Programski jezik Java

Interno gradivo za predmet

Algoritmi in programski jeziki (4. letnik)

Applets

(neprečiščeno besedilo)

bostjan.vouk@tsc.si

Aplet

- Beseda aplet (applet) asocira angleško govorečim ljudem na majhen program, kar je bil pomen besede v računalniški industriji, preden se je pojavila Java. Aplet pri Javi pomeni vsak program, ki je napisan v programskem jeziku Java, in je pognan iz HTML dokumenta. Pomen programa oziroma aplikacije (application) je ostal nespremenjen in pomeni izvršljiv zapis, ki ga prožimo iz ukazne vrstice. Java nima omejitev glede velikosti in kompleksnosti apletov. Nasprotno, v nekaterih pogledih imajo apleti večjo izrazno moč kot programi napisani v Javi. Kljub temu je večina apletov majhnih, saj imajo komunikacijske poti omejeno hitrost in s tem dolge čase nalaganja.
- Osnovni postopek izdelave apleta v Javi:
  1. Napišemo aplet in ga shranimo v datoteke s končnico java
  2. Napišemo datoteko HTML, kamor vključimo aplet
  3. Aplet prevedemo s prevajalnikom javac, ki izdela datoteke s končnico class
  4. Datoteko HTML si ogledamo s poljubnim brskalnikom ali pa s pregledovalnikom appletviewer

bostjan.vouk@tsc.si

Razlike med programom in apletom

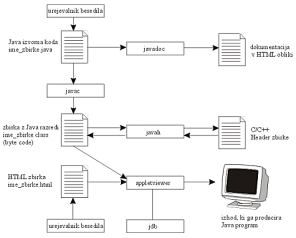
- Gledano s tehnične strani se razlika med apletom in programom nanaša na okolje v katerem tečeta. Program teče v najpreprostejšem okolju. Njegov edini vhod iz okolja so parametri ukazne vrstice. Za razliko od programa potrebuje aplet vrsto informacij, ki jih običajno dobi od pregledovalnika za svetovni splet, v primeru lokalnega poganjanja pa od pregledovalnika apletov (Applet Viewer).
- Aplet potrebuje naslednje informacije:
  - kdaj je inicializiran, kdaj in kje naj se pojavi (nariše) v pregledovalnikovem oknu in seveda kdaj je aktiven oziroma neaktiven.
- Aplet in program imata različne minimalne zahteve. Odločitev kdaj se odločimo za program in kdaj za aplet je v veliki meri odvisna od potreb in zahtev, ki jih imamo. Velja naslednje:
  - Aplikacije, ki niso mrežno naravnane in morajo biti pomnilniško skromne, kličejo po programski implementaciji, medtem ko so aplikacije za Internet in uporabniško prijazne aplikacije nekoliko lažje izvedljive v obliki apleta.

bostjan.vouk@tsc.si

Razlike med programom in apletom

	Java program	Java aplet
uporaba grafike	opcijska	grafičen zavoljo dedovanja
zahteve po pomnilniku	minimalno, kolikor zahteva program	zahteve aplikacije plus zahteve pregledovalnika
Porazdeljenost	naložen z datotečnega sistema ali prirejenega procesa nalaganja razredov	povezan preko HTML -ja in prenesen s pomočjo HTTP-ja
vhodi okolja	parametri ukazne vrstice	parametri o lokaciji in velikosti posredovani s strani pregledovalnika in uporabniški parametri posredovani iz gostujočega HTML dokumenta
rutine, ki jih zahteva navidezni stroj (VM)	main - zagonska rutina	init - inicializacijska rutina start - zagonska rutina stop - zaustavitvena rutina in rutina mirovanja destroy - zaključna rutina paint - rutina risanja oziroma obnavljanja
tipična aplikacija	omrežni strežnik (npr. HTTP), razvijalska orodja, programi za elektronske naprave, poslovne aplikacije	javno dostopni vstopni sistemi za svetovni splet, multimedijske predstavitve, animacije na spletnih straneh

Izgradnja in uporaba apleta



1. Napišemo aplet (.java)
2. Prevedemo s prevajalnikom javac
3. Dobimo zlogokod ("byte code" .class)
4. Naložimo aplet preko omrežja (nalagalnik)
5. Sledi prevajanje kode
6. Podvrženo omejitvam upravitelja varnosti Security Manager
7. Vmesno kodo interpretira JVM znotraj pregledovalnika

Preprost applet

- Preprost applet, ki ne naredi ničesar, je lahko definiran kot razred, ki vsebuje samo metodo **paint()**.
  - import java.awt.*;
    import java.applet.*;
    public class **ime-appleta** extends Applet {
    public void paint(Graphics g) {
    // programski stavki
    }
    }
- Prevajanje:
  - javac JavaAplet.java
- Zagon
  - appletviewer JavaAplet.html

Primer

```
import java.applet.*;
import java.awt.*;

public class mojAplet extends Applet {
    public void init () { System.out.println("Aplet inicializiran"); }
    public void start () { System.out.println("Aplet zacenja - start"); }
    public void stop () {System.out.println("Aplet se je ustavil"); }
    public void destroy () {System.out.println("Aplet zbrisan");}
    public void paint (Graphics g) {g.drawString("Pozdrav", 100, 100);}
}
```

bostjan.vouk@tsc.si

Primer kode apleta

- Za izdelavo apleta potrebujemo tudi gostiteljsko HTML zbirko, v katero aplet vključimo na sledeči način:
  - **<applet Code="mojAplet.class" width=300 height=300 </applet>**
- Splošna oblika:
  - **<applet code=ime_razreda width=300 height=300**

    <param name = ime1 value = vrednost>

    Na tem mestu je aplet, če ga ne vidite in berete to sporočilo, boste za pogajanje apletov potrebovali drug spletni pregledovalnik

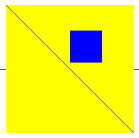
    </applet>
    - Ob nalaganju apleta se izvede konstruktor apleta, pokliče se metoda *init()* in proži metoda *start()* Pri ponovnem nalaganju se najprej pokliče *stop()*, nato *destroy()*, nakar se izvede konstruktor, pokliče *init()* in proži *start()*. Ob odhodu s strani se pokliče metoda *stop()*, ob vrnitvi s spletne strani pa *start()*.

bostjan.vouk@tsc.si

Applet na spletni strani

- **Applet HTML Tag**
- *<applet*
- *[codebase=url]*
- *[alt=text]*
- *[name=appName]*
- *width=wpixels height=hpixels*
- *[align=alignment]*
- *[vspace=vspixels]*
- *[hspace=hspixels]*
- *>*
- *[<param name=pname1 value=value1>]*
- *[<param name=pname2 value=value2>]*
- *……..*
- *[<param name=pnameN value=valueN>]*
- *</applet>*

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="BackgroundForeground" width=200 height=200>
  </applet>
*/

public class BackgroundForeground extends Applet {

 public void paint(Graphics g) {
   setBackground(Color.yellow);
   setForeground(Color.blue);
   g.drawLine(0, 0, 200, 200);
   g.fillRect(100, 40, 50, 50);
 }
}
```



bostjan.vouk@tsc.si

Življenjski cikel Appleta

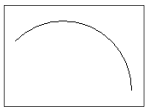
```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="AppletLifecycle" width=300 height=50>
  </applet>
*/
public class AppletLifecycle extends Applet {
  String str = "";
  public void init() {
    str += "init; ";
  }
  public void start() {
    str += "start; ";
  }
  public void stop() {
    str += "stop; ";
  }
  public void destroy() {
    System.out.println("destroy");
  }
  public void paint(Graphics g) {
    g.drawString(str, 10, 25);
  }
}
```

- init() : Klicana samo takrat ko se začne applet izvajati
- start() :Izvajati se začne po metodi init(). Klicana je s strani brskalnika ali applet predvajalnika (applet viewer)
- stop() : ko je applet predvajalnik minimiziran
- destroy() : metoda je klicana s strani brskalnika ali applet predvajalnika (applet viewer) preden se applet konča

bostjan.vouk@tsc.si

Razred Graphics

- *abstract void drawArc(int x, int y, int w, int h, int degreesO, int degrees1)*
- *abstract boolean drawImage(Image img, int x, int y, ImageObserver io)*
- *abstract void drawLine(int x0, int y0, int x1, int y1)*
- *abstract void drawOval(int x, int y, int w, int h)*
- *abstract void drawPolygon(int x[], int y[], int n)*
- *abstract void drawPolyline(int x[], int y[], int n)*
- *void drawRect(int x, int y, int w, int h)*
- *abstract void drawString(String str, int x, int y)*
- *abstract void fillArc(int x, int y, int w, int h, int degree0, int degree1)*
- *abstract void fillOval(int x ,int y, int w, int h)*
- *abstract void fillPolygon(int x[], int y[], int n)*
- *void fillRect(int x, int y, int w, int h)*
- *abstract Color getColor()*
- *abstract Font getFont()*
- *abstract FontMetrics getFontMetrics()*
- *abstract void setColor(Color c)*
- *abstract void setFont(Font f)*



```
import java.applet.Applet;
import java.awt.Graphics;
/*
  <applet code="DrawArc" width=200 height=200>
  </applet>
*/

public class DrawArc extends Applet {

  public void paint(Graphics g) {
    g.drawArc(20, 20, 160, 160, 0, 135);
  }
}
```

bostjan.vouk@tsc.si

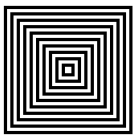
Primer *drawPolygon*

```
public void paint(Graphics g) {
    int xpoints[] = {25, 145, 25, 145, 25};
    int ypoints[] = {25, 25, 145, 145, 25};
    int npoints = 5;

    g.drawPolygon(xpoints, ypoints, npoints);
}
```

bostjan.vouk@tsc.si

Razred Graphics

```
import java.awt.*;
import java.applet.*;
public class Kvadrati extends Applet {

public void paint(Graphics g) {
int size=200;
// Set up the off-screen image (buf) for double-buffering
Image buf = createImage (size,size);
Graphics bufg = buf.getGraphics( );
// Draw into the off-screen buffer
for(int i = 0;i<size; i += 5)
{
bufg.setColor(i % 10 == 0 ? Color.black: Color.white);
//green, yellow, orange, pink, magenta, ...
bufg.fillRect(i, i, size -i * 2, size - i * 2) ;
}
// Now copy the off-screen buffer onto the screen
g.drawImage(buf,0,0,null);
}
}
```



bostjan.vouk@tsc.si

Uporaba barv

- **Color Constructors**
- *Color(int red, int green, int blue)*
- *Color(int rgb)*
- *Color(float r, float g, float b)*

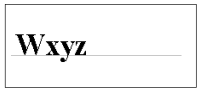
- **Method of Graphics Class**
- *static Color decode(String str) throws NumberFormatException*
- *static int HSBtoRGB(float h, float s, float b)*
- *static float[] RGBtoHSB(int r, int g, int b, float hsb[])*
- *Color brighter()*
- *Color darker()*
- *boolean equals(Object obj)*
- *int getBlue()*
- *int getGreen()*
- *int getRGB()*
- *int getRed()*


```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
/*
  <applet code="BlueString" width=300 height=100>
  </applet>
*/

public class BlueString extends Applet {

  public void paint(Graphics g) {
    g.setColor(Color.blue);
    g.drawString("Blue String", 100, 50);
  }
}
```

bostjan.vouk@tsc.si

Prikazovanje teksta

- **Font Constructor**
- *Font(String name, int style, int ps)*
- **setFont() Method**
- *abstract void setFont(Font font)*
- **FontMetrics Constructor**
- *FontMetrics(Font font)*

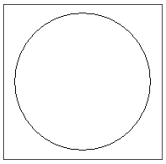



```
import java.applet.Applet;
import java.awt.*;
/*
  <applet code="FontDemo" width=200 height=200>
  </applet>
*/

public class FontDemo extends Applet {

  public void paint(Graphics g) {

    // Draw baseline
    int baseline = 100;
    g.setColor(Color.lightGray);
    g.drawLine(0, baseline, 200, baseline);

    // Draw String
    g.setFont(new Font("Serif", Font.BOLD, 36));
    g.setColor(Color.black);
    g.drawString("Wxyz", 5, baseline);
  }
}
```

bostjan.vouk@tsc.si

Dimenzije Appleta

- **getSize() Method**
- *Dimension getSize()*
- **Dimension Constructors**
- *Dimension()*
- *Dimension(Dimension d)*
- *Dimension(int w, int h)*

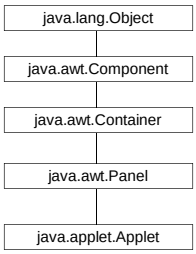


```
import java.applet.*;
import java.awt.*;
/*
 <applet code="Circle" width=200 height=200>
 </applet>
*/

public class Circle extends Applet {

 public void paint(Graphics g) {
   Dimension d = getSize();
   int xc = d.width / 2;
   int yc = d.height / 2;
   int radius = (int)((d.width < d.height) ?  0.4 * d.width : 0.4 * d.height);
   g.drawOval(xc - radius, yc - radius, 2 * radius, 2 * radius);
 }
}
```

bostjan.vouk@tsc.si

Testiranje parametrov appleta

Applet razred



Applet in njegovi super razredi

```
import java.applet.*;
import java.awt.*;
/*
 <applet code="AppletParameters" width=300 height=300>
 <param name="background" value="0xffffff">
 <param name="foreground" value="0x000000">
 <param name="message" value="Testiranje  parametrov appleta">
 </applet>
*/

public class AppletParameters extends Applet {

 public void paint(Graphics g) {
   String background = getParameter("background");
   String foreground = getParameter("foreground");
   String message = getParameter("message");
   setBackground(Color.decode(background));
   setForeground(Color.decode(foreground));
   Font font = getFont();
   FontMetrics fm = getFontMetrics(font);
   Dimension d = getSize();
   int x = (d.width - fm.stringWidth(message)) / 2;
   int y = d.height / 2;
   g.drawString(message, x, y);
 }
}
```

bostjan.vouk@tsc.si

Vmesnik vsebine

- **AppletContext Interface**
- *Applet getApplet(String appName)*
- *Enumeration getApplets()*
- *AudioClip getAudioClip(URL url)*
- *Image getImage(URL url)*
- *void showDocument(URL url)*
- *void showDocument(URL url, String target)*
- *void showStatus(String str)*

```
import java.applet.*;
import java.awt.*;
import java.net.*;
/*
 <applet code="ShowDocument" width=200 height=50>
 </applet>
*/

public class ShowDocument extends Applet {

 public void init() {
   AppletContext ac = getAppletContext();
   try {
     URL url = new URL("http://www.tsc.si");
     ac.showDocument(url, "frame2");
   }
   catch(Exception e) {
     showStatus("Exception: " + e);
    }
 }

 public void paint(Graphics g) {
   g.drawString("ShowDocument Applet", 10, 25);
 }
}
```

bostjan.vouk@tsc.si

Uporaba slik

- **getImage() Methods**
- *Image getImage(URL url)*
- *Image getImage(URL base, String fileName)*
- **drawImage() Methods**
- *abstract boolean drawImage(Image img, int x, int y, ImageObserver io)*

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="DrawImage" width=280 height=280>
  <param name="file" value="slika.jpg">
  </applet>
*/

public class DrawImage extends Applet {
  Image image;

  public void init() {
    image = getImage(getDocumentBase(),
      getParameter("file"));
  }

  public void paint(Graphics g) {
    g.drawImage(image, 0, 0, this);
  }
}
```

bostjan.vouk@tsc.si

Uporaba niti

- **update() and repaint() Methods**
- *repaint() : request an update of the applet display*
- *update() : clears the applet display with the background color and then invokes the paint() method*

-
```
import java.applet.*;
import java.awt.*;
public class Counter extends Applet
implements Runnable {
  int counter;
  Thread t;

  public void init() {
    // Initialize counter
    counter = 0;
    // Start thread
    t = new Thread(this);
    t.start();
  }
```

6

```
  public void run() {
    try {
      while(true) {
        // Request a repaint
        repaint();
        //Sleep before displaying next count
        Thread.sleep(1000);
        //Increment counter
        ++counter;
      }
    }
    catch(Exception e) {  }
  }
  public void paint(Graphics g) {

    // Set Font
    g.setFont(new Font("Serif", Font.BOLD, 36));
    // Get font metrics
    FontMetrics fm = g.getFontMetrics();
    // Display counter
    String str = "" + counter;
    Dimension d = getSize();
    int x = d.width / 2 - fm.stringWidth(str) / 2;
    g.drawString(str, x, d.height / 2);
  }
}
```

bostjan.vouk@tsc.si

Double Buffering (No Double Buffering)

- Double Buffering can be used to avoid display "flicker" in applets.
-
```
import java.applet.*;
import java.awt.*;
/*
  <applet code="NoDoubleBuffer" width=300
        height=100>
  </applet>
*/
public class NoDoubleBuffer extends Applet
implements Runnable {
  int x = 0;
  Thread t;
  public void init() {
    // Start thread
    t = new Thread(this);
    t.start();
  }
  public void run() {
    try {
      while(true) {
```

-
```
        // Request a repaint
        repaint();
        //Sleep before update
        Thread.sleep(100);
      }
    }
    catch(Exception e) {
    }
  }
  public void paint(Graphics g) {
    // Draw filled circle
    Dimension d = getSize();
    g.fillOval(x, d.height / 4, 50, 50);
    // Increment x
    x += 5;
    if (x + 50 > d.width)
      x = 0;
  }
}
```

bostjan.vouk@tsc.si

Double Buffering (With Double Buffering)

```
import java.applet.*;
import java.awt.*;
/* <applet code="DoubleBuffer" width=300
      height=100>
  </applet> */
public class DoubleBuffer extends Applet
implements Runnable {
 int x = 0;
 Thread t;
 Image buffer;
 Graphics bufferg;
 public void init() {
  // Start thread
  t = new Thread(this);
  t.start();
  // Create buffer
  Dimension d = getSize();
  buffer = createImage(d.width, d.height);
 }
 public void run() {
  try {
   while(true) {

    //Request a repaint
    repaint();
    Thread.sleep(100); // Sleep before update
   }
  }
  catch(Exception e) {
  }
 }
 public void update(Graphics g) {
  paint(g);
 }
 public void paint(Graphics g) {
  //Get graphics object for buffer
  if (bufferg == null)
   bufferg = buffer.getGraphics();
  //Draw to buffer
  Dimension d = getSize();
  bufferg.setColor(Color.white);
  bufferg.fillRect(0, 0, d.width, d.height);
  bufferg.setColor(Color.black);
  bufferg.fillOval(x, d.height / 4, 50, 50);
  //Update screen
  g.drawImage(buffer, 0, 0, this);
  //Increment x
  x += 5;
  if (x + 50 > d.width)
   x = 0;
 }
}
```

bostjan.vouk@tsc.si