Aplety
Javy

Definicja

Aplet (ang. applet – zdrobnienie od aplikacja)
Aplet jest specjalnym rodzajem programu napisanego w Javie, który jest uruchamiany w przeglądarce, wykorzystując wirtualną maszynę Javy albo w samodzielnej aplikacji AppletViewer służącej do testowania apletów Javy.

Aplety mogą być pisane zarówno w Javie jak i innych językach kompilowanych do kodu bajtowego - na przykład Jython (implementacja języka programowania Python napisana w języku Java.).

Aplety są wykonywane po stronie klienta, serwlety – po stronie serwera.

Klasa Applet

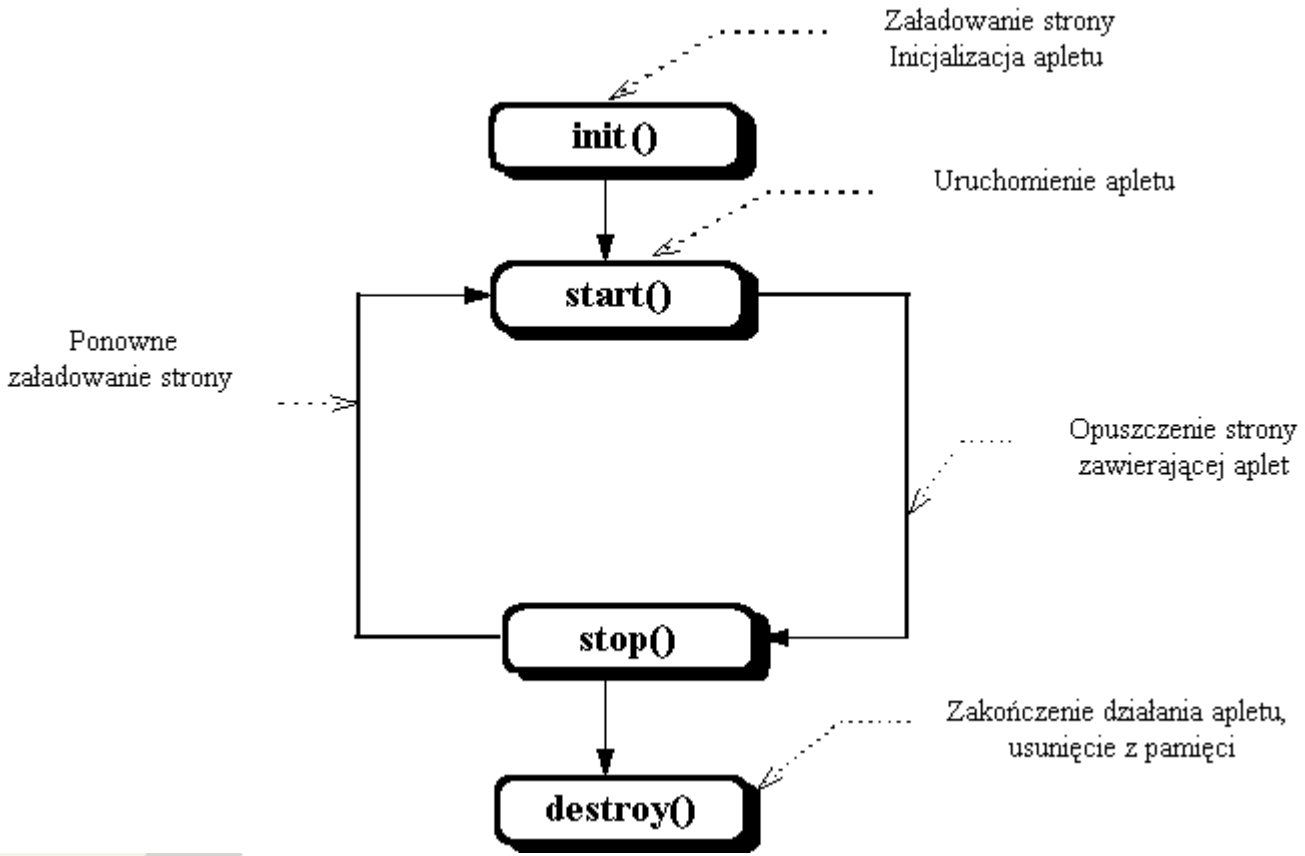
Aplet musi być dziedziczyć z klasy **java.applet.Applet** (lub jednego z jej wariantów), dostarczającej standardowy interface pomiędzy apletem a środowiskiem przeglądarki.

```
import java.applet.Applet;
import java.awt.Graphics;

public class Simple extends Applet {

    ....................................
    ....................................
}
```

```
import javax.swing.JApplet;

public class Simple extends JApplet {

    ....................................
    ....................................
}
```

Cykl życia apletu

Brak metody **main !!!**



Cykl życia apletu

- **Ładowanie apletu**
  - Tworzenie instancji klasy Applet (JApplet)
  - Nowoutworzony aplet inicjalizuje się
  - A następnie uruchamia się

- **Opuszczanie i powrót do strony z apletem**
  - Gdy użytkownik opuszcza stronę przeglądarka zatrzymuje i usuwa aplet. Stan apletu nie jest przechowywany. Po powrocie do strony z apletem, przeglądarka tworzy nowy obiekt apletu.

- **Przeładowywanie apletu**
  - Podczas odświeżania lub przeładowywania strony z apletem, obecna instancja apletu jest zatrzymywana i usuwana, a następnie tworzona jest nowa instancja apletu.

- **Zamknięcie przeglądarki**
  - Zanim przeglądarka zostanie całkowicie zamknieta, aplet musi mieć możliwość zatrzymania się i przeprowadzenia ostatecznego czyszczenia.

Prosty aplet informujący o swoim stanie

```
import java.applet.Applet;
import java.awt.Graphics;

public class Simple extends Applet {

    StringBuffer buffer;

    public void init() {
        buffer = new StringBuffer();  addItem("initializing... ");
    }
    public void start() {
        addItem("starting... ");
    }
    public void stop() {
        addItem("stopping... ");
    }
    public void destroy() {
        addItem("preparing for unloading...");
    }
    private void addItem(String newWord) {
        System.out.println(newWord);  buffer.append(newWord);  repaint();
    }
    public void paint(Graphics g) {
//Draw a Rectangle around the applet's display area.
        g.drawRect(0, 0, getWidth() - 1, getHeight() - 1);
//Draw the current string inside the rectangle.
        g.drawString(buffer.toString(), 5, 15);
    }
}
```

Środowisko wykonywania apletów

- Aplety uruchamiane są wewnątrz przeglądarki

  Aplety umieszczane są wewnątrz strony internetowej. Po ściągnięciu z serwera są uruchamiane w przeglądarce w tzw. piaskownicy (ang. sand box), aby ograniczyć dostęp do zasobów lokalnych.

- Java plug-in zainstalowany w przeglądarce kontroluje uruchamianie i działanie apletów.

- Przeglądarka musi również zawierać interpreter JavaScript

Uruchamianie apletów w JRE

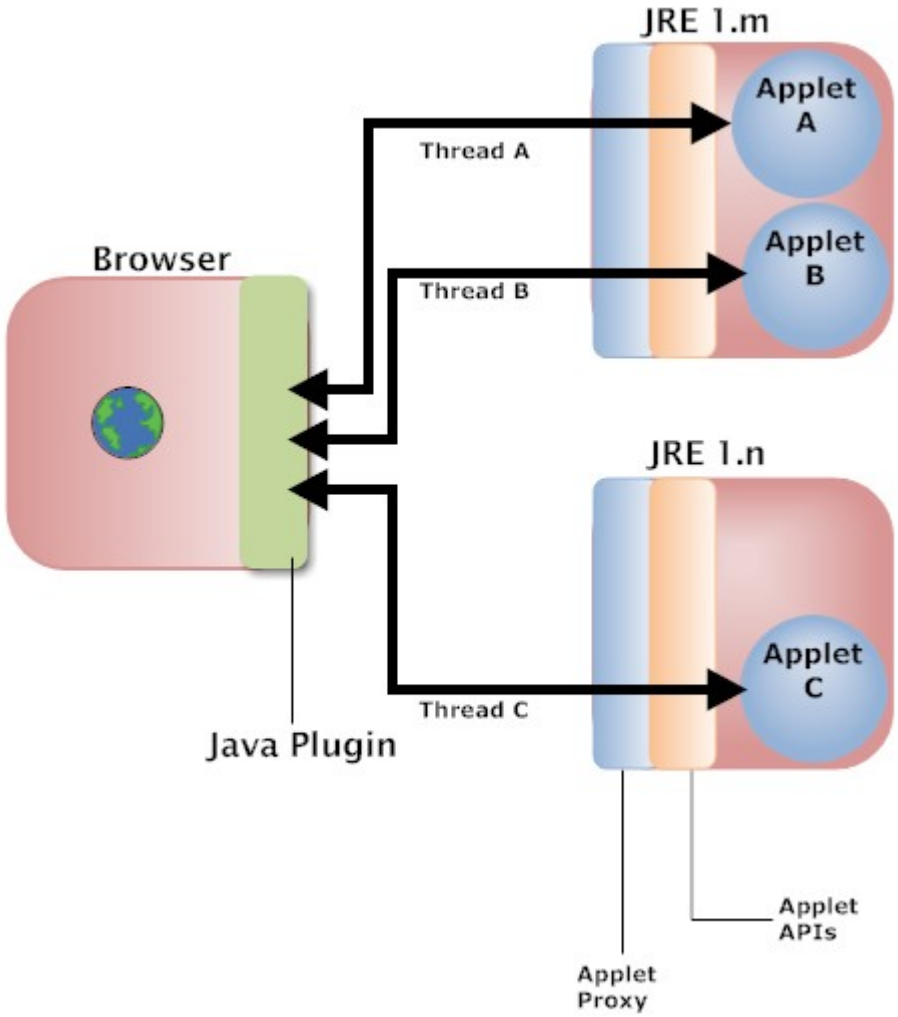
Java plug-in tworzy wątek dla każdgo apletu. Uruchamia on aplet w instancji Java Runtime Environment (JRE). Zwykle wszystkie aplety działają w ramach tej samej instancji JRE.

Java Plug-in otwiera nową instancję JRE w następujących przypadkach:

- Gdy aplet wymaga specyficznej wersji JRE.
- Gdy aplet określa swoje własne parametry startowe JRE, np. wielkość stosu. Nowy aplet używa instającego JRE, jeśli jego wymagania są podzbiorem parametrów JRE. W przeciwnym przypadku nowa instancja JRE jest uruchamiana.

Aplet będzie uruchamiany w istniejącym JRE, gdy spełnione są warunki:

- Wymagana przez aplet wersja JRE odpowiada insiejącemu JRE.
- Gdy parametry startowe JRE spełniają wymagania apletu.

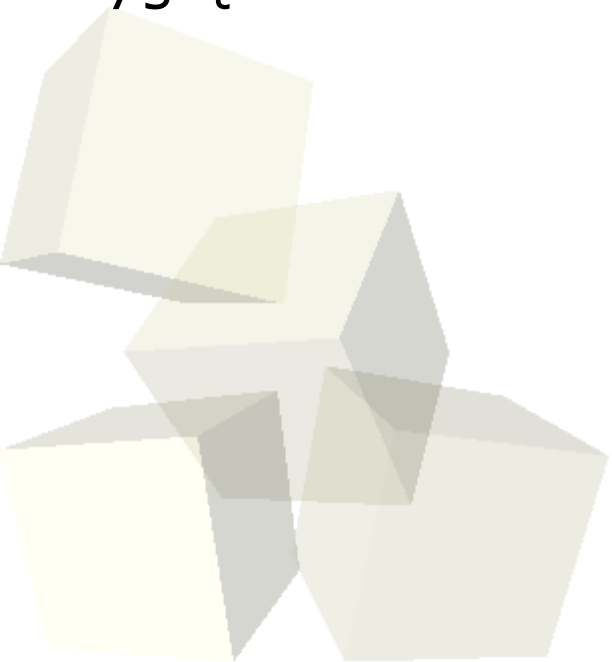


Aplet GUI

Aplety mogą oczywiście posiadać własny interfejs użytkownika (GUI).

- AWT
- Swing

Aplet może jednocześnie korzystać z obydwu typu bibliotek, jednak nie jest to polecane, ze wzgledu na pewne restrykcje (problemy z wyglądem i rozmieszczeniem komponentów).



AWT (Abstract Window ToolKit) – przenośna biblioteka GUI zarówno dla aplikacji desktopowych oraz apletów. Dostarcza wysokopoziomowy komponety wizualnej interkacji dla aplikacji napisanych w języku Java. Obecna w Javie od samego początku.

Cechy AWT:

- Bogaty zbiór komponentów graficznych
- Bardzo stabilny model zdarzeniowy (event-handling model).
- Narzędzia do obsługi grafiki, obrazów umożliwiające zmianę kształtu, koloru, czcionki etc.
- Elastyczne zarządzanie layout-em okna (wygląd obiektów niezależny od rozmiaru okna, czy rozdzielczości)
- Klasy umożliwiające transfer danych (kopiuj/wklej poprzez natywny clipboard).

Komponenty AWT są “cieżkie”, czyli realizowane poprzez użycie graficznych bibliotek GUI systemu operacyjnego.

Swing

Swing w całości jest zaimplementowany w Javie.

Cechy Swing:

- Wszystkie cechy AWT.
- W 100% obsługuje komponenty AWT (Button, Scrollbar, Label, etc.).
- Zbiór dodatkowych wysokopoziomowych komponentów (tree view, list box, tabbed panes etc.).
- Javowy wygląd (niezależny od systemu operacyjnego).

Komponenty Swing są “lekkie” - rysowane za pomocą kodu Javy w obszarze jakiegoś komponentu ciężkiego znajdującego się wyżej w hierarchii zawierania się komponentów (zwykle jest to kontener najwyższego poziomu).

Komponenty “lekkie” mogą być przezroczyste, a zatem mogą przybierać wizualnie dowolne kształty oraz wygląd niezależny od platformy.

Hello world

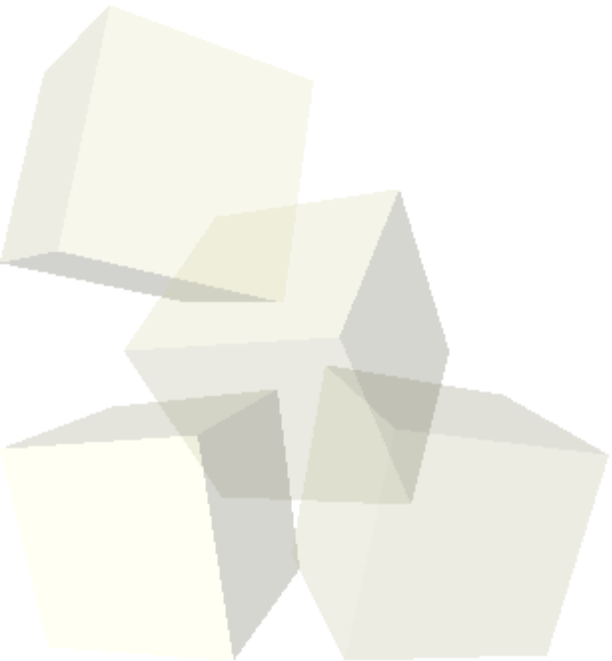
```
import javax.swing.JApplet;
import javax.swing.SwingUtilities;
import javax.swing.JLabel;

public class HelloWorld extends JApplet {
    //Called when this applet is loaded into the browser.
    public void init() {
        //Execute a job on the event-dispatching thread; creating
this applet's GUI.
        try {
            SwingUtilities.invokeAndWait(new Runnable() {
                public void run() {
                    JLabel lbl = new JLabel("Hello World");
                    add(lbl);
                    JButton but = new JButton("Click to start");
                    add(but);
                }
            });
        } catch (Exception e) {
            System.err.println("createGUI didn't complete
successfully");
        }
    }
}
```

Uruchamianie apletów

Aby uruchomić aplet należy do najpierw skompilować (class, jar).

Aplety mogą być uruchamiane na 2 sposoby:

- Za pomocą znaczników <applet> (stara szkoła)
- Za pomocą JNLP (Java Network Launch Protocol)



Umieszczanie apletów w HTML

```
<HTML>
  <HEAD>
    <TITLE>Applet HTML Page</TITLE>
  </HEAD>
  <BODY>
    <H3>Applet HTML Page</H3>
    <P>   <APPLET code="mojprogram.class" width=350
                height=200></APPLET>  </P>
  </BODY>
</HTML>
```

Gdy kod apletu powinien być pobrany z innej lokalizacj niż bieżący folder → parametr **codebase**.

```
<APPLET codebase="http://www.aplety.org/sciezka/do/apletu"
        code="mojprogram.class" width=350 height=200></APPLET>
```

Umieszczanie apletów w HTML

Aplet w postaci jar-a może zawierać dodatkowe elementy np. obrazki.

Przykład uruchomienia apletu (klasa Aplecik) skompilowanego do bajtkodu Javy w toir.jar”

```
<applet code=Aplecik width=256 height=256 archive="toir.jar">
<param name=img value=test.gif>
</applet>
```

Polecenie tworzące jar-a (można dodać “v” - verbose)

jar cf jar-file input-file(s)

jar cvf TicTacToe.jar TicTacToe.class audio images

Wynikowy plik **TicTacToe.jar** zawierać bedzie plik **TicTacToe.class** oraz podkatalogi **audio** i **images**.

Umieszczanie apletów w HTML

Zwiększenie stosu dla apletu (do 256MB) .

```
<applet code="MyApplet.class" width="800" height="600">
    <param name="java_arguments" value="-Xmx256m">
</applet>
```

Składnia znacznika APPLET

```
 <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>
```

Parametry obowiązkowe: CODE, WIDHT, HEIGHT.

JNLP – Java Network Launch Protocol

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="" href="">
    <information>
        <title>Dynamic Tree Demo</title>
        <vendor>Dynamic Team</vendor>
    </information>
    <resources>
        <!-- Application Resources -->
        <j2se version="1.6+"
              href="http://java.sun.com/products/autodl/j2se" />
        <jar href="DynamicTreeDemo.jar" main="true" />
    </resources>
    <applet-desc
         name="Dynamic Tree Demo Applet"
         main-class="components.DynamicTreeApplet"
         width="300"
         height="300">
     </applet-desc>
     <update check="background"/>
</jnlp>
```

JNLP wewnątrz HTML

```
<body>
    ....
    <script src="http://www.java.com/js/deployJava.js"></script>
    <script>
        var attributes = { code:'components.DynamicTreeApplet',
width:300, height:300} ;
        var parameters = {jnlp_href: 'dynamictree-applet.jnlp'} ;
        deployJava.runApplet(attributes, parameters, '1.6');
    </script>
    ....
</body>
```

Parametry wewnątrz i na zewnątrz JNLP

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="" href="">
    ...
    <applet-desc
          name="Applet Takes Params"
          main-class="AppletTakesParams"
          width="800"
          height="50">
              <param name="paramStr" value="someString"/>
              <param name="paramInt" value="22"/>
     </applet-desc>
   ...
</jnlp>
```

```
<script src="http://www.java.com/js/deployJava.js"></script>
 <script>
  var attributes = { code:'AppletTakesParams.class',
  archive:'applet_AppletWithParameters.jar',  width:800,height:50};
  var parameters = {jnlp_href: 'applettakesparams.jnlp',
                    paramOutsideJNLPFile: 'fooOutsideJNLP'};
  deployJava.runApplet(attributes, parameters, '1.4');
</script>
```

Odczyt parametrów apletu

```
public void init() {
  final String  inputStr = getParameter("paramStr");
  final int inputInt = Integer.parseInt(getParameter("paramInt"));
  final String inputOutsideJNLPFile =
                                getParameter("paramOutsideJNLPFile");

}
```