JAVA.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 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;

  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){

  …

  }

}
```

Programe a classe Amigo[1]. 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".

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

```
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;
  }

  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;
}

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
  }

}
```