SLOVENSKÁ TECHNICKÁ UNIVERZITA
Fakulta chemickej a potravinárskej technológie
Katedra informatizácie a riadenia procesov
Radlinského 9, 812 37 Bratislava

MODELOVANIE A RIADENIE V JAZYKU JAVA

Školiteľ: Ing. Ľuboš Čirka

Vypracoval: Miroslav Jančík

Bratislava 2005

Moje poďakovanie patrí vedúcemu bakalárskej práce Ing. Ľubošovi Čirkovi za pomoc pri získavaní vedomostí z oblasti programovania v jazyku JAVA a za rady, pripomienky, ktoré mi poskytol počas práce na bakalárskom projekte.

OBSAH

1 ÚVOD

JAVA je jednoduchý, objektovo orientovaný, robustný a nezávislý na architektúre programovací jazyk.

- **Jednoduchosť** – obsahuje len minimálne množstvo jazykových konštrukcií, ktoré sú odvodené od existujúcich programovacích jazykov
- **Objektovo orientovaný** – znamená, že konkrétne dáta a metódy popisujú stav a chovanie objektu. V objekte sú zoskupené triedy (class), ktoré sú usporiadané smerom od všeobecnejších implementácii ku konkrétnejším. Java sama obsahuje veľké množstvo balíkov, ktoré sa dajú využiť pri programovaní.
- **Robustnosť** – od samých počiatkov bola Java navrhnutá na písanie software pre spotrebnú elektroniku, preto jazyk musí podporovať vytváranie robustného softvéru (Applety, aplikácie pre mobilné telefóny, Java Server Pages a pod.).
- **Nezávislý na architektúre –** programy písané v Jave nie sú priamo prekladané do strojového kódu procesoru, ale do tzv. bajtového kódu. Výhodou je prenášanie programov medzi jednotlivými architektúrami, ako sú Linux, Solaris, MacOS a rôzne klony UNIXu.

2 JAVA APPLET

Applety sú vecou, ktorá stala hneď na počiatku vývoja a popularity Javy ako takej. Je to vlastne program, ktorý sa dá implementovať do prostredia webových stránok. Applet je vlastne trieda, ktorá je vložená do webovej stránky pomocou HTML TAGu (Hypertext Markup Language) **<APPLET>**. Vždy keď prehliadač narazí na túto značku, začne sťahovanie príslušnej triedy zo servera, applet je potom spustený na strane klienta v časti okna, ktorá mu bola predtým vyhradená.

Applet je spustený v bezpečnostnom sandboxe (obmedzený rozsah), aby sa zabránilo nebezpečenstvám na klientskej strane. Teda nie sú plnohodnotnými programami, pokiaľ im to nie je explicitne povolené. Preto sú niekedy applety podpísané certifikátmi vydanými certifikačnými autoritami.

Každý applet je odvodený od triedy **Applet** z balíka **java.applet**. Okrem toho pri vytváraní appletu, je potrebné importovať balík **java.awt**, ktorý umožňuje používať a zobrazovať grafické užívateľské prostredie **AWT** (Abstract Windows Toolkit). Applety sa

dajú potom považovať za úvod do spôsobu programovania z grafickým užívateľským prostredím (GUI).

2.1 Požiadavky na hardware a software

Nároky na systém nie sú veľmi veľké. Postačujúcou konfiguráciou počítača je procesor 486 s pevným diskom kapacity 500 MB pre inštaláciu Windowsu s Internet Explorerom resp. iným prehliadačom webových stránok (Mozilla, Firefox, Opera a pod.). Keďže applety nie sú kompilované do strojového kódu, je potrebné mať nainštalovaný **JRE** (Java Runtime Enviroment). Verzia JRE je závislá od použitých balíkov v applete. Veľkosť operačnej pamäte (RAM) závisí od veľkosti pamäte, ktorú si applet pre svoj beh alokuje (vyhradí).

2.2 Základ appletu

Základ appletu tvorí trieda odvodená od triedy **java.applet.Applet.** Táto trieda by mala definovať 5 základných metód, ktoré umožňujú ovládať applet.

```
public class JadroAppletu extends Applet {

        public void init() {
        //volaná prehliadačom pri inicializácii appletu
        }

        public void start() {
        //volaná vždy keď má byť stránka appletu zobrazená
        }

        public void stop() {
        //volaná vždy keď má byť stránka appletu skrytá (ukončená)
        }

        public void destroy() {
        //volaná prehliadačom za účelom odstránenia appletu
        }


        public void paint() {
        //volaná vždy keď je potrebné applet znovu vykresliť
        }
}
```

Aby sa dal applet na WWW stránke použiť, musí sa vložiť nasledujúci HTML kód:

```
<applet code=”JadroAppletu” width=”300” height=”200”>
</applet>
```

2.3 Vývojové prostredie a kompilácia appletu

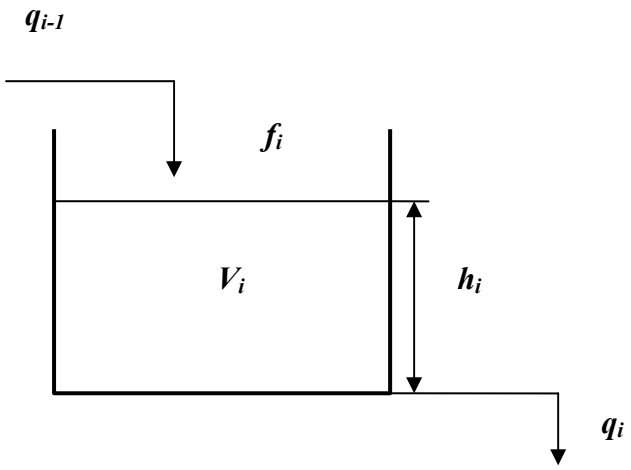
Pre vytváranie appletov, ako aj iných aplikácii založených na technológii Java, je vhodné používať nejaké vývojové prostredie. Existuje ich mnoho, niektoré sú voľne dostupné (NetBeans, JEdit apod.), niektoré nie (JBuilder, IntelliJ IDEA). Hlavnou výhodou väčšiny prostredí je farebné odlišovanie písaného kódu a priebežné upozorňovanie na jednotlivé syntaktické chyby. K vývojovému prostrediu je nutné mať doinštalovaný balík **JDK** (Java Development Kit), ktorý zabezpečuje potrebné balíky (packages) objektov, tried, ale hlavne kompilátor do bajtového kódu.

Samotná kompilácia appletu sa dá realizovať dvoma spôsobmi. Prvým spôsobom je pomocou vývojového prostredia výberom s ponuky. Napríklad v prostredí IntelliJ IDEA ponuka **Build**, a následne **Compile** (resp. klávesová skratka Ctrl+Shit+F9). Zložitejší spôsob je daný súbor nakopírovať do adresára, kde sa nachádza JDK a najmä súbor javac.exe (pre platformu Windows). Potom kompiláciu je potrebné vykonať v tomto adresári príkazom **javac.exe NazovAppletu.java**. Názov appletu musí byť však zhodný z názvom triedy! Ak všetko prebehlo správne, tak sa v danom adresári nachádza súbor **NazovAppletu.class**.

3 ZÁSOBNÍKY KVAPALINY

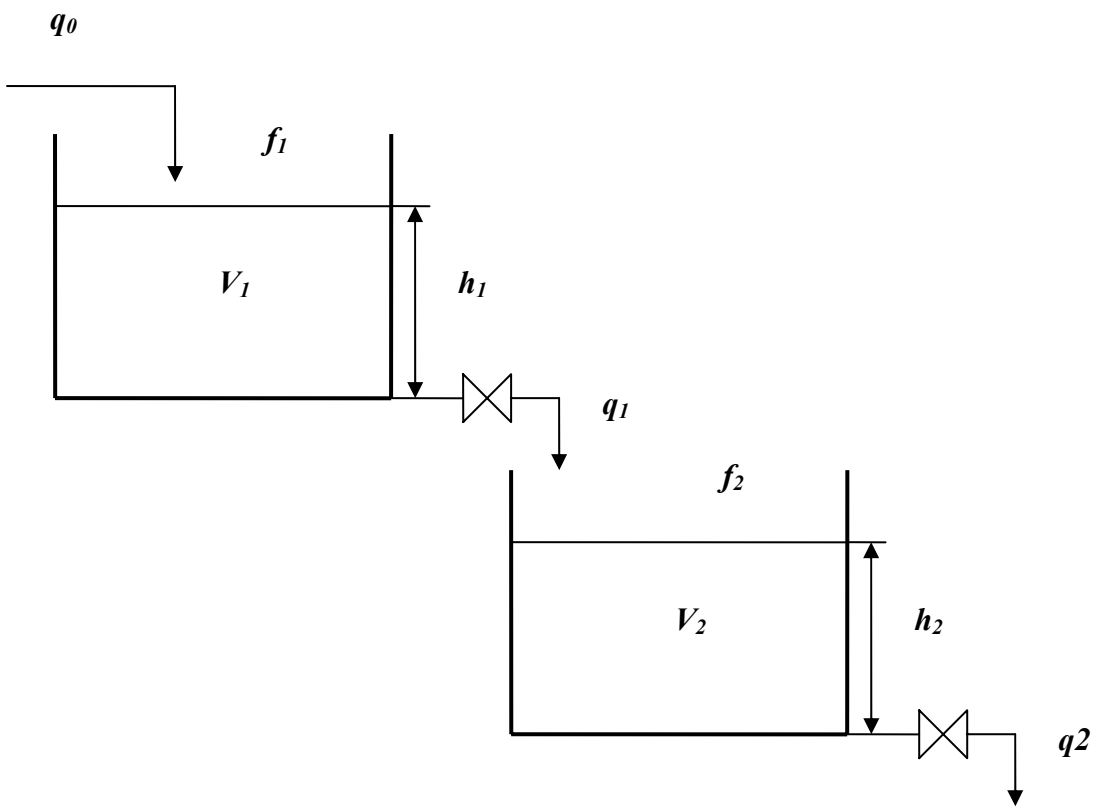
Zásobníky kvapaliny sú prietokové systémy s jednoduchou akumuláciou hmoty. Sú najrozšírenejšími a najpoužívanejšími zariadeniami chemickej technológie ako aj priemyslu.

3.1 Druhy zapojenia zásobníkov kvapaliny

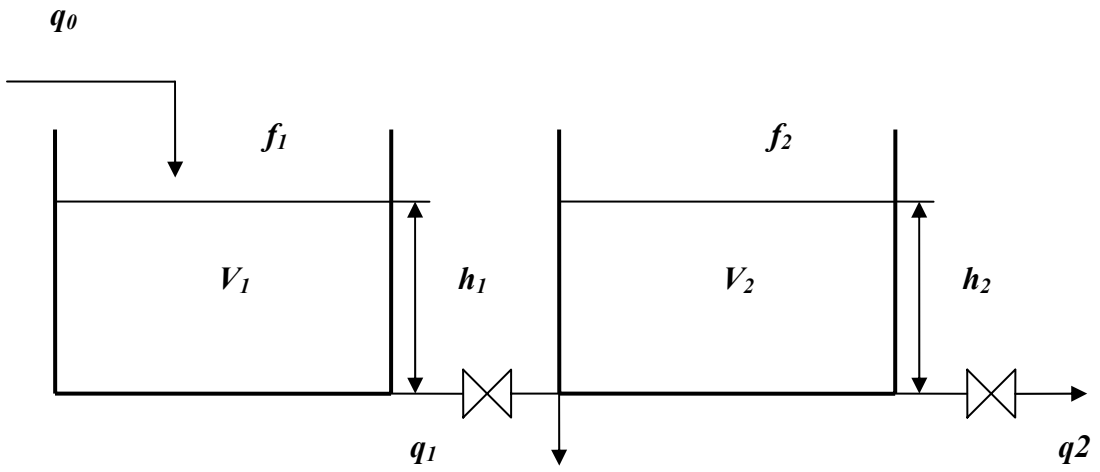


Na obrázku je znázornená schéma zásobníka kvapaliny. Je reprezentovaný nádržou, do ktorej priteká kvapalina o prietoku $\mathbf{q_{i-1}}$ **[m.s^{-1}]** a vyteká o prietoku $\mathbf{q_i}$ **[m.s^{-1}]**. Ako výstupná veličina sa najčastejšie uvažuje výška hladiny kvapaliny zásobníka $\mathbf{h_i}$ **[m]**, ktorá je vo väčšine prípadov aj regulovanou veličinou. Zásobník s kvapalinou patrí medzi akumulačný systém 1. rádu. Parameter, ktorý ovplyvňuje jeho dynamické vlastnosti je plocha povrchu hladiny kvapaliny v zásobníku $\mathbf{f_i}$ **[m]**.

Sériovým zapojením n zásobníkov vzniká systém n-tého rádu. Nádoby jednotlivých zásobníkov môžu byť spojené bez interakcie, t.j. 2 alebo viac zásobníkov je spojených, pričom jednotlivé nádoby nie sú na jednej úrovni.



Ďalším druhom sériového zapojenia zásobníkov je s interakciou. To znamená, že jednotlivé zásobníky sú vedľa seba na takej istej úrovni.



3.2 Matematický model zásobníkov kvapaliny bez interakcie

Ak **V [m^3]** je objem kvapaliny v zásobníku, pre i-tú nádrž systému, potom platí rovnica dynamiky materiálovej bilancie.

$$\frac{dV}{dt} = q_{i-1} - q_i$$

resp.

$$f_i \frac{dh_i}{dt} + q_i = q_{i-1}$$

Keď uvažujeme aj konštanty ventilov $\mathbf{k_i}$ zásobníkov, potom závislosť medzi prietokom $\mathbf{q_i}$ a výškou hladiny $\mathbf{h_i}$ je nelineárna, tvaru:

$$q_i = k_i \sqrt{h_i}$$

Ak máme dva zásobníky zapojené sériovo a bez interakcie potom môžeme písať:

$$\frac{dh_1}{dt} = \frac{q_0 - k_{11} \cdot \sqrt{h_1}}{f_1}$$

Platí pre zmenu hladiny v prvom zásobníku. Pre druhý zásobník môžeme napísať:

$$\frac{dh_2}{dt} = \frac{k_{11} \cdot \sqrt{h_1} - k_{22} \sqrt{h_2}}{f_2}$$

Tento matematický model bol použitý pri vytváraní Java apletu simulácie dvoch zásobníkov bez interakcie. Pričom ako vstupný parameter pre výšky **h1** a **h2** boli použité ustálené hladiny zadané na začiatku spolu so všetkými parametrami zásobníkov.

3.3 Matematický model zásobníkov kvapaliny s interakciou

Zásobníky s interakciou tvoria systém druhého rádu, pre ktorý platia vzťahy:

$$q_{i-1} = k_{i-1}\sqrt{h_{i-1} - h_i}$$

$$q_i = k_i\sqrt{h_i}$$

Matematický model dvoch zásobníkov zapojených bez interakcie vyzerá následovne:

$$\frac{dh_1}{dt} = \frac{q_0 - k_{11}\cdot\sqrt{h_1 - h_2}}{f_1}$$

resp. pre druhý zásobník:

$$\frac{dh_2}{dt} = \frac{k_{11}\cdot\sqrt{h_1 - h_2} - k_{22}\cdot\sqrt{h_2}}{f_2}$$

4 APPLETY ZÁSOBNÍKOV

Tieto applety slúžia na simuláciu zmeny výšok jednotlivých hladín v dvoch zásobníkoch kvapaliny bez interakcie, ako aj s interakciou. Simulácia prebieha v reálnom čase. Okrem to pri spustení je vykreslená závislosť zmeny výšok hladín od ich pôvodných ustálených hodnôt od času.

4.1 Popis appletu zasobniky.java

Applet Zasoniky.java slúži na simuláciu zmien výšok dvoch zásobníkov bez interakcie.

Výpis apletu **zasobniky.java** (po kompilácii zasobniky.class):

```
import java.applet.Applet;
import java.awt.*;
import java.lang.*;
import java.text.NumberFormat;
import java.text.DecimalFormat;
```

Kľúčové slovo **import** slúži na vkladanie jednotlivých balíkov tried jazyka, ktoré chceme použiť.

- **java.lang** – obsahuje všeobecne použiteľné triedy tvoriace základ jazyka
- **java.awt** - triedy pre vytváranie a podporu grafického rozhrania aplikácií
- **java.applet –** triedy pre vytváranie programov (appletov), ktoré sú použiteľné pre internetové prehliadače
- **java.text –** pomocné triedy pre spracovanie a formátovanie textu

```
public class Zasobniky extends Applet
   implements Runnable

 {
```

Začiatok appletu (štruktúra popísaná v kapitole 2.2) a implementácia rozhrania **Runnable**, ktoré zabezpečuje simuláciu.

```
  int stav1 = 0;
  int stav2 = 0;
  double q0s=3;           //prietok pri pôvodných ustálených výškach hladín
  double q0=5;            //nový vstupný prietok
  double k11=2.5;         //konštanta ventilu 1
  double k22=2.7;         //konštanta ventilu 2
  double F1=3;            //plocha povrchu hladiny kvapaliny v 1. zásobníku
  double F2=2.64;         //plocha povrchu hladiny kvapaliny v 2. zásobníku
  double h1s=1;           //ustálená výška hladiny v zásobníku 1 pri q0s
  double h2s=1.2346;      //ustálená výška hladiny v zásobníku 2 pri q0s
  double x1=h1s;          //ustálená výška v zásobníku 1
```

```
double x2=h2s;          //ustálená výška v zásobníku 2
int cassimulacie=30;    //čas simulácie
double korekcia;        //korekcia prierezov pri simulácii
double sys1;            //dh1/dt
double sys2;            //dh2/dt
volatile Thread thread; //vlakno pre animaciu
```

Definícia globálnych premenných, ktoré sa používajú v celom applete. Každá premenná obsahuje meno a typ (**double –** reálne čísla, **int** – celé čísla).

```
public void start() {
  thread=new Thread(this);
  thread.start();
}
```

Metóda **start()** sa stáva základom pre vlákno pretože trieda appletu implementuje rozhranie **runnable** a nastáva tu štart vlákna pre simuláciu.

```
public void stop() {
    thread=null;
}
```

Tu nastáva zastavenie simulácie po uplynutí času zadaného v premennej **cassimulacie.** Zastavenie musí znamenať aj zastavenie vlákna.

```
public void init() {
  setBackground(Color.WHITE);
}
```

Pri spustení prebieha metóda **init()**. Nastaví sa tu akurát farba pozadia prehliadača, pomocou metódy **setBackground(Color.WHITE).**

```
public void paint(Graphics g)
{
  int i;
  double sys1graph;
  double sys2graph;
  double q0graph=q0;
  double k11graph=k11;
  double k22graph=k22;
  double F1graph=F1;
  double F2graph=F2;
  double x1graph=h1s;
```

```
double x2graph=h2s;
int znamienko1=1;
int znamienko2=1;
double yscale,xscale;
double ynula;
double min=9999,max=0;
```

Deklarácia premenných, ktoré sa používajú v metóde **paint().** V tejto metóde sa vykresľuje celá simulácia zásobníkov kvapaliny.

```
for(i=0;i<=cassimulacie;i=i+1){
 sys1graph=(q0graph-k11graph*(Math.sqrt(x1graph)))/F1graph;
 sys2graph=(k11graph*(Math.sqrt(x1graph))-k22graph*(Math.sqrt(x2graph)))/F2graph;
 x1graph=x1graph+sys1graph;
 x2graph=x2graph+sys2graph;
}
```

Pomocou cyklu s pevným počtom opakovaní **for(),** ktorý prebieha až po stanovený čas simulácie sa vypočíta podľa matematického modelu zásobníkov bez interakcie nová ustálená výška hladín v zásobníkoch. Použitá tu je trieda **Math** a metóda **sqrt(double)** na výpočet odmocniny.

```
  NumberFormat format = new DecimalFormat("##.####");
```

Týmto riadkom v programe sa vytvára premenná **format** typu **NumberFormat.** Táto premenná vo svojej podstate definuje odkaz na objekt. Potom dôjde k fyzickému vytvoreniu objektu v pamäti pomocou príkazu **new**. Súčasne dôjde k priradeniu odkazu na nový objekt do premennej **format.** Ide tu vlastne o nastavenie formátovania čísel pri výpise.

```
g.drawString("Dva zásobníky kvapaliny bez interakcie",110,20);
g.drawString("t[s]",810,365);
g.drawString("h1[m]",485,60);
g.drawString("q0[m3/s]="+format.format(q0),10,20);
g.drawString("h1[m]="+format.format(x1),160,150);
g.drawString("h2[m]="+format.format(x2),380,400);
g.drawString("k11="+format.format(k11),195,230);
g.drawString("k22="+format.format(k22),415,480);
g.drawString("F1[m2]="+format.format(F1),75,265);
g.drawString("F2[m2]="+format.format(F2),290,515);
g.drawString("Vstupné údaje: h1s[m]="+format.format(h1s),550,400);
g.drawString("h2s[m]="+format.format(h2s),635,415);
g.drawString("q0s[m3/s]="+format.format(q0s),635,430);
```

```
g.drawString("Výstupné údaje: h1s[m]="+format.format(x1graph),550,460);
g.drawString("h2s[m]="+format.format(x2graph),640,475);
g.drawString("q0[m3/s]="+format.format(q0),640,490);
g.drawString("Čas simulácie[s]="+format.format(cassimulacie),640,505);
```

Metódou **drawString(*reťazec, pozícia x, pozícia y*)** triedy **Graphics** dochádza k výpisu reťazca. Pomocou tejto metódy sa vypisujú aktuálne stavy, jednotlivé parametre a iný text na obrazovku.

```
double pole[] = new double[4];
pole[0]=h1s;
pole[1]=h2s;
pole[2]=x1graph;
pole[3]=x2graph;
for(i=0;i<=3;i++){
 if(pole[i] < min) min=pole[i];
 if(pole[i] > max) max=pole[i];
}
yscale=Math.abs(max - min);
```

Cyklom s pevným počtom opakovaní **for()** vypočíta mierka y-ovej osi grafu závislosti zmeny výšok od času. Najskôr sa deklaruje pole **double pole[] = new double[počet prvkov]** do ktorého sa priradia jednotlivé ustálené výšky hladín zásobníkov pred a po simulácii. Potom po prebehnutí cyklu, kde sa zistí maximálna a minimálna hodnota sa vypočíta rozdiel týchto hodnôt, čo tvorí základ pre mierku.

```
ynula=0;
if(min == h1s) ynula=h1s;
if(min == h2s) ynula=h2s;
if(min == x1graph) ynula=x1graph;
if(min == x2graph) ynula=x2graph;
```

Príkazom **if** za zisťuje počiatočná hodnota y-ovej osi porovnávaním minimálnej hodnoty zo všetkých hladín.

```
//definícia počiatku súradnicovej sústavy pre závislosť h1
int x1old = 520, x1new =520, y1old=350, y1new=350;
y1old=y1old-(int)((h1s-ynula)/yscale*300);
y1new=y1new-(int)((h1s-ynula)/yscale*300);

//definícia počiatku súradnicovej sústavy pre závislosť h2
int x2old = 520, x2new =520, y2old=350, y2new=350;
```

```
        y2old=y2old-(int)((h2s-ynula)/yscale*300);
        y2new=y2new-(int)((h2s-ynula)/yscale*300);

         //kontrola, či nový ustálený stav bude mať menšiu výšku hladín
        if(x1graph < h1s){
         x1old = 520; x1new =520; y1old=50; y1new=50;
         y1old=y1old-(int)((ynula-x1graph)/yscale*300);
         y1new=y1new-(int)((ynula-x1graph)/yscale*300);
         znamienko1=-1;
        }

         if(x2graph < h2s){
         x2old = 520; x2new =520; y2old=50; y2new=50;
         y2old=y2old-(int)((ynula-x2graph)/yscale*300);
         y2new=y2new-(int)((ynula-x2graph)/yscale*300);
         znamienko2=-1;
        }
```

V týchto riadkoch dochádza k nastaveniu hodnôt začiatku závislosti jednotlivých výšok, kvôli vykresľovaniu. Je tu použitý príkaz **(int)**, ktorý prevádza hodnotu **double** na **int**. Tu sa vyskytuje preto, lebo jednotlivé obrazovkové body majú celočíselný charakter.

```
xscale=300/cassimulacie;          //vypočet mierky pre x-ovú os

x1graph=h1s;
x2graph=h2s;

for(i=0;i<=cassimulacie-1;i=i+1){
  sys1graph=(q0graph-k11graph*(Math.sqrt(x1graph)))/F1graph;
  sys2graph=(k11graph*(Math.sqrt(x1graph))-k22graph*(Math.sqrt(x2graph)))/F2graph;
  x1graph=x1graph+sys1graph;
  x2graph=x2graph+sys2graph;

if(x1old == y1new){    //výpočet súradníc konečného bodu závislosti h1=f(t)
   x1new=x1new+(int)xscale;
   y1new=y1new-((int)((Math.abs(sys1graph)/yscale)*300)*znamienko1);
}
else{
   x1old=x1new;
   y1old=y1new;
   x1new=x1new+(int)xscale;
   y1new = y1new - (int) (Math.abs(sys1graph) / yscale * 300) * znamienko1;
}
```

```
if(x2old == y2new){    //výpočet hodnoty konečného bodu závislosti h2=f(t)
x2new=x2new+(int)xscale;
y2new = y2new - (int) (Math.abs(sys2graph) / yscale * 300) * znamienko2;
}
else{
 x2old=x2new;
 y2old=y2new;
 x2new=x2new+(int)xscale;
 y2new=y2new-((int)((Math.abs(sys2graph)/yscale)*300)*znamienko2);
}

g.setColor(new Color(255,252,0));
g.drawLine(x1old,y1old,x1new,y1new);   //zakreslenie casti ciary pre h1
g.setColor(new Color(18,255,0));
g.drawLine(x2old,y2old,x2new,y2new);   //zakreslenie casti ciary pre h2
}
```

Cyklom **for()** a pomocou matematického modelu sa vypočítavajú konečné hodnoty časti jednotlivých čiar závislosti výšok od času. Na konci cyklu sa zakreslia metódou **g.drawLine(x1, y1, x2 y2),** pričom farba jednotlivých čiar je nastavená metódou **g.setColor(farba)**.

```
        //zakreslenie x-ovej a y-ovej osi
        g.setColor(new Color(0,0,0));
        g.drawLine(520,350,820,350);
        g.drawLine(520,350,520,50);

        //vykreslenie zásobníka 1
        g.drawLine(50,50,50,250);
        g.drawLine(50,250,150,250);
        g.drawLine(150,250,150,50);
        g.drawLine(150,50,50,50);

        //prítokové potrubie do zásobníka 1
        g.drawLine(55,50,55,30);
        g.drawLine(55,30,10,30);

        //prítokové potrubie do zásobníka 2
        g.drawLine(150,245,200,245);
        g.drawLine(220,245,275,245);
        g.drawLine(275,245,275,300);
```

```
        //ventil 1
        g.drawLine(200,235,200,255);
        g.drawLine(220,235,220,255);
        g.drawLine(200,235,220,255);
        g.drawLine(200,255,220,235);

        //zásobník 2
        g.drawLine(270,300,270,500);
        g.drawLine(270,500,370,500);
        g.drawLine(370,500,370,300);
        g.drawLine(370,300,270,300);

        //výtokové potrubie zo zásobníka 2
        g.drawLine(370,495,420,495);   //
        g.drawLine(440,495,500,495);
        g.drawLine(500,495,500,550);

        //ventil 2
        g.drawLine(420,485,420,505);
        g.drawLine(440,485,440,505);
        g.drawLine(420,485,440,505);
        g.drawLine(420,505,440,485);


        g.setColor(new Color(26,145,200));
        g.fillRect(51,51,99,199);
        g.setColor(new Color(255,255,255));
        g.fillRect(51,51,99,199-stav1);
        g.setColor(new Color(26,145,200));
        g.fillRect(271,301,99,199);
        g.setColor(new Color(255,255,255));
        g.fillRect(271,301,99,199-stav2);
    }
```

Koniec metódy **paint()** appletu. V tejto časti sa metódou **g.drawLine(x1, y1, x2, y2)** vykreslia zásobníky spolu s aktuálnymi výškami hladín. Metóda **g.fillRect(x, y, šírka, výška)** kreslí modrou farbou (na nastavenie farby slúži **g.setColor(farba)**) vyplnený obdĺžnik, kde výška sa vypočítava odčítaním premennej **stav1** resp. **stav2** stav od aktuálnej hodnoty. Nové hodnoty premenných **stav1** a **stav2** sa priradzujú v metóde **run()**.

```
public void run() {
```

```
//hodnota v ms, na ktorú sa má zastaviť prebiehajúce vlákno
int delay = 1000;
int cas = cassimulacie;

while (thread == Thread.currentThread()){

     //zastavenie behu appletu ak hodnota aktuálnej výšky je rovná nule
     if (x1 < 0){
      x1=0;
      this.stop();
     }
     if(x2 < 0){
      x2=0;
      this.stop();
     }

     //koniec reálnej simulácie po uplynutí času simulácie
     if(cas == 0){
       this.stop();
     }
     else{
      cas=cas-1;
     }

     //výpočet aktuálnych stavou výšok hladín h1 a h2
     sys1 = (q0 - k11 * (Math.sqrt(x1))) / F1;
     sys2 = (k11 * (Math.sqrt(x1)) - k22 * (Math.sqrt(x2))) / F2;
     x1=x1+sys1;
     x2=x2+sys2;

     if(F1 > F2){
       korekcia=F1/F2;
       stav1=(int)(x1*20);
       stav2=(int)(x2*20*korekcia);
      }
      if(F1 < F2){
        korekcia=F2/F1;
        stav1=(int)(x1*korekcia)*20;
        stav2=(int)x2*20;
       }

       if(F1 == F2){
         stav1=(int)x1*20;
         stav2=(int)x2*20;
        }
```

```
            //vykreslenie metódy paint() po zmene hodnot stav1 a stav2
            repaint();

            try {
              //zastavenie behu vlákna na hodnotu delay
              Thread.sleep(delay);
            }
            catch (InterruptedException e) {
              e.printStackTrace();
             }
          }

        }//koniec metódy run()
}//koniec appletu Zasobniky
```

V metóde **run()** dochádza k výpočtu aktuálnych výšok hladín (premenné **stav1** a **stav2**), k zastaveniu behu vlákna na čas (ms) daný hodnotou premennej **delay**, ktorá je nastavená na hodnotu 1000, kvôli reálnemu behu simulácie. Následne, po prebehnutí výpočtou a zastavenia vlákna je volaná metóda **repaint(),** ktorá znovu prekreslí zásobníky ale už s novými výškami hladín.

4.2 Popis appletu zasobniky1.java

Simulácia dvoch zásobníkov s interakciou realizovaná appletom zasobniky1.java. Programový kód appletu zasobniky1.java je skoro rovnaký s appletom zasobniky1.java. Rozdiely sú len v metódach **run()** a **paint()**, kde je zmenený matematický model a nahradený kódom:

```
sys1 = (q0 - k11 * (Math.sqrt(x1-x2))) / F1;
sys2 = (k11 * (Math.sqrt(x1-x2)) - k22 * (Math.sqrt(x2))) / F2;
x1=x1+sys1;
x2=x2+sys2;
```

5 PRÍKLAD SIMULÁCIE

5.1 Zásobníky kvapaliny bez interakcie

Vstupné parametre:

Vstupný prietok kvapaliny pri ustálených výškach hladín $\mathbf{q_{0s}} = 3$

Ustálená výška hladiny v zásobníku 1 $\mathbf{h_{1s}} = 1$

Ustálená výška hladiny v zásobníku 2 $\mathbf{h_{2s}} = 1.2346$

Konštanta ventilu 1 $\mathbf{k_{11}} = 2.5$

Konštanta ventilu 2 $\mathbf{k_{22}} = 2.7$

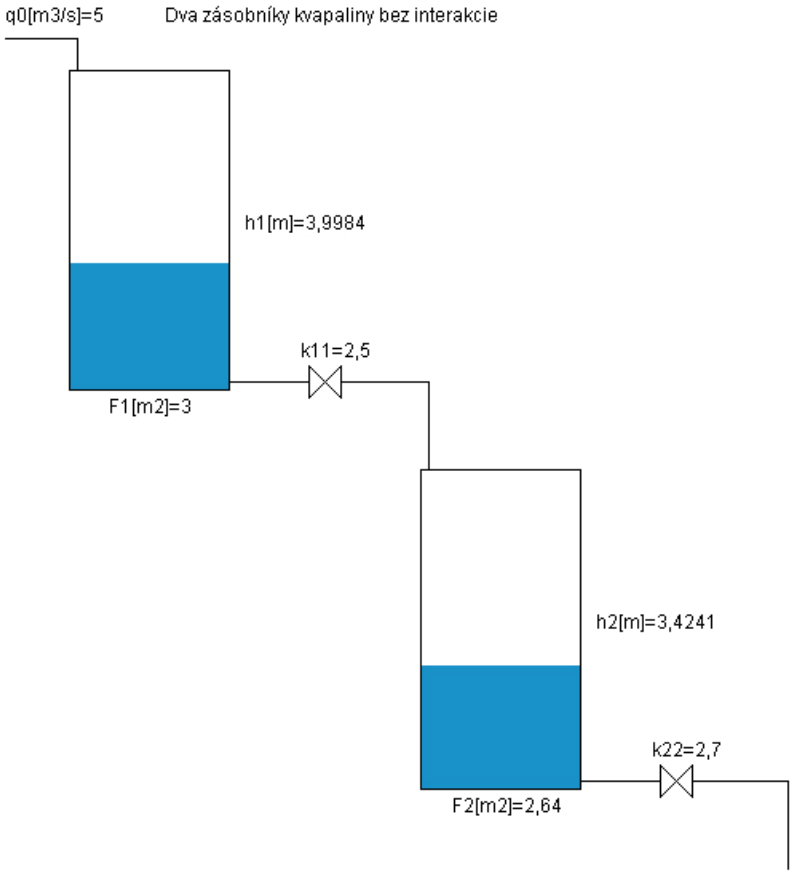
Plocha prierezu zásobníka 1 $\mathbf{F_1} = 3$

Plocha prierezu zásobníka 2 $\mathbf{F_2} = 2.64$

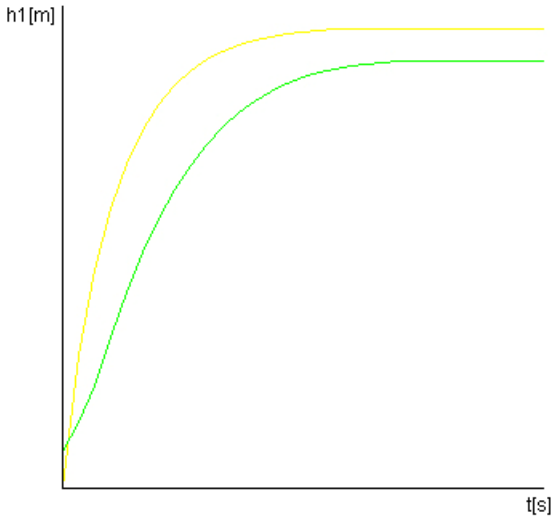
Nový vstupný prietok kvapaliny $\mathbf{q_0} = 5$

Čas simulácie $\mathbf{t} = 30$

Parametre simulácie sa nastavia v globálnych premmenných appletu **Zasobniky.java**, ešte pred kompiláciou. Po kompilácii treba znovu načítať webovú stránku **Zasobniky.html.**



Po znovunačítaní webovej stránky **Zasobniky.html** sa zobrazí applet so simuláciou. Sú to 2 zásobníky bez interakcie a ich reálna simulácia v danom čase. Vedľa každého zásobníka sa nachádza aktuálna výška hladiny, ktorá sa z časom mení. Takisto sa menia hladiny kvapalín v jednotlivých zásobníkoch (zobrazené modrou farbou). Simulácia prebieha až po stanovený čas simulácie, ktorý je zadaný v sekundách.



Vstupné údaje: h1s[m]=1
h2s[m]=1,2346
q0s[m3/s]=3

Výstupné údaje: h1s[m]=3,9984
h2s[m]=3,4241
q0[m3/s]=5
Čas simulácie[s]=30

Okrem reálnej simulácie applet vykresľuje graf závislosti zmeny výšok hladín jednotlivých zasobníkov od času. Žltou čiarou je znázornený priebeh zmeny výšky hladiny v zásobníku 1 oproti predchádzajúcemu ustálenému stavu $\mathbf{h_{1}s}$ pri prietoku $\mathbf{q_{0s}}$, zelená čiara charakterizuje priebeh zmeny výšky hladiny v zásobníku 2 voči predchádzajúcemu ustálenému stavu $\mathbf{h_{2}s}$ pri prietoku $\mathbf{q_{0s}}$. Nové ustálené hladiny $\mathbf{h_{1s}}$, $\mathbf{h_{2s}}$ pri prietoku $\mathbf{q_{0}}$ sú zobrazené pod grafom. Graf aj nové ustálené hladiny v zásobníkoch applet nevykresľuje reálne s časom, ale hneď pri štarte.

5.2 Zásobníky kvapaliny s interakciou

Vstupné parametre:

Vstupný prietok kvapaliny pri ustálených výškach hladín $\mathbf{q_{0s}} = 8$

Ustálená výška hladiny v zásobníku 1 $\mathbf{h_{1s}} = 2.5242$

Ustálená výška hladiny v zásobníku 2 $\mathbf{h_{2s}} = 1.5242$

Konštanta ventilu 1 $\mathbf{k_{11}} = 8$

Konštanta ventilu 2 $\mathbf{k_{22}} = 6.48$

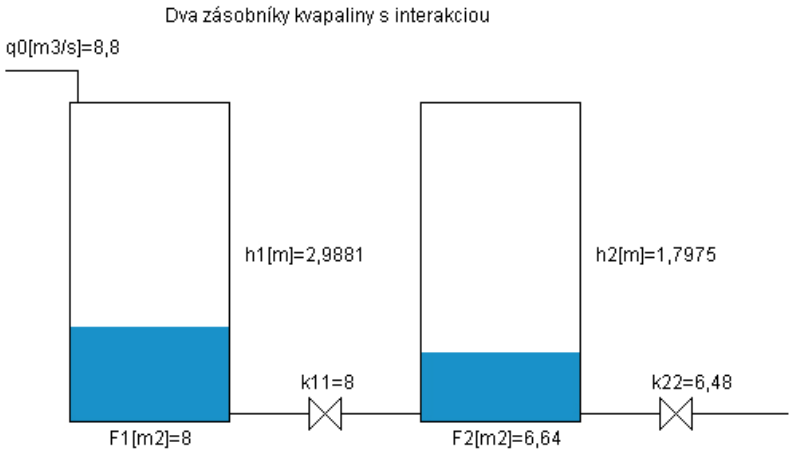
Plocha prierezu zásobníka 1 $\mathbf{F_1} = 8$

Plocha prierezu zásobníka 2 $\mathbf{F_2} = 6.64$

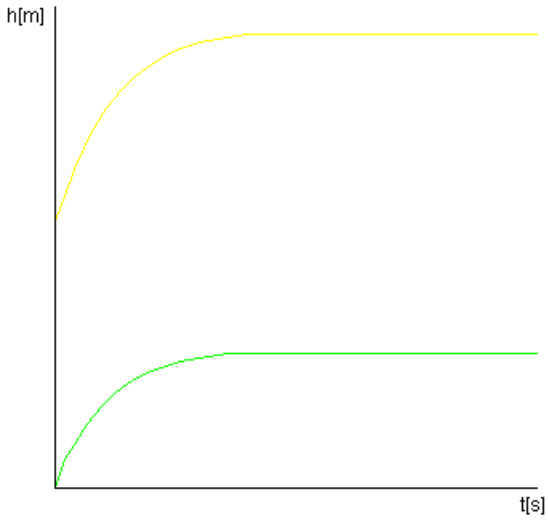
Nový vstupný prietok kvapaliny $\mathbf{q_0} = 8.8$

Čas simulácie $\mathbf{t} = 50$

Parametre simulácie sa nastavia v globálnych premmenných appletu **Zasobniky1.java**, ešte pred kompiláciou. Po kompilácii treba znovu načítať webovú stránku **Zasobniky1.html.**



Na obrázku je znázornený priebeh zmeny výšok hladín pri novom vstupnom prietoku oproti pôvodným výškam pri pôvodnom prietoku. Zapojenie zásobníkov je sériové a s interakciou. Simulácia prebieha v reálnom čase.



Vstupné údaje: h1s[m]=2,2542
h2s[m]=1,2542
q0s[m3/s]=8

Výstupné údaje: h1s[m]=3,0537
h2s[m]=1,8439
q0[m3/s]=8,8
Čas simulácie[s]=50

Podobne ako pri zásobníkoch zapojených bez interakcie applet zobrazuje graf závislosti zmeny výšok hladín jednotlivých zásobníkov od času. A takisto hodnoty nových ustálených stavov hladín.

6 ZÁVER

Cieľom mojej práce bolo odsimulovať v jazyku Java priebeh výšok hladín dvoch zásobníkov (zapojených s interakciou a bez interakcie) pri zmene vstupného prietoku. Bolo potrebné zvládnuť základy tvorby Java appletov. V budúcnosti by som chcel s touto prácou pokračovať v diplomovej práci a chcel by som dopracovať niektoré časti, ako sú: zadávanie údajov pri priamo pri štarte alebo behu appletu, vykresľovanie trendovej čiary reálne s časom a pod. Ďalej by som chcel vytvoriť Java applety modelov iných technologických procesov (napr. výmenníky tepla, chemický reaktor a pod.).

7 ZOZNAM POUŽITEJ LITERATÚRY

[1] Ľubomír Brúha, Java – Hotová řešení, Vydavateľstvo Computer Press, Brno, 2003.

[2] Bogdan Kiszka, 1001 tipú a trikú pro programovaní v jazyce Java, Vydavateľstvo Computer Press, Brno, 2003.

[3] Alojz Mészáros, Základy automatizácie, Vydavateľstvo STU, Bratislava, 1997.

[4] API specification for the Java 2 Platform Standard Edition 5.0, dostupné na http://java.sun.com/j2se/1.5.0/docs/api/