        bacterias + " BactÃľrias Vivas");
42              }
43
44              if (hora > 11) {
45                  System.out.println("Quantidade de Lixo
                        consumido em " + hora + " Horas = " + bf
                        .format(lixo) + " picogramas");
46              } else {
47                  System.out.println("Quantidade de Lixo
                        consumido em " + hora + " Horas = " +
                        lixo + " picogramas");
48              }
49              System.out.println("");
50          }
51      }
52  }
```

Lista01/Colonia.java

ExercÃŋcios para entender heranÃğa, reescrita e polimorfismo.

18. Criar as classes "Pessoa", "Funcionario", e "Cliente".

```java
package empresa;

public class Pessoa {

}
```

Lista01/Pessoa2.java

```java
package empresa;

public class Funcionario {

}
```

Lista01/Funcionario2.java

```java
package empresa;

public class Cliente {

}
```

Lista01/Cliente2.java

19. FuncionÃąrio tem os atributos nome, cpf e rg, que sÃčo comuns com a classe Cliente.

```java
package empresa;

public class Funcionario {
//ATRIBUTOS

    private String nome;
    private String cpf;
    private String rg;
}
```

Lista01/Funcionario2b.java

20. Criar 3 variÃąveis do tipo Pessoa.
Pessoa p1 = new Pessoa();
Pessoa p2 = new Funcionario();
Pessoa p3 = new Cliente();

```java
package empresa;

public class Empresa {

    public static void main(String[] args) {

Pessoa p1 = new Pessoa();
Pessoa p2 = new Funcionario();
Pessoa p3 = new Cliente();
    }

}
```

Lista01/Empresa2.java

21. Verificar se houve erro em umas das linhas da atividade 4. Se houve ou nÃčo erros, porque?

```java
package empresa;

public class Empresa {

    public static void main(String[] args) {

Pessoa p1 = new Pessoa(); //NÃčo houve erro, pois a
    variÃąvel p1 Ãľ do tipo "Pessoa"
Pessoa p2 = new Funcionario(); //Houve erro nessa linha,
    pois p2 nÃčo pode ser do tipo "Pessoa" e sim do tipo "
    FuncionÃąrio"
Pessoa p3 = new Cliente(); //Houve erro nessa linha, pois p3
     nÃčo pode ser do tipo "Pessoa" e sim do tipo "Cliente"
    }

}
```

Lista01/Empresa2b.java

22. Criar duas variÃąveis dos tipos: Funcionario e Cliente:
Funcionario f = new Pessoa();
Cliente c = new Pessoa();

```java
package empresa;

public class Empresa {

    public static void main(String[] args) {

Pessoa p1 = new Pessoa();
Funcionario f = new Pessoa();
Cliente c = new Pessoa();

}
```

Lista01/Empresa2c.java

23. As linha da atividade 6 ocorreram erros? Porque?

```java
package empresa;

public class Empresa {

    public static void main(String[] args) {

Pessoa p1 = new Pessoa(); //NÃčo houve erro, pois p1 Ãľ do tipo "Pessoa"
Funcionario f = new Pessoa();  //Houve erro, pois f deve ser do tipo "Pessoa"
Cliente c = new Pessoa(); //Houve erro, pois c deve ser do tipo "Pessoa"

}
```

Lista01/Empresa2d.java

24. Porque as linhas abaixo nÃčo apresentam erros se o mÃľtodo toString() nÃčo existe em nenhuma das classes.
Pessoa p = new Pessoa();
Funcionario f = new Funcionario();
Cliente c = new Cliente();
Sout(f.toString());
Sout(c.toString());
Sout(p.toString());

R: NÃčo hÃą erro, pois o "*toString*" Ãľ um mÃľtodo herdado do "*Object*".

25. Todas as classes herdam de Object?

R: Sim

26. Se uma classe Endereco for declarada abaixo:
public class Endereco{
//Atributos
//MÃľtodos }
Esta classe estÃą herdando de alguma outra classe? PorquÃł?

R: NÃčo, pois para herdar de outra classe seria necessÃąrio o cÃ§digo ser reescrito da seguinte forma:
public class Endereco extends NomeDaClasseHerdada{
//Atributos
//MÃľtodos }

27. Declarar um vetor de 3 posiÃğÃ§es do tipo Pessoa. Instanciar 2 objetos de Funcionario e 1 de Cliente. Colocar suas referencias no vetor. Percorrer o vetor imprimindo o atributo nome de cada um dos objetos.

```java
package empresa;

public class Empresa {

    public static void main(String[] args) {

        Pessoa vetor[] = new Pessoa[3];
        Pessoa f1 = new Funcionario();
        Pessoa f2 = new Funcionario();
        Pessoa c1 = new Cliente();

        f1.setNome("Egmon");
        f2.setNome("Gabriel");
        c1.setNome("Douglas");

        vetor[0] = f1;
        vetor[1] = f2;
        vetor[2] = c1;

        for (int i = 0; i < vetor.length; i++) {
            System.out.println("Nome: " + vetor[i].getNome()
                );
        }
    }
}
```

Lista01/Empresa2e.java

```java
package empresa;

public class Pessoa {
//ATRIBUTOS

    private String nome;
    private String cpf;
    private String rg;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;

}
```

Lista01/Pessoa2e.java

```java
package empresa;

public class Funcionario extends Pessoa{


  }
```

Lista01/Funcionario2e.java

```java
package empresa;

public class Cliente extends Pessoa{

}
```

Lista01/Cliente2e.java

ExercÃŋcios para entender heranÃğa e construtores.

28. Alterar a classe Pessoa da atividade anterior. Criar um construtor que receba como parÃćmetro: nome, cpf e rg. Desta forma ignorando o construtor padrÃčo.

```java
package pessoa;

public class Pessoa {
//ATRIBUTOS

    private String nome;
    private String cpf;
    private String rg;

    public Pessoa(String nome, String cpf, String rg){

    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Lista02/Pessoa.java

29. Ocorreu algum erro nas classes filhas? Porque?

R: Sim. Porque nas classes filhas nÃčo estÃčo sendo chamdados os parÃćmetros do *Construtor*.

30. Corrigir o problema, alterando as classes filhas de Pessoa. Visto que o construtor padrÃčo (fornecido pelo Java) das filhas nÃčo atendem.

```java
package pessoa;

public class Cliente extends Pessoa {

    public Cliente(String nome, String cpf, String rg) {
        super(nome, cpf, rg);
    }

}
```

Lista02/Cliente.java

```java
package pessoa;

public class Funcionarios extends Pessoa {

    public Funcionarios(String nome, String cpf, String rg)
        {
        super(nome, cpf, rg);
    }

}
```

Lista02/Funcionarios.java

31. A classe FuncionÃąrio possui codFuncionario, criar dois construtores: um recebendo nome, CPF e RG; outro recebendo codfuncionario, nome, CPF e RG.

```java
package pessoa;

public class Funcionarios extends Pessoa {

    public Funcionarios(int codFuncionarios, String nome,
        String cpf, String rg) {
        super(codFuncionarios, nome, cpf, rg);
    }

}
```

Lista02/Funcionarios1.java

32. Idem Ãă atividade 4 porÃľm envolvendo a classe Cliente.

```java
package pessoa;

public class Cliente extends Pessoa {

    public Cliente(int codCliente, String nome, String cpf,
        String rg) {
        super(codCliente, nome, cpf, rg);
    }

}
```

Lista02/Cliente1.java

33. Conceitualmente, Pessoa ou Cliente ou FuncionÃąrio pode herdar de EndereÃğo? Ou EndereÃğo pode herdar de Pessoa ou Cliente ou FuncionÃąrio? Porque?

R: NÃčo. Porque **Endereco** nÃčo Ãľ uma **Pessoa**, nem um **Cliente** e nem um **Funcionarios**. Analogamente, **Pessoa**, **Cliente** e **FuncionÃąrios** nÃčo sÃčo **Endereco**.

34. Se a relaÃğÃčo da classe Endereco com Pessoa, Cliente ou Funcionario nÃčo Ãľ de heranÃğa, entÃčo qual Ãľ a relaÃğÃčo?

R: RelaÃğÃčo de ComposiÃğÃčo.

35. Como resolver em OrientaÃğÃčo a Objetos para Cliente e FuncionÃąrio possuir endereÃğo. Codifique.

R: Basta instanciar as Classes **Funcionarios** e **Cliente**:

```java
package pessoa;

public class Funcionarios extends Pessoa {

    public Funcionarios(int codFuncionarios, String nome,
        String cpf, String rg) {
        super(codFuncionarios, nome, cpf, rg);
    }

    Endereco e = new Endereco();
}
```

Lista02/Funcionarios2.java

```java
package pessoa;

public class Cliente extends Pessoa {

    public Cliente(int codCliente, String nome, String cpf,
        String rg) {
        super(codCliente, nome, cpf, rg);
    }

    Endereco e = new Endereco();
}
```

Lista02/Cliente2.java

Classes Abstratas, Interfaces e tratamento de exceÃğÃ¸es.

36. Crie uma hierarquia de animais cujo ancestral em comum Ãľ a classe abstrata **Animal**. Algumas subclasses da classe animal implementarÃčo a interface chamada **AnimalDeEstimacao**. Treinaremos com variaÃğÃ¸es destes animais, seus mÃľtodos e polimorfismo.

36.1 Crie a classe Animal, que Ãľ a superclasse abstrata de todos os animais;

a. Declare um atributo inteiro protected com o nome numeroDePernas, que armazena o nÃžmero de pernas do animal;

b. Declare um atributo protected do tipo String representando o nome do animal;

c. Defina um construtor protected que inicia o atributo numeroDePernas;

d. Declare o mÃľtodo abstrato comer;

e. Declare o mÃľtodo concreto caminhar que exibe uma mensagem do tipo "**Anda com <numero> pernas.**";

36.2 Crie mÃľtodos **get/set** conforme diagrama. Crie a classe Aranha.

a. A classe Aranha herda da classe Animal;

b. Defina um construtor que chama o construtor da superclasse para especificar que todas aranhas possuem 8 pernas;

c. Implemente o mÃľtodo comer. Crie a interface **AnimalDeEstimacao** especificada no diagrama **UML**.

36.3 Crie **classe Gato** que herda de **Animal** e implementa **AnimalDeEstimacao**.

a. Defina um construtor que recebe um parÃćmetro do tipo String que especifica o nome do gato. Este construtor deve chamar o construtor de sua superclasse para especificar que todos gatos possuem 4 pernas.

b. Defina outro construtor que nÃčo recebe parÃćmetros. Dentro deste construtor chame o construtor anterior (utilizando a palavra reservada this) e passe uma string vazia como argumento.

c. Implemente o mÃľtodo da interface **AnimalDeEstimacao**.

d. Implemente o mÃľtodo **comer**.

36.4 Crie a classe **Peixe** que herda de **Animal**. Reescreva os mÃľtodos de **Animal** para especificar que **Peixe** nÃčo anda e nÃčo possui **pernas**

36.5 Crie um programa chamado **TestaAnimais**. Dentro do mÃľtodo *main*, crie e manipule instÃćncias das classes criadas acima. Inicie com:

Peixe p = new Peixe();

Gato g = new Gato(?Tom?);

Animal a = new Peixe();

Animal ab = new Aranha();

AnimalDeEstimacao ae = new Gato();

Experimente utilizar:

a. Chamando mÃľtodos em cada objeto,

b. Fazendo "**casting**" de objetos (conversÃ£es),

c. utilizando polimorfismo, e

d. utilizando super para chamar mÃľtodos da superclasse.

```
1  package animal;
2
3  public abstract class Animal {
4
5      //ATRIBUTOS
6      protected String nome;
7      protected int numPernas;
8
9      //CONSTRUTOR
10     protected Animal(int numPernas) {
11
12     }
13
14     //METODOS
15     public abstract void comer();
16
17     public abstract void caminhar();
18
19     public String getNome() {
20         return nome;
21     }
22
23     public void setNome(String nome) {
```

```
24          this.nome = nome;
25      }
26
27      public int getNumPernas() {
28          return numPernas;
29      }
30
31      public void setNumPernas(int numPernas) {
32          this.numPernas = numPernas;
33      }
34
35  }
```

Lista02/Animal.java

```
1  package animal;
2
3  public interface AnimalDeEstimacao {
4
5      public abstract void brincar();
6  }
```

Lista02/AnimalDeEstimacao.java

```
1  package animal;
2
3  public class Aranha extends Animal {
4
5      public Aranha(String nome) {
6          super(8);
7          this.nome = nome;
8      }
9
10     public Aranha() {
11         this("Matilde");
12     }
13
14     @Override
15     public void caminhar() {
16         System.out.println("A Aranha anda com " +
               getNumPernas() + " pernas.");
17
18     }
19
20     public void comer() {
21         System.out.println("A Aranha come insetos");
22         System.out.println("");
23     }
```

```
}
```

Lista02/Aranha.java

```
package animal;

public class Gato extends Animal implements
    AnimalDeEstimacao {

    public Gato(String nome) {
        super(4);
        this.nome = nome;
    }

    public Gato() {
        this("Bixano");
    }

    @Override
    public void caminhar() {
        System.out.println("O Gato anda com " + getNumPernas
            () + " pernas.");
    }

    public void comer() {
        System.out.println("O Gato come carne e bebe leite."
            );
        System.out.println("");
    }

    public void brincar() {

    }
}
```

Lista02/Gato.java

```
package animal;

public class Peixe extends Animal {

    public Peixe(String nome) {
        super(0);
        this.nome = nome;
    }

    public Peixe() {
        this("Nemo");
```

```
12      }
13
14      @Override
15      public void caminhar() {
16          System.out.println("O Peixe nada com " +
                getNumPernas() + " pernas.");
17      }
18
19      public void comer() {
20          System.out.println("O Peixe come raÃğÃčo.");
21          System.out.println("");
22      }
23  }
```

Lista02/Peixe.java

```
1   package animal;
2
3   public class TesteAnimais {
4
5       public static void main(String[] args) {
6           // TODO code application logic here
7           Peixe p = new Peixe();
8           Gato g = new Gato();
9           Aranha a=new Aranha();
10          Animal ab = new Peixe();
11          Animal ac = new Aranha();
12          AnimalDeEstimacao ae = new Gato();
13
14          p.caminhar();
15          p.comer();
16          g.caminhar();
17          g.comer();
18          a.caminhar();
19          a.comer();
20          ab.caminhar();
21          ab.comer();
22          ac.caminhar();
23          ac.comer();
24
25      }
26
27  }
```

Lista02/TesteAnimais.java

37. Crie um programa que receba 2 nÃžmeros inteiros e imprima o resultado da divisÃčo do primeiro pelo segundo. O programa deverÃą tratar algumas exceÃğÃțes (nÃčo deixando o programa finaliza com erro) como : o usuÃąrio entrar com valores nÃčo inteiros e ocorrÃłncia de divisÃčo por zero.

```java
package divisao;

import javax.swing.JOptionPane;

public class Divisao {

    public static void main(String[] args) {

        int a, b, aux;
        double d;

        do {
            try {
                aux = 0;
                a = Integer.parseInt(JOptionPane.
                    showInputDialog("Digite um valor:"));
                b = Integer.parseInt(JOptionPane.
                    showInputDialog("Digite um valor
                    diferente de ZERO"));

                d = a / b;

                System.out.println("A divisÃčo de " + a + "
                    por " + b + " = " + d);

            } catch (Exception e) {
                JOptionPane.showMessageDialog(null, "Valor
                    invÃąlido.\nTente novamente:");
                aux = 1;
            }

        } while (aux == 1);
    }
}
```

Lista02/Divisao.java

*"Nenhum homem diz "*Deus nÃčo existe*", a nÃčo ser*

aquele que tem interesse em que ele nÃčo exista"
(Agostinho)