

Herança (exemplo)

Simplificando, podemos afirmar que, uma pessoa é alguém de quem sabemos o nome, o sexo e a nacionalidade.

Programa a classe Pessoa com os construtores, seletores e modificadores que considerar necessários bem como com os métodos públicos toString, clone e equals (ainda que pouco naturais para uma pessoa!).

```
public class Pessoa{

    final public static String MAS = "Masculino";
    final public static String FEM = "Feminino";

    private String nome;
    private String sexo;
    private String nacionalidade;

    // construtores

    public Pessoa(String nome, String sexo) {

        this.nome = nome;

        this.sexo = sexo;
        nacionalidade = new String("Portuguesa");
    }

    public Pessoa(String nome, String sexo, String
                                nacionalidade) {

        //uso explícito do construtor anterior
        this(nome, sexo);

        this.nacionalidade = new String(nacionalidade);
    }
}
```

```
//Construtor de cópia

public Pessoa(Pessoa p) {

    //uso explícito do construtor anterior
    this( p.nome, p.sexo, p.nacionalidade);
}

public Object clone() {
    return new Pessoa(this);
}

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

public String getNome(){
    return nome;
}

public String getNacionalidade(){
    return nacionalidade;
}

public String getSexo(){
    return sexo;
}

public String toString(){
    return "NOME: " + nome + ";\t SEXO: " + sexo +
           ";\t NAC.: " + nacionalidade;
}

public boolean equals (Object o){

... //exercício
}
}
```

Programa a classe Amigo¹. Um amigo é uma pessoa de quem sabemos a data de nascimento, o ano em que o conhecemos, um contacto, o nível de amizade que por ela nutrimos e ainda o “parceiro” com quem normalmente “anda”.

```
import java.util.GregorianCalendar;

public class Amigo extends Pessoa{

    final public static int EXCELENTE = 19;
    final public static int BOM = 16;
    final public static int NORMAL = 12;
    final public static int DA_ONCA = 5;

    private GregorianCalendar dataNasc;
    private String contacto;
    private int anoConhec;
    private int nivelAmiz;

    private Pessoa parceiro;

    public Amigo(Pessoa p) {
        super(p);

        anoConhec = new
            GregorianCalendar().get(GregorianCalendar.YEAR);

        nivelAmiz = NORMAL;

        contacto = null; parceiro = null; dataNasc = null;
    }
}
```

¹ Exercício adaptado de “Programação com classes em C++”, Pedro Guerreiro, FCA, 2000

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

```
public Amigo(Pessoa a, int anoConhec, int nivelAmiz) {  
    super (a) ;  
    this.anoConhec = anoConhec;  
    this.nivelAmiz = nivelAmiz;  
    parceiro = null; contacto = null; dataNasc = null;  
}
```

```
public Amigo(Pessoa a, int anoConhec, int nivelAmiz,  
             Pessoa p) {  
    this(a, anoConhec, nivelAmiz);  
    if (p != null)  
        //clone de p (mas o que é p?)  
        parceiro = (Pessoa) p.clone();  
  
        //parceiro = p; //será que é isto que queremos?  
}
```

//Construtor de cópia

```
public Amigo(Amigo copia) {  
  
    this(copia, copia.anoConhec, copia.nivelAmiz,  
         copia.parceiro);  
  
    contacto = copia.contacto;  
  
    if (copia.dataNascConhecida())  
        dataNasc =  
            (GregorianCalendar) copia.getDataNasc().clone();  
}
```

```
public boolean dataNascConhecida(){ return dataNasc !=  
null; }
```

```
public GregorianCalendar getDataNasc(){ return  
dataNasc; }
```

```
public void setContacto(String contacto){  
    this.contacto = contacto;  
}
```

```
public String getContacto() {
    return contacto;
}

public void setAnoConhec(int ano) {
    this.anoConhec = ano;
}

public int getAnoConhec() {
    return anoConhec;
}

public int duracaoConhec() {

    return
    (new GregorianCalendar().get(GregorianCalendar.YEAR)
     -
     this.anoConhec);
}

public int getNivelAmiz() {
    return nivelAmiz;
}

public void incNivelAmiz(int inc) {
    this.nivelAmiz += inc;
}

public void decNivelAmiz(int inc) {
    this.nivelAmiz -= inc;
}

public boolean melhorAmigoQue (Amigo outro) {

    return  nivelAmiz > outro.nivelAmiz ||
           nivelAmiz == outro.nivelAmiz &&
           anoConhec < outro.anoConhec;
}
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

```
public void setDataNasc(GregorianCalendar data){
    this.dataNasc = (GregorianCalendar) data.clone();
}

public int idade(){
    //PRE: dataNascConhecida()
    GregorianCalendar gc = new GregorianCalendar();
    int idade =
        gc.get(GregorianCalendar.YEAR) -
            dataNasc.get(dataNasc.YEAR);
    if (gc.get(gc.DAY_OF_YEAR) <
dataNasc.get(dataNasc.DAY_OF_YEAR))
        return idade-1;

    return idade;
}

public boolean solteiro(){
    return this.parceiro == null;
}

public boolean casadoAmigo(){
    return !solteiro() && parceiro instanceof Amigo;
}

public Pessoa getParceiro(){ return this.parceiro; }

public void casa(Pessoa p){
    this.parceiro = (Pessoa) p.clone();
}

public void divorcio(){ this.parceiro = null; }
```

```
public Object clone() { return new Amigo(this); }

//método auxiliar
private String getDataFormatada() {
    return ( dataNascConhecida() ?
        dataNasc.get(dataNasc.YEAR) + "/"
        +(dataNasc.get(dataNasc.MONTH)+1)+ "/"
        + dataNasc.get(dataNasc.DAY_OF_MONTH) :
        "desconhecida" );
}

public String toString(){
    return
        "DADOS@AMIGO\n" +
        super.toString() +
        "\nCONHECI EM " + anoConhec + ";\tÉ AMIGO NOTA "
        + nivelAmiz +
        ";\t CONTACTO: " + ( contacto == null ?
        "perdido" : contacto ) + ";\t ANIVERSÁRIO: " +
        getDataFormatada() +
        (solteiro() ? "" : "\nPARCEIRO: " +
        parceiro.getNome() );

    //com parceiro.toString() poderia surgir recursividade
    infinita
    //O parceiro do parceiro do objecto seria o próprio
    objecto
}
}
```

Pequenos (ou nem tanto) exemplos de utilização:

```
public class TesteAmigo {
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

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

    Amigo a1 = new Amigo( new Pessoa("Maria Só Amadeu",
Pessoa.FEM), 1998, Amigo.NORMAL,
                        new Pessoa("C. Amadeu", Pessoa.MAS));

    a1.setContacto("maria@vaicom.asoutras.pt");
    System.out.println("\n" + a1.toString() + "\n");

    a1.divorcio();

    a1.setNome("Maria Só");//onde está o método setNome?

    a1.incNivelAmiz(5);

    a1.setDataNasc( new GregorianCalendar(1975,
        GregorianCalendar.NOVEMBER , 28)); //MESES
        de 0 a 11

    System.out.println (
        "\n" + a1 + "\nIDADE: " + a1.idade()
+ "\n");

    Amigo a2 = new Amigo(new Pessoa("José Silva",
Pessoa.MAS), 1984, Amigo.BOM, a1);
//que vai acontecer no interior de a2?

    a2.setContacto("275123456");

    a2.setNome("José Silva Só");

    System.out.println("\n" + a2 + "\n");
// omissão de toString()!?!

    a1.casa(a2);
    a2.divorcio();
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

```
Pessoa p1 = new Pessoa("Amadêncio Gaudeu",
                        Pessoa.MAS);

a1.casa(p1);
a1.setNome("Maria Só Gaudeu");

System.out.println("\nCasada com amigo ?" +
a1.casadoAmigo());

p1 = new Amigo(p1, 2002, Amigo.NORMAL, a1);

//que acontece aqui ?

System.out.println("\nCasada com amigo ?" +
                    a1.casadoAmigo());

a1.casa(p1);

System.out.println("\nCasada com amigo ?" +
                    a1.casadoAmigo());

System.out.println("\n\nTESTE DE AMIZADE\n" +
                    ( a1.melhorAmigoQue(a2) ? a1 : a2) + "\n");

System.out.println("\nTESTE DE CÓPIA DE
                    REFERÊNCIAS");

System.out.print("a.c. A2: " + a2.getAnoConhec());

Amigo a4 = a2;

a4.setAnoConhec(1900);

System.out.print("\t\t a.c. A4: " +
                    a4.getAnoConhec());
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

```
System.out.print("\t\t a.c. A2: " +
                a2.getAnoConhec());
System.out.println("\n");

System.out.println("\nTESTE DE CLONAGEM DE
OBJECTOS");

a4 = (Amigo)a1.clone();
System.out.print("a.c. A1: " + a1.getAnoConhec());
a4.setAnoConhec(2000);
System.out.print("\t\t a.c. A4: " +
                a4.getAnoConhec());
System.out.print("\t\t a.c. A1: " +
                a1.getAnoConhec());
}
}
```

DADOS@AMIGO
NOME: Maria Só Amadeu; SEXO: Feminino; NAC.: Portuguesa
CONHECI EM 1998; É AMIGO NOTA 12; CONTACTO: maria@vaicom.asoutras.pt; ANIVERSÁRIO:
desconhecida
PARCEIRO: C. Amadeu

DADOS@AMIGO
NOME: Maria Só; SEXO: Feminino; NAC.: Portuguesa
CONHECI EM 1998; É AMIGO NOTA 17; CONTACTO: maria@vaicom.asoutras.pt; ANIVERSÁRIO:
1975/11/28
IDADE: 37

DADOS@AMIGO
NOME: José Silva Só; SEXO: Masculino; NAC.: Portuguesa
CONHECI EM 1984; É AMIGO NOTA 16; CONTACTO: 275123456; ANIVERSÁRIO: desconhecida
PARCEIRO: Maria Só

Casada com amigo ?false
Casada com amigo ?false
Casada com amigo ?true

TESTE DE AMIZADE
DADOS@AMIGO
NOME: Maria Só Gaudeu; SEXO: Feminino; NAC.: Portuguesa
CONHECI EM 1998; É AMIGO NOTA 17; CONTACTO: maria@vaicom.asoutras.pt; ANIVERSÁRIO:
1975/11/28
PARCEIRO: Amadêncio Gaudeu

TESTE DE CÓPIA DE REFERÊNCIAS

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

a.c. A2: 1984	a.c. A4: 1900	a.c. A2: 1900
TESTE DE CLONAGEM DE OBJECTOS		
a.c. A1: 1998	a.c. A4: 2000	a.c. A1: 1998

```
public class TesteAmigoEPessoa {

    public static void main(String[] args) {

        Amigo a1 = new Amigo
                (new Pessoa("Lurdes", Pessoa.FEM), 1984,
                16);
        a1.setContacto("789123456");
        Pessoa p1 = new Pessoa("Paulo", Pessoa.MAS);

        a1.casa(p1);
        System.out.println(a1);

        Pessoa p2;
        p2 = a1;
        System.out.println("\n" + p2 + "\n");

    }
}
```

```
DADOS@AMIGO
NOME: Lurdes; SEXO: Feminino; NAC.: Portuguesa
CONHECI EM 1984; É AMIGO NOTA 16; CONTACTO: 275123456; ANIVERSÁRIO: desconhecida
PARCEIRO: Paulo
```

```
DADOS@AMIGO
NOME: Lurdes; SEXO: Feminino; NAC.: Portuguesa
CONHECI EM 1984; É AMIGO NOTA 16; CONTACTO: 275123456; ANIVERSÁRIO: desconhecida
PARCEIRO: Paulo
```

```
public class TestePessoaEAmigo{  
  
    public static void main(String[] args) {  
  
        Amigo a1 = new Amigo  
            (new Pessoa("Lurdes", Pessoa.FEM), 1984, 16);  
        a1.setContacto("789123456");  
        Pessoa p1 = new Pessoa("Paulo", Pessoa.MAS);  
  
        a1.casa(p1);  
        System.out.println(a1);  
  
        Amigo a2;  
        a2 = p1;//acham bem?  
  
        //ERRO DE COMPILAÇÃO: incompatible types found  
  
        a2 = (Amigo) p1;//melhor assim?  
  
        //ERRO DE EXECUÇÃO: java.lang.ClassCastException  
  
    }  
}
```