Applet-uri Java, noțiuni de grafică

Java applets, graphics concepts

Obiective:

- implementarea applet-urilor; metode specifice;
- applet-aplicații
- înțelegerea și aplicarea noțiunilor elementare de grafică

Objectives:

- implementing applets; specific methods;
- understanding and applying the basic graphics concepts;

Implementarea applet-urilor

Pentru că o clasă Java să aibă comportament de *applet* şi să fie interpretată ca atare, trebuie să fie derivată din clasa de bază *java.applet.Applet.*

Metode principale:

Metoda constructor

- *Applet()*

=> crează o instanţă a clasei Applet

Metode membre principale

- *void destroy()*

=> apelată de browser-ul de web sau de utilitarul *appletviewer.exe* pentru a informa applet-ul în cauză că urmează să fie distrus şi toate resursele care i-au fost alocate urmează să fie eliberate

- *AccessibleContext getAccessibleContext()*

=> returnează obiectul de tip *AccessibleContext* asociat cu applet-ul curent

- *AppletContext getAppletContext()*

=> determina contextul applet-ului curent, obiect care va permite interacţionarea cu mediul în care rulează acest applet

- *String getAppletInfo()*

=> returnează informaţia textuala legată de applet-ul curent

- *AudioClip getAudioClip(URL url)*

=> returnează clip-ul audio identificat prin URL-ul primit ca parametru

- *AudioClip getAudioClip(URL url, String name)*
- => returnează clip-ul audio identificat prin URL-ul şi prin numele acestuia, primite ca parametri
- *URL getCodeBase()*

=> returnează URL-ul de unde a fost încărcat codul applet-ului

- *URL getDocumentBase()*

=> returnează URL-ul documentului web în care a fost înglobat applet-ul curent

- *Image getImage(URL url)*

=> returnează un obiect *Image* de la locaţia URL primită ca parametru

- *Image getImage(URL url, String name)*

=> returnează un obiect *Image* de la locaţia URL primită ca parametru şi cu numele reprezentat prin variabilă de tip *String* primită ca parametru

- *Locale getLocale()*

=> returnează un obiect *Locale* (conţine informaţii geografice, culturale, etc. despre o anumită regiune) dacă această

informaţie este disponibilă

- *String getParameter(String name)*

=> returnează valoarea unui parametru care este primit din fişierul HTML ataşat şi identificat prin numele primit ca parametru

- *String[][] getParameterInfo()*

=> returnează informaţiile despre parametrii care sunt "înţeleşi" de applet-ul curent

- *void init()*

=> metodă apelată de browser-ul de web sau de utilitarul *appletviewer.exe* pentru a informa applet-ul că a fost încărcat în sistem

- *boolean isActive()*

=> determină dacă applet-ul curent este activ sau nu

- *static AudioClip newAudioClip(URL url)*

=> obţine un clip audio de la URL-ul primit ca parametru

- *void play(URL url)*

=> redă clip-ul încărcat de la URL-ul primit ca parametru

- *void play(URL url, String name)*
- => redă clip-ul încărcat de la URL-ul primit ca parametru, identificat prin numele primit ca parametru
- *void resize(Dimension d)*

=> redimensionează applet-ul potrivit dimensiunilor stocate în obiectul de tip *Dimension* primit ca parametru

- *void resize(int width, int height)*

=> redimensionează applet-ul potrivit dimensiunilor primite ca parametri

- *void setStub(AppletStub stub)*

=> setează componenta *stub* a applet-ului curent

- *void showStatus(String msg)*

=> afişează parametrul primit în zona de status a ferestrei de rulare a applet-ului

- *void start()*

=> metodă apelată de browser-ul de web sau de utilitarul *appletviewer.exe* pentru a informa applet-ul că trebuie să îşi înceapă execuţia

- *void stop()*

=> metodă apelată de browser-ul de web sau de utilitarul *appletviewer.exe* pentru a informa applet-ul că trebuie să îşi oprească execuţia

Ciclul de viaţă al unui applet

Întreg ciclul de viaţă al unui applet este controlat şi implementat cu ajutorul metodelor:

- *public void init()*
- *public void start()*
- *public void stop()*
- *public void destroy()*

Aceste metode sunt apelate în ordinea menţionată mai sus. În clasa *java.applet.Applet*, acestea au implementare vidă şi este la latitudinea programatorului să le suprascrie pentru a le conferi funcţionalităţi specifice fiecărei aplicaţii în parte. Dacă nu sunt tratate în nici un fel, metodele mai sus amintite nu fac nimic.

Pentru a controla aspectul grafic al unui applet, se poate face uz de suprascrierea unei metode care este moştenită din super-clasa *java.awt.Container*, şi anume

public void paint(Graphics g)

Aceasta metodă este apelată ori de câte ori este necesară desenarea/redesenarea contextului grafic al applet-ului. Prin conferirea unei implementări specifice poate fi controlat modul de prezentare vizuală a applet-urilor.

Rularea unui applet

Paşii care trebuie urmaţi pentru rularea unui applet:

1. Scrierea codului sursa a aplicaţiei **.java*. Este recomandat ca clasa principală (cea care e derivată din *java.applet.Applet*) să fie marcată cu specificatorul de vizibilitate *public*.
2. Compilarea codului sursă a aplicaţiei pentru a obţine fişierul (fişierele) bytecode **.class* aferente

   C:> javac HelloWorld.java

3. Înglobarea codului applet-ului într-un fişier **.html* folosind tag-ul *<APPLET>...</APPLET>*

```
<HTML>
<BODY>

<APPLET code="HelloWorld" width="300" height="300">
</APPLET>

</BODY>
</HTML>
```

Parametrul *code* identifica numele fişierului **.class* rezultat din compilarea codului sursa al applet-ului. Ceilalţi doi parametri specifică dimensiunea (lăţime şi înălţime) zonei care va fi alocată pentru afişarea applet-ului.

Dacă liniile HTML de mai sus au fost salvate în fişierul *applet_launcher.html*, pornirea aplicaţiei se poate face în următoarele moduri:

a. Se încarcă fişierul **.html* creat într-un browser de web (apar probleme la rulare datorate restricțiilor introduse în ultima perioadă)
b. Se foloseşte utilitarul *appletviewer.exe*:

   C:> appletviewer applet_launcher.html

c. Din mediul de programare Eclipse: Run -> Run as applet

appletviewer.exe pune la dispoziţie un context software care simulează perfect rularea applet-ului în cazul real al încărcării într-un browser de web.

Transmiterea parametrilor către applet din codul HTML

Applet-urile pot comunica unidirecţional cu paginile web care le încarcă, şi anume pot prelua din codul HTML anumiţi parametri de intrare.

Specificarea parametrilor în codul **.html* se face prin intermediul tag-ului *<PARAM>*, inclus în tag-ul *<APPLET>...</APPLET>*:

<PARAM NAME="nume_parametru" VALUE="valoare_parametru">

Fiecare parametru are un identificator reprezentat prin numele acestuia şi o valoare. Indiferent de natura datelor care trebuie să fie transmise către applet, acestea vor fi preluate de codul Java în format *String*. Este sarcina programatorului să le transforme ulterior în tipul de date dorit.

Pentru rulare, se poate apela la variantele menționate mai sus sau se pot configura parametrii adiționali astfel: din mediul

de programare Eclipse: Run -> Run Configurations -> Parameters: se introduc perechile [nume, valoare] care urmează a fi prelucrate în codul Java.

Appletcation-uri Java

Un *appletcation* (applet-aplicație) este acea aplicaţie Java care are atât atributele unui applet cât şi atributele unei aplicaţii independente (metoda *public static void main(String... args)*) fiind în general orientat spre aplicaţii grafice. Efectul acestui lucru este o mai mare mobilitate în ceea ce priveşte rularea lor, ele putând fi pornite atât ca aplicaţii de sine stătătoare, cât şi că applet-uri.

Comunicarea între applet-uri

Java pune la dispoziţie un mecanism prin care două sau mai multe applet-uri care sunt încărcate în aceeaşi pagină HTML pot comunica între ele. Această tehnologie nu poate fi extinsă pentru a realiza mecanisme complexe de comunicare de date intre applet-uri încărcate în pagini diferite sau în browser-e de Internet diferite. Este o tehnologie locală care serveşte la unificarea contextului de rulare în cadrul aceleiaşi locaţii de web.

Din moment ce applet-urile încărcate în aceeaşi pagină de web rulează în acelaşi context Java (sunt încărcate de aceeaşi JVM) este oarecum normal ca ele să poată să se "vadă" între ele şi să îşi apeleze metodele reciproc.
Singura problemă care trebuie rezolvată de către un applet este obţinerea referinţelor către celelalte aplicaţii similare care rezidă în aceeaşi locaţie web. Clasa *AppletContext* pune la dispoziţia programatorului mecanismele prin care se pot crea referinţe la alte applet-uri pe baza numelui acestora. Există de asemenea posibilitatea de a crea o listă cu toate applet-urile detectate în contexul web curent.

Grafica în Java

Clasa *Graphics* https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics.html
Clasa Graphics2D https://docs.oracle.com/javase/8/docs/api/java/awt/Graphics2D.html

Se folosesc instanțe ale claselor de mai sus pentru a avea acces la contextul grafic atașat unui container (de ex. *Applet*) sau a unor componente specifice (de ex. *Canvas*).

Lucru individual

1. Creați un applet ce preia ca și parametri HTML numele vostru și grupa din care faceți parte. Applet-ul va afișa apoi textul: ”Salut {nume}! Grupa ta este {grupa}." Textul va fi de culoare albastră și centrat atât pe orizontală, cât și pe verticală, ținând cont de dimensiunea implicită a unui applet.

==========

2. Scrieți un appletcation care dacă este rulat ca și aplicație va prelua de la tastatură un mesaj și-l va afișa în consolă, iar dacă este rulat ca și applet va desena un cerc de culoare roșie tangent la dimensiunea cea mai mică (înălțime, lățime applet) a appletului.

==========

3. Creați un applet ce afișează o imagine stocată local, cu dimensiunea de 100x100 și sub care este afișat și numele fișierului din care provine.

==========

4. Creați un applet ce afișează o mină de televiziune. Mira va conține minim 10 nivele de gri și culorile de bază: roșu, verde, albastru, galben, cyan și magenta. Mira va acoperi întreaga dimensiune a applet-ului, dimensiune definită de utilizator ca și parametri HTML.

==========

5. Scrieți un applet Java care desenează un cerc colorat în toate culorile posibile. Se va începe cu culoarea roșie, tranzițiile dintre culori trebuie să fie line, iar cercul este redesenat pe măsură ce culorile sunt generate.

==========

6. Creați un applet ce desenează o mașină schematică și afișează textul ”Vruuummm!”. Culoarea liniilor ce delimitează mașina, precum și culorile roților și a textului vor fi preluate ca și parametri HTML individuali, dați sub forma unei secvențe de 3 întregi separați prin virgulă. De ex. Parametru=”culoare_masina”, Valoare=”123, 123, 123”.

* Folosind principiul de double buffering, creați o imagine de dimensiune 200x200 care să conțină mașina desenată anterior și care să se deplaseze spre dreapta cu câte 5px până ce iese din applet complet.

==========

7. Implementați un applet care umple containerul visual al aplicației ecranul cu cercuri a caror rază este generată aleator între 2 limite primite ca parametri din fișierul HTML. Cercurile sunt tangente exterior și colorate aleator.

==========

8. Scrieți o aplicație de tip applet care desenează toată traiectoria parcursă de un obiect care este “aruncat” pe orizontală cu o viteză predefinită. Traiectoria are ca limită marginea inferioară a ferestrei aplicației. Se iau în considerare și situațiile în care obiectul atinge în timpul căderii una din marginile laterale (ricoșare perfect elastică). Se afișează viteza verticală atinsă de obiect la finalul mișcării.

Individual work

1. Create an applet which uses 2 HTML parameters: your *name* and the *group* you are part of. The applet will then display the text: "Hello {name}! Your group is {group}." The text will be blue and centered both on the horizontal and vertical, given the default dimensions of an applet.

==========

2. Write an appletcation which when ran as an application will read a message from the keyboard and then display it. If it is ran as an applet, it will draw a red circle tangent to the smallest dimension of the applet (height, width).

==========

3. Create an applet which displays an image stored locally, with the dimension 100x100, and also, below the image, the name of the original file.

==========

4. Create an applet which displays a TV color test pattern. The pattern will contain at least 10 gray levels and the base colors: red, green, blue, yellow, cyan and magenta. The pattern will cover the entire dimension of the applet, which is given by the user as HTML parameters.

==========

5. Write a Java applet which draws a circle colored in all the possible colors. Start with the color red, make sure the color transitions are smooth, and the redraw the circle whenever a new color is generated.

==========

6. Write an applet which draws a schematic car and print the text "Vruuummm!". The color of the lines which make up the car, as well as the color of the wheels and the text will be given as individual HTML parameters by using a sequence of 3 integer values separated by comma. For example, parameter="car_color", value="123, 123, 123".

* Using the double buffering principle, create a 200x200 image which contains the previously drawn car and which moves to the right in 5px increments until the car is completely out of the applet.

==========

7. Implement an applet which fill the visual container of the application with circles of randomly generated radius. The radius is limited by 2 values given as HTML parameters. The circles are externally tangent and have random colors.

==========

8. Write an applet which draws the entire trajectory of an object thrown on the horizontal axis with a predefined speed. The trajectory is limited by the lower boundary of the applet window. Take into consideration the situations in which the object reaches one of the lateral boundaries before falling to the ground. Print out the vertical speed reached by the object before the movement stops.