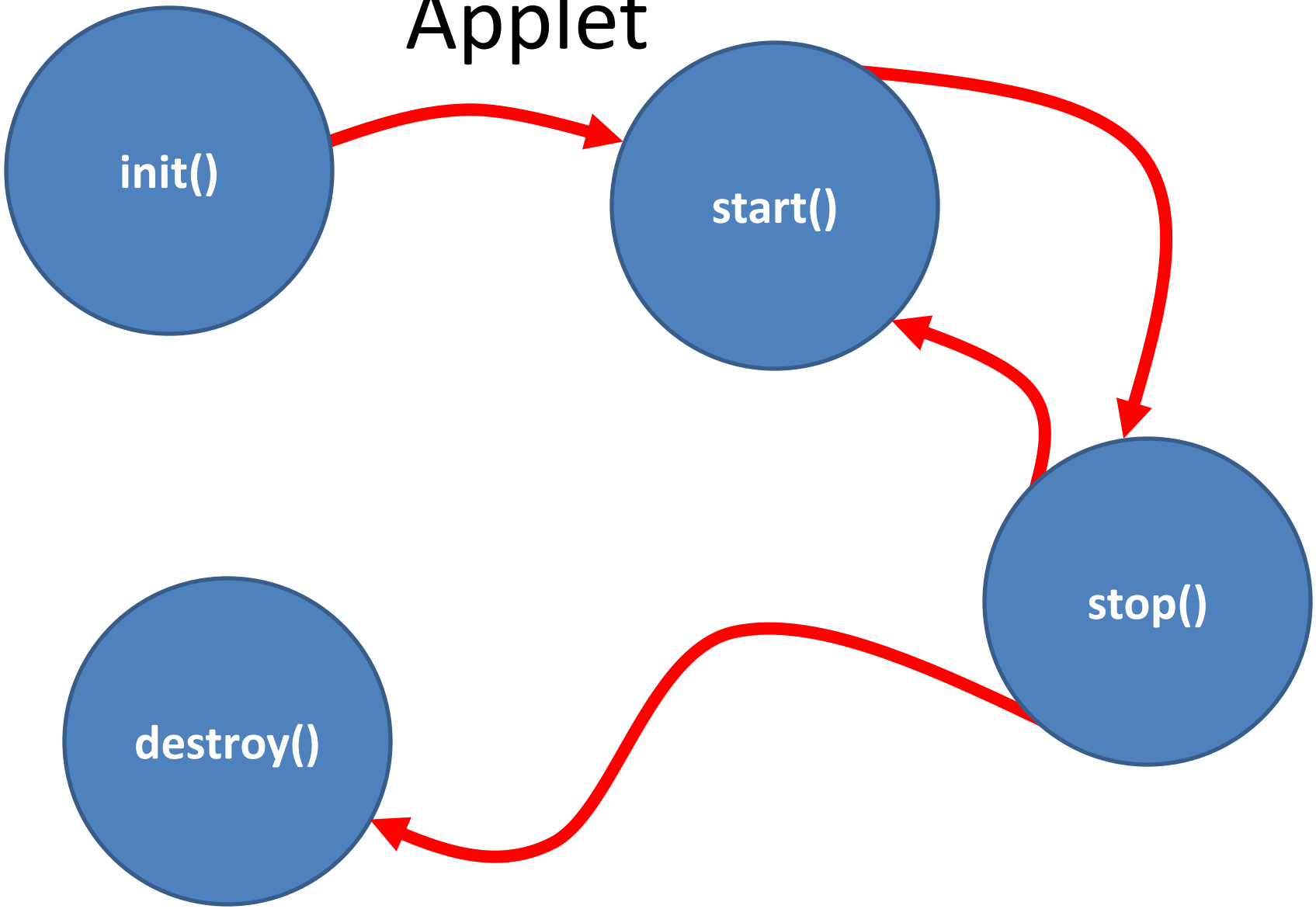
# Java
# Applet

Vitor Vaz da Silva

# Applet

- Programa em Java que pode ser descarregado e executado através da Internet num browser.
- Executado a partir do html
- Terá de haver autorização para o applet funcionar no computador
- Terá de estar instalada a máquina virtual
- O programa executa no computador cliente

# Applet



# init()

- Para inicializações
- Minimizar este método para facilitar o download

# start()

- Começa a execução da applet
- Caso haja muito a executar é preferível lançar um processo
- Chamado quando depois de minimizado é activado.

# stop()

- Deve suspender a execução quando o applet não está em uso para garantir que os recursos são minimizados.
- Quando o applet não está a ser visto pode não fazer sentido estar a mostrá-lo.
- Chamado quando é minimizado!

# destroy()

- Para terminar completamente o applet.

```
package aula;

public class Applet_ex_vazio extends javax.swing.JApplet {

public void inicio(){
       //colocar aqui o nosso programa principal
}

public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                    inicio();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
```

# Thread

- Um objecto que é um programa.
- Permite executar ao mesmo tempo vários métodos no mesmo programa.
- Concorrência
- Sincronismo

# Thread

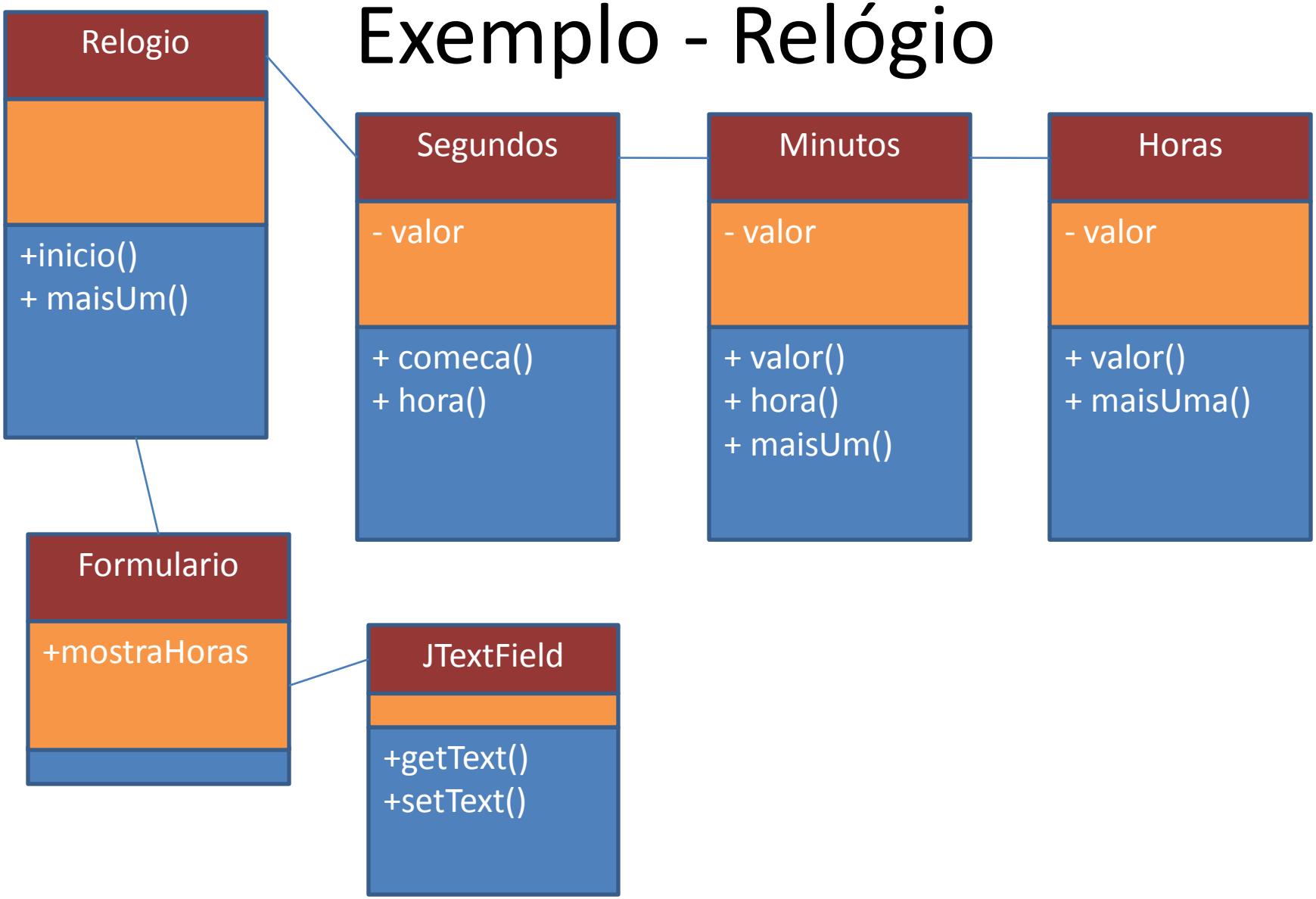
```
public class Segundos extends Thread{
    private boolean continua=true;

    public void comeca(){
        this.start();
    }

    public void termina(){
        continua = false;
    }

    @Override
    public void run(){
        while(continua){
            // colocar aqui o código do processo
        }
    }
}
```

# Exemplo - Relógio



```
public class Segundos extends Thread{
    private Minutos minutos = new Minutos();
    private int valor=0;
    private boolean continua=true;
    private Relogio relogio;

    public String hora(){
        String str = "" + minutos.hora() + ":" + minutos.valor() + ":" + valor;
        return(str);
    }

    public void comeca(Relogio in){
        this.start();
        inicio=in;
    }

    @Override
    public void run(){
        while(continua){
            try {
                this.sleep(1000);
            } catch (InterruptedException ex) {
                System.out.println("Erro em Segundos.run");
            }
            if(++valor >= 60){
                valor=0;
                minutos.maisUm();
            }
            relogio.maisUm();
        }
    }
}
```

```
public class Relogio extends javax.swing.JApplet {
Segundos segundos;

    public void maisUm(){
        String str = segundos.hora();
        mostraHoras.setText(str);
    }

    public void inicio(){
        segundos = new Segundos();
        segundos.comeca(this);
    }
```

```
class Minutos {
    private Horas horas = new Horas();
    private int valor=0;

    public int valor(){
        return(valor);
    }

    public int hora(){
        return(horas.valor());
    }

    void maisUm() {
        if(++valor>=60){
            valor=0;
            horas.maisUma();
        }
    }
```

```
class Horas {
    private int valor=0;

    public int valor(){
        return(valor);
    }

    public void maisUma() {
        if(++valor>=24) valor=0;
    }
}
```

# Desenhos

- paint() – chamado cada vez que é necessário refrescar o ecrã (a visualização da applet)
- repaint() – forçar o refrescar do ecrã; apaga tudo primeiro e chama o método paint()
- this.getGraphics() – devolve o contexto gráfico

# Desenhos

```
public void desenhaTudo(){
  // colocar aqui o refrescar dos nossos desenhos()
}

 public void desenhaUmaVez(){
  // colocar aqui o que não muda nos desenhos
}

@Override
public void paint(Graphics g){
  super.paint(g);
  desenhaUmaVez();
  desenhaTudo();
}

public Graphics2D tela(){
    return(Graphics2D) (this.getGraphics());
}
```

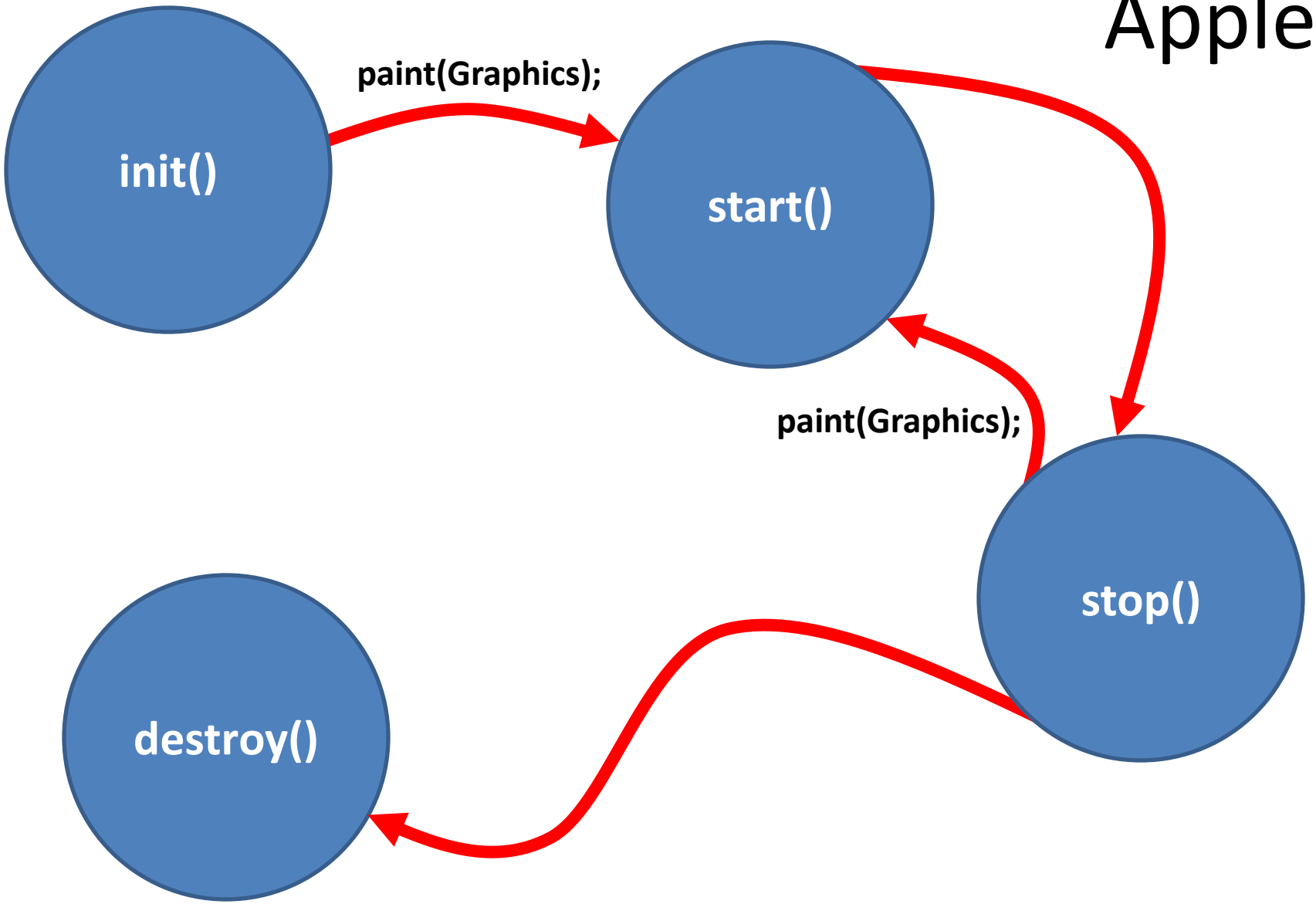
*Acrescentar este código para desenhar componentes*

# Desenhos

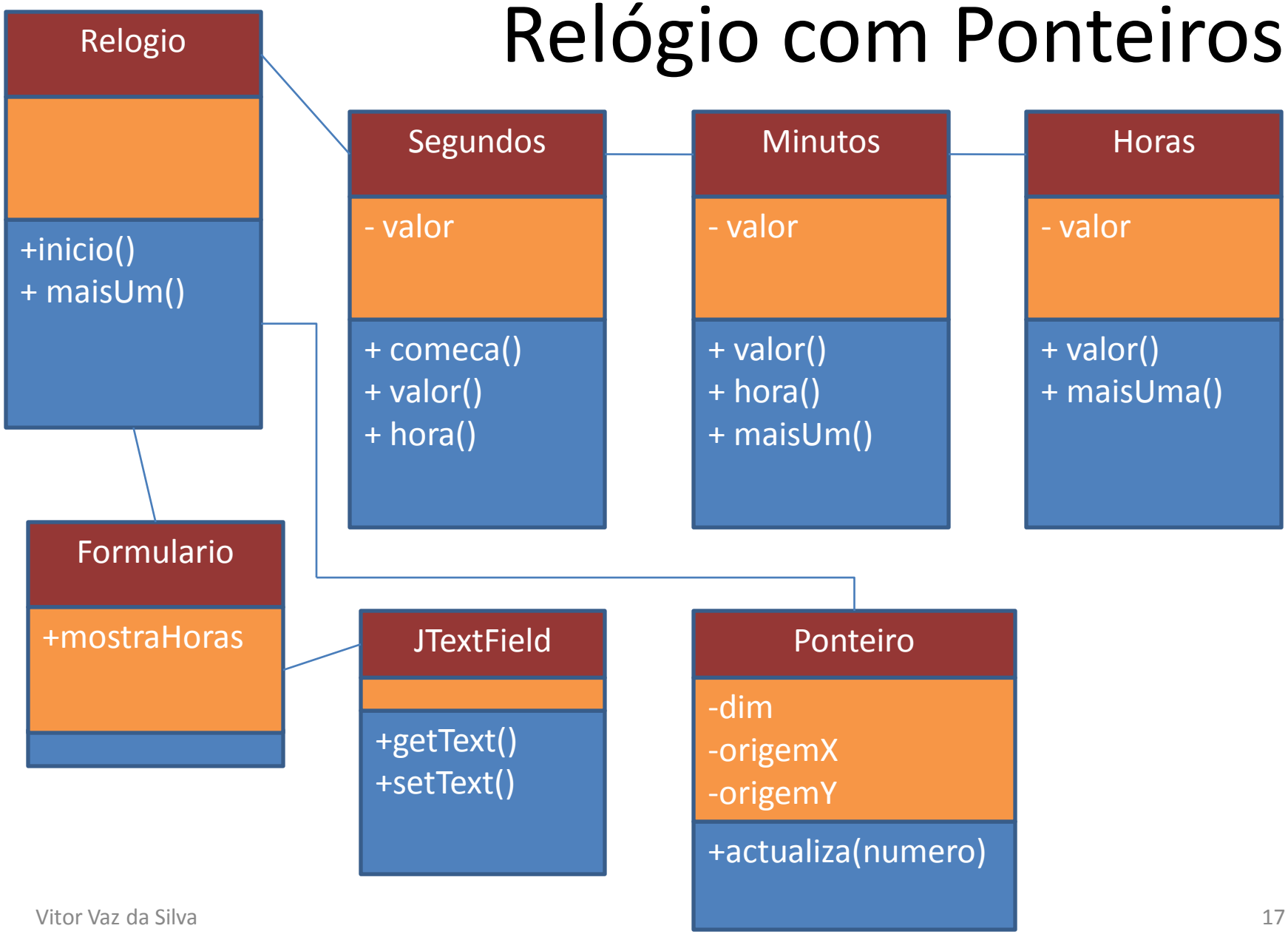
Com o código anterior:

- repaint() – chama o desenhaTudo();
- Para actualizar o applet não precisamos de estar sempre  chamar o repaint() para actualizar. Basta chamar o desenhaTudo() se tivermos o cuidado de apagar apenas o essencial!
- O desenhaUmaVez() só é chamado quando a janela é mexida (redimensionada ou danificada). Desenha o que permanece sempre inalterado nos nossos desenhos.
- O método tela() devolve o contexto gráfico onde pretendemos desenhar.

# Applet



# Relógio com Ponteiros



# Html

**<APPLET**

CODEBASE = *codebaseURL*
ARCHIVE = *archiveList*
**CODE = *appletFile*** ...or... **OBJECT = *serializedApplet***
ALT = *alternateText* NAME = *appletInstanceName*
**WIDTH = *pixels***
**HEIGHT = *pixels***
ALIGN = *alignment*
VSPACE = *pixels*
HSPACE = *pixels*

**>**

<PARAM NAME = *appletAttribute1* VALUE = *value*>
<PARAM NAME = *appletAttribute2* VALUE = *value*>
. . .
*alternateHTML*

**</APPLET>**

**Obrigatório**
Opcional

# Html

CODEBASE = *codebaseURL*

*Directoria onde está o código da applet. Se não estiver presente utiliza-se o url do ficheiro.*

ARCHIVE = *archiveList*

*Separados por , nome de arquivos a carregar relativamente à directoria CODEBASE*

**CODE = *appletFile***

*Nome do ficheiro que contém a classe a executar. O nome pode ser nomepackage.nomeclass*.class
*O nome é relativo ao CODEBASE e não pode ser absoluto*

# Html

**OBJECT = *serializedApplet***

*Descerializa o applet e o método init() não é chamado. O método start() é invocado.*

*CODE e OBJECT são exclusivos*

ALT = *alternateText*

*Texto escrito caso o java não possa ser executado*

NAME = *appletInstanceName*

*Identificação da applet para ser usado no html*

# Html

**WIDTH = *pixels* HEIGHT = *pixels***

*Dimensão do applet. Pode ser um número ou em %*

ALIGN = *alignment*

*Idêntico ao alinhar de uma imgem, pode ser: left, right, top, texttop, middle, absmiddle, baseline, bottom, absbttom.*

VSPACE = *pixels* HSPACE = *pixels*

*Distância da applet à janela*

# Html

<PARAM NAME = *attribute* VALUE = value>

*A applet utiliza o método getParameter() para aceder ao conteúdo do atributo. Deste modo a applet pode ler valores de um formulário em html.*

# Browser

**Loading the Applet**

- Ao ser carregada pode ver-se o texto "initializing... starting...". Após o carregamento é criada um instância da controlling class (subclass). A applet inicializa-se e começa a correr.

**Leaving and Returning to the Applet's Page**

- Caso seja visitada outra página na mesma janela do browser, a applet é destruída. Nem o contexto é guardado, por isso ao voltar à página do applet esta é carregada novamente e uma nova instncia é criada como se da primeira vez se tratasse.

**Reloading the Applet**

- O refresh ou recarregar da página pára a applet e destroi-a procedendo de seguida à criação de uma nova instância.

**Quitting the Browser**

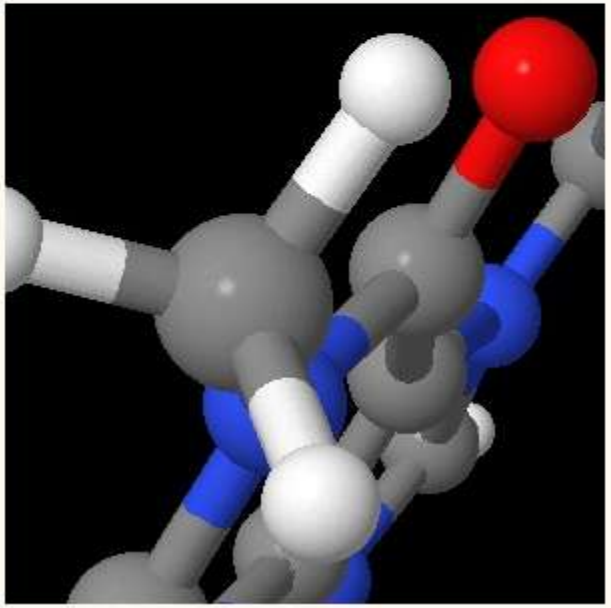
- Ao fechar o browser, a applet é parada e destruída.

# JMOL

Jmol

## Jmol: an open-source Java viewer for chemical structures in 3D

with features for chemicals, crystals, materials and biomolecules



Jmol is an interactive web browser applet.

This is a still image, but you can get an animated display of Jmol abilities by clicking here.

(The applet may take some seconds to load. Please, wait and do not reload the page in the meantime.)

- Java Applets para Física, Matemática, Química, e outras matérias

  http://www.walter-fendt.de/ph14pt/

  http://www.falstad.com/mathphysics.html

  http://www.prof2000.pt/users/jmtcor/applets.htm

  http://web.uconn.edu/~cdavid/

# Referências


- http://download.oracle.com/javase/tutorial/deployment/applet/index.html
- http://www.duckware.com/applets/reference.html#06202000
- http://download.oracle.com/javase/1.4.2/docs/guide/plugin/developer_guide/using_tags.html
- http://tektonia.com