Herança (exemplo):
Simplificando, podemos afirmar que, uma pessoa é alguém de quem sabemos o nome, o sexo e a nacionalidade.
Programe a classe Pessoa com os construtores, seletores e modificadores, 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 = "Portuguesa";
  }

 public Pessoa(String nome, String sexo, String nacionalidade) {
//uso explícito do construtor anterior
   this(nome, sexo);
   this.nacionalidade = 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  …

  }
```

Herança (exemplo):

Programe a classe Amigo. 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".

Exercício adaptado de "Programação com classes em C++", Pedro Guerreiro, FCA, 2000

```
import java.time.LocalDate;

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 LocalDate dataNasc;
  private String contacto;
  private int anoConhec;
  private int nivelAmiz;
  private Pessoa parceiro;
```

Herança (exemplo):

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

   LocalDate hoje = LocalDate.now();
   anoConhec = hoje.getYear();

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

Herança (exemplo):

```
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?
 }
```

Herança (exemplo):

```
//Construtor de cópia
  public Amigo(Amigo copia) {

     this(copia, copia.anoConhec, copia.nivelAmiz,  copia.parceiro);
     contacto = copia.contacto;

     dataNasc = copia.getDataNasc() ;     // objeto imutável;
  }

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

Herança (exemplo):

```
 public LocalDate 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 ( LocalDate.now().getYear() - anoConhec );
  }
```

Herança (exemplo):

```
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;
  }
```

Herança (exemplo):

```
public void setDataNasc(LocalDate data){
  this.dataNasc = data;  //apesar de ser um objeto, não clonamos
 }

public int idade(){
  //PRE: dataNascConhecida()
  LocalDate dataHoje = LocalDate.now();
  int idade =  dataHoje.getYear() - dataNasc.getYear();
  if (dataHoje.getDayOfYear() <  dataNasc.getDayOfYear())
       return idade-1;
  return idade;
 }
```

Herança (exemplo):

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

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); }
```

Herança (exemplo):

```
//método auxiliar
private String getDataFormatada(){
    return ( dataNascConhecida() ?
            dataNasc.getYear() + "/"
            +dataNasc.getMonthValue()+ "/"
            + dataNasc.getDayOfMonth() :
           "desconhecida" );
 }
```

Herança (exemplo):

```
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
  }
}// classe Amigo
```

```
public class TesteAmigo {
  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(  LocalDate.of (1975, 12, 20) );
    System.out.println (
                "\n" + a1 + "\nIDADE: " + a1.idade() +"\n");
```

Output:

DADOS@AMIGO
Pessoa{nome=Maria Só Amadeu, sexo=Feminino, nacionalidade=Portuguesa}
CONHECI EM 1998; É AMIGO NOTA 12; CONTACTO: maria@vaicom.asoutras.pt; ANIVERSÁRIO: desconhecida
PARCEIRO: C. Amadeu

DADOS@AMIGO
Pessoa{nome=Maria Só, sexo=Feminino, nacionalidade=Portuguesa}
CONHECI EM 1998; É AMIGO NOTA 17; CONTACTO: maria@vaicom.asoutras.pt; ANIVERSÁRIO: 1975/12/20
IDADE: 43

Herança (exemplo):

```
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()

//Output:
DADOS@AMIGO
Pessoa{nome=José Silva Só, sexo=Masculino,
nacionalidade=Portuguesa}
CONHECI EM 1984;     É AMIGO NOTA 16;      CONTACTO:
275123456;      ANIVERSÁRIO: desconhecida
PARCEIRO: Maria Só
```

```
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());

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

Output:

TESTE DE CÓPIA DE REFERÊNCIAS
a.c. A2: 1984 a.c. A4: 1900 a.c. A2: 1900

```
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());
  }
}
```

Output:

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;                              //Amigo é subclasse de Pessoa
    System.out.println("\n" + p2 + "\n");
   }
}
```

Output:

DADOS@AMIGO
Pessoa{nome=Lurdes, sexo=Feminino, nacionalidade=Portuguesa}
CONHECI EM 1984; É AMIGO NOTA 16; CONTACTO: 789123456; ANIVERSÁRIO: desconhecida
PARCEIRO: Paulo

DADOS@AMIGO
Pessoa{nome=Lurdes, sexo=Feminino, nacionalidade=Portuguesa}
CONHECI EM 1984; É AMIGO NOTA 16; CONTACTO: 789123456; 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?**

Output:

//ERRO DE EXECUÇÃO: java.lang.ClassCastException