Applet

Applet = un program java care se execută de un browser într-o pagina web

Un applet poate fi compus din mai multe clase, din care una este principală şi extinde **clasa Applet** (sau subclasa **JApplet**). Această clasă se precizează în pagina web astfel:

```
<applet code=nume_clasa.class [lista_atribute] >
  [<param name="nume" value="valoare>]...
</applet>
```

In lista de atribute din tag-ul applet trebuie să apară cel puțin atributele **width** şi **height**, care precizează dimensiunile appletului. Mai pot apare **atributele (parametrii)**:

- **archive=listă_fisiere.jar** - o lista de **arhive jar** unde se află clase ale appletului;
- **codebase=director** - precizează directorul unde se caută appletul;
- **alt=text** - textul care se afişează dacă browserul nu poate executa appletul;

Exemplu:

```
<applet id="applet" code="fractali.class" width="1200" height="700">
   alt="Browserul nu poate executa &lt;APPLET&gt;-ul precizat in fisier"
</applet>
```

Un **applet** descrie o suprafață de afişare (ca un **Panel** sau **JPanel**). In clasa principală (precizată şi în applet) se implementează metodele:

- **init**() - care se execută o singură dată, la **inițializarea appletului**;
- **start**() - care se execută la **vizualizarea appletului** (inițial după init, sau după o oprire temporară şi se o revenire prin activarea appletului)
- **stop**() - care se execută la **minimizarea ferestrei** care conține appletul (minimizare efectuată prin browser)
- **destroy** - care se execută la **închiderea browserului**

Aceste metode se execută **automat** de browser aşa cum este precizat mai sus, deci **nu sunt necesare apeluri explicite**.

In **applet <u>se poate</u>** implementa şi metoda **paint**(Graphics g) - care se execută la vizualizare (pagina web a fost acoperită şi se revine la ea, pagina web îşi schimbă dimensiunile), sau la cerere cu metoda **repaint()** (de exemplu, unele componente îşi schimbă conținutul şi se cere redesenarea lor).

In applet se pot prelua valorile parametrilor precizați în pagina web prin metoda **getParameter("numeparametru")**. Valoarea parametrului precizat se dă ca şir de caractere. Dacă parametrul precizat nu există, atunci se furnizează valoarea **null**.

Intr-un applet se pot reda sunete cu metoda **play(url)** pentru un obiect **AdioClip** (acest obiect se poate crea cu **getAudioClip**). Se mai pot folosi metodele: **loop()** şi **stop()**.

Clasa **JOGLAppletLauncher** este o subclasă a clasei **Applet** şi permite lucrul cu **Jogl**.

Un applet nu poate să:

- Citească sau să scrie fişiere pe calculatorul pe care este executat (la client).
- Lanseze în execuție programe pe calculatorul client.
- Citească informații de la sistemul de operare al clientului.

Thread

Un **proces** este un program aflat în execuție. In general (pe sistemele multitasking) pot să fie mai multe procese (pot exista mai multe aplicații în execuție).

Un şir de instrucțiuni care se execută în cadrul unui proces formează un **thread** (sau fir de execuție). In cadrul unui proces pot exista mai multe thread-uri.

Scopul folosirii unui thread în cadrul acestui curs este de a face un "demo" pentru o aplicație (care se execută ca un applet). Thread-ul va executa un şir de instrucțiuni, iar la anumite momente se va întrerupe execuția acestui şir de instrucțiuni pentru un anumit interval de timp.

In Java se poate crea un thread:

- ca o extensie a **clasei Thread**,
- prin implementarea **interfeței Runnable** (clasa Thread implementează această interfață).

Indiferent de modalitatea folosită este necesară implementarea metodei ***public void run()***, care va conține **instructiunile care se vor executa în cadrul threadului**. Un fir de execuție creat se va lansa în execuție prin metoda ***start()*** (metoda definită în clasa Thread).

Dacă se doreşte crearea unei singure clase care extinde o **clasă Applet** şi care va defini şi un **Theread**, ar trebui să fie ceva de felul următor:

```
class Problema extends Applet, Thread ...
```

care **este eronat**, deoarece **nu avem moştenire multiplă**. Rămâne varianta ca în clasa care se va construi să se implementeze şi interfața Runnable (se vor pierde unele facilități din clasa Thread). Clasa ce se va construi va fi:

```
class Problema extends Applet implements Runnable {
  public void init() {
  }
  public void start() {
  }
  public void stop() {
  }
  public void destroy() {
  }
  public void paint(Graphics g) {  //optionala
  }
  public void run() {
  }
}
```

Deoarece pentru exemplul nostru vom folosi un thread cu facilitati restrânse (numai local, pentru un demo), vom putea alege o variantă de forma următoare: în metoda care implementeaza demo-ul se va **crea un obiect thread** şi se implementează metoda **run**.

Definirea unui obiect thread se face prin:

```
Thread fir = null ;
```

Crearea thread-ul se face astfel:

```
fir = new Thread (this);
```

```
//cu this => se va crea thread-ul cu codul din clasa curenta
```

şi se va lansa în execuție:

```
fir.start ();
//se vor executa instrucțiunile precizate în metoda run().
```

Pentru a preciza o pauză în execuția firului de execuție se poate folosi metoda **sleep(nr_milisecunde)**, definită în clasa **Thread**. Deoarece pot apare excepții (de tip InterruptedException), apelul se va include într-o instrucțiune try-catch:

```
try {
Thread.sleep (3000) ;
} catch (InterruptedException e) {}
```

Firul de execuție se va întrerupe din execuție, iar după expirarea timpului se va relua. Un fir de execuție se termină la finalul execuției metodei **run()**.

Metoda **destroy()** va distruge un **thread**.