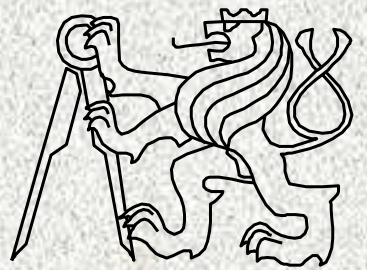
VLÁKNA



A0B36PR2-Programování 2

Fakulta elektrotechnická
České vysoké učení technické

Vlákna v Javě

- Oblasti použití
  - Nesoulad časových nároků různých nezávislých částí programu
  - Dlouhé periferní přenosy, čekání na odezvu na vstupu
  - Simulační výpočty, opakující se výpočty
  - Úlohy typu producent/konzument, server/klient

Vlákna v Javě

- Vlákna vytváříme:
  - potomkem třídy **`Thread`**
    - přepisujeme **`run()`** a pouštíme **`start()`**
  - implementací rozhraní **`Runnable`**
    - metoda **`run()`**

- Další metody
  - **`getName()`** – vrací jméno vlákna
  - **`getPriority()`** – vrací prioritu vlákna
  - **`isAlive()`** – zda vlákno běží
  - **`join()`** – pozastaví provádění až do ukončení vlákna
  - **`sleep()`** – pozastaví vlákno na určenou dobu
  - …

Nezávislá „paralelní“ vlákna, náhodné střídání

Metoda **`run()`** je překryta v obou přístupech, popisuje činnost vlákna

- Metoda **`run()`** se spouští metodou **`start()`**
- V konstruktoru je předáno jméno vlákna
- Metoda **`getName()`** předá metodě **`run()`** jméno vlákna
- Metoda **`sleep()`** uspí vlákno a začne pracovat jiné
- Metoda **`main()`** vytvoří 3 vlákna
- Vlákna končí skončením svého **`run()`**

Třída **Thread**

```
public class Vlakno1 extends Thread {
  public Vlakno1(String jmeno) {
    super(jmeno); // předání jména vlákna
  }
   public void run() {
     for (int i = 1;  i <= 3;  i++) {
      System.out.println(i + ". " + getName()); // vrací jméno vlákna
      try {
          Thread.sleep(1000); // uspání na 1 sec., šance pro jiná
      } catch(InterruptedException e) {
          System.out.println("Jsem vzhuru - " + getName());
      }  // pro předčasné probuzení
     }
   }

   public static void main(String[] args) {
     Vlakno1 vlAhoj = new Vlakno1("ahoj");
     vlAhoj.start();
     new Vlakno1("nazdar").start();
     new Vlakno1("cao").start();
  }
}
```

1. nazdar
1. cao
1. ahoj
2. ahoj
2. cao
2. nazdar
3. ahoj
3. cao
3. nazdar

Rozhraní Runnable

```
public class Vlakno1R implements Runnable {
   @Override
   public void run() {
      // vrátí jméno aktuálního vlákna
      String jmeno = Thread.currentThread().getName();

      for (int i = 1; i <= 3; i++) {
         System.out.println(i + ". " + jmeno);
         try {
            Thread.sleep(1000);
         } catch (InterruptedException e) {
            System.out.println("Jsem vzhuru - " + jmeno);
         }
      }
   }

   public static void main(String[] args) {
      Vlakno1R vl = new Vlakno1R();
      Thread t1 = new Thread(vl, "ahoj"),
             t2 = new Thread(vl, “nazdar"),
             t3 = new Thread(vl, “cao");
      t1.start(); t2.start(); t3.start();
    }
}
```

1. ahoj
1. cao
1. nazdar
2. nazdar
2. cao
2. ahoj
3. nazdar
3. ahoj
3. cao

Sdílení dat mezi vlákny a náhodné spouštění

```
public class Vlakno implements Runnable {
  int sdilena = 0;
  public void run() {
   for(int i = 0; i<100;i++, sdilena++) {
     System.out.printf("Sdilena:%5d, i:%5d%n", sdilena, i);
  } }

public static void main(String[] args) {
     Vlakno v1 = new Vlakno();
     Thread t1 = new Thread(v1), t2 = new Thread(v1);
     //obě vlákna musí být nad stejným
      // objektem typu Runnable - zde v1
    t2.start(); t1.start();
    System.out.printf("Konec hlavniho vlakna");
} }
```

```
Konec hlavniho vlakna
Sdilena:    0, i:    0
Sdilena:    1, i:    1
Sdilena:    2, i:    2
.....
Sdilena:   14, i:   14
Sdilena:    0, i:    0
Sdilena:   15, i:   15
...
Sdilena:  109, i:   98
Sdilena:  110, i:   99
Sdilena:   94, i:   11
Sdilena:  112, i:   12
....

Sdilena:  197, i:   97
Sdilena:  198, i:   98
Sdilena:  199, i:   99
```

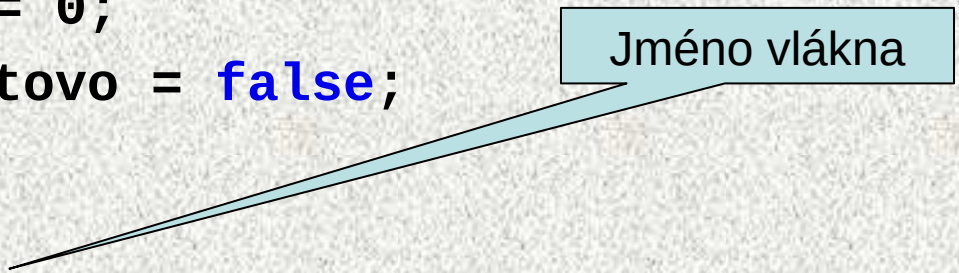
Řízená spolupráce dvou vláken

- Vlákno bude sloužit pro čtení souboru o jednom milionu řádků
- „Souběžné“ vlákno bude „občas“ informovat o stavu výpočtu
  - Druhé vlákno bude vypisovat mezisoučty
- Řešení předávání informace mezi vlákny proměnnými třídy, statickými proměnnými – lepší řešení synchronizací …
- Důsledné oddělení výkonné části programu a části informační

Vlákno pro čtení

```
class ReadVl extends Thread {
  FileReader fr;
  BufferedReader in;
  String jmenoSouboru;
  static public long suma = 0;
  static public boolean hotovo = false;

  ReadVl(String jmeno) {
    super("Vlakno pro cteni");
    jmenoSouboru = jmeno;
  }
```



Vlákno pro čtení

```
public void run() {
    String radka;

    try {
      fr = new FileReader(jmenoSouboru);
      in = new BufferedReader(fr);
      while((radka = in.readLine()) != null) {
        suma += Integer.parseInt(radka);
        Thread.yield();
      }
    } catch (IOException e) {
        System.out.println("Chyba v souboru!");
    } finally {
        fr.close();
        hotovo = true;
    }
  }
}
```

Přečtení souboru, signál pro

Vlákno pro vypisování – „pravidelné“ střídání

```
public class Vlakno2 extends Thread {
public void run() {
    while (ReadVl.hotovo == false) {
      System.out.print(ReadVl.suma + "\n");
      Thread.yield();
    } // yield() předává řízení dobrovolně
    System.out.println(ReadVl.suma);
  }

  public static void main(String[] args) {
    ReadVl vlCteni = new ReadVl("data.txt");
    vlCteni.start();
    Vlakno2 vlVypis = new Vlakno2();
    vlVypis.start();
  }
}
```

Přečtení souboru

Vlákno se vzdává řízení a předává jinému vláknu, po skončení se začíná opět zde

Start vlákna

Start vlákna

Vlákno pro vypisování, střídání nepravidelné, třída Thread

```
public class Vlakno3 extends Thread {
public void run() {
    while (ReadVl.hotovo == false) {
      System.out.print(ReadVl.suma + "\n");
      try {
        Thread.sleep(100);  // 100 milisekund
      }
      catch (InterruptedException e) {
         e.printStackTrace();}
    }
    System.out.println(ReadVl.suma);
  }

  public static void main(String[] args) {
    ReadVl vlCteni = new ReadVl("data.txt");
    vlCteni.start();
    Vlakno3 vlVypis = new Vlakno3();
    vlVypis.start();
  }
}
```

Vlákno se vzdává na 100ms řízení a předává jinému vláknu, po skončení se začíná opět zde, nesymetrické časy, nepřevezme řízení, až za 100 ms

Vlákno pro vypisování,nepravidelné, rozhraní Runnable

```
public class Vlakno5 implements Runnable {

  public void run() {
    while (ReadVl.hotovo == false) {
      System.out.print(ReadVl.suma + "\n");
      try {
        Thread.sleep(100);  // 100 milisekund
      }
      catch (InterruptedException e) {}
    }
    System.out.println(ReadVl.suma);
  }
  public static void main(String[] args) {
    ReadVl vlCteni = new ReadVl("data.txt"); // potomek Thread
    vlCteni.start();
    Vlakno5 vlVypis = new Vlakno5(); // rozhraní Runnable
    Thread zobrazVl = new Thread(vlVypis);
    zobrazVl.start();
  }
}
```

spuštění vlákna

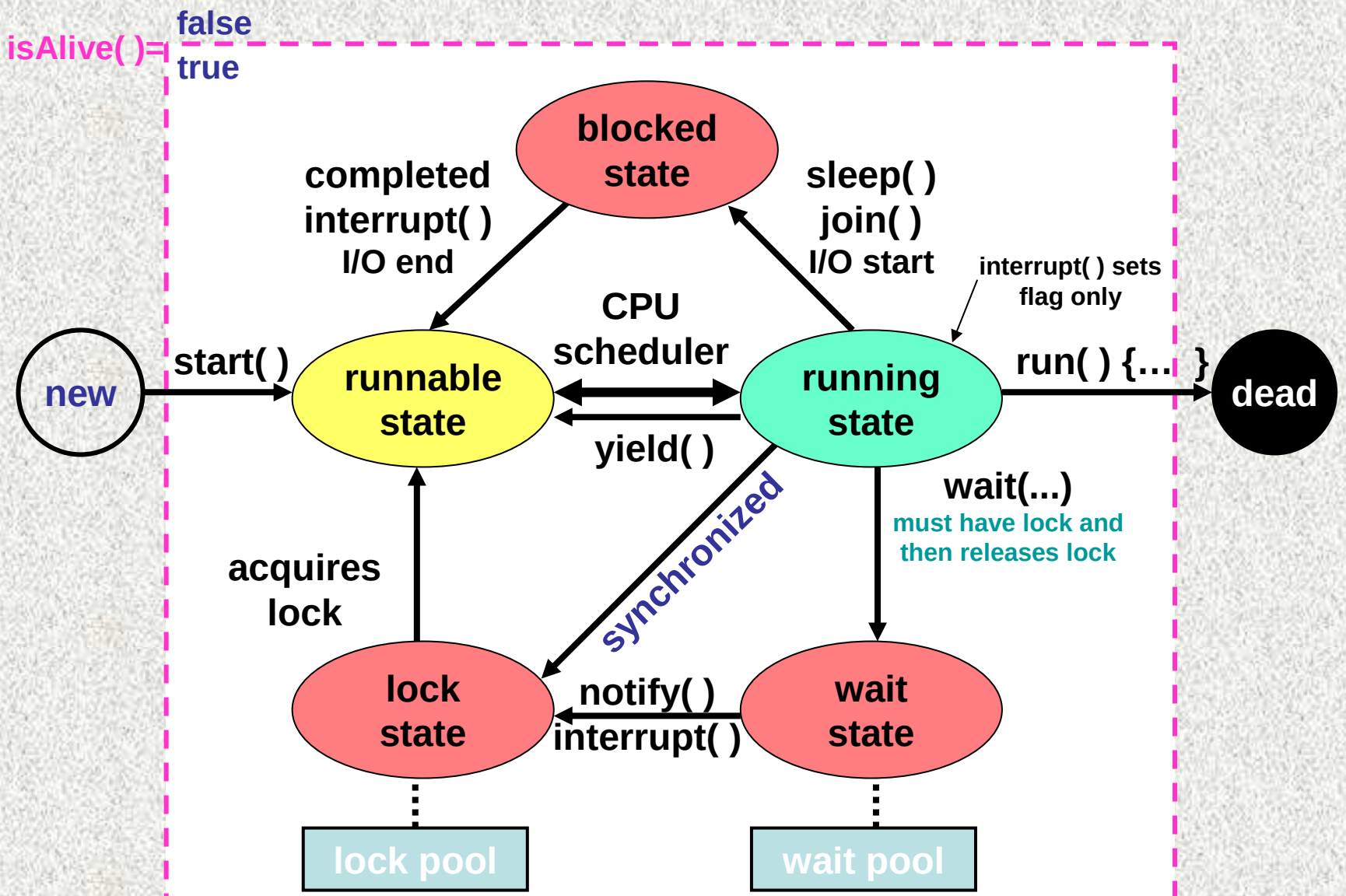
Stavy vlákna

- Nové vlákno, spuštěno **`start()`**
- Běhuschopné **`(runnable),`** je spuštěno, ale neběží, čeká na předání řízení, potom jedno je **`běžící (running)`**
  - jedno z vláken je běžící, ostatní čekají
- Neběhuschopné
  - Blokováno (blocked)
    - uspáno **`sleep(), join(),`** čeká na **O/I**
  - Čekající (wait)
    - čeká **`wait()`**
- Mrtvé vlákno, skončila metoda **`run()`**

Doba a pořadí předávání řízení závisí na:

- na stavu okolních vláken,
- na prioritě vlákna,
- na schopnostech OS.

Stavy vlákna



Priorita vláken

Jsou-li běhuschopná dvě vlákna, pak je řízení předáno tomu vláknu, které má vyšší prioritu a je ve stavu runnable

O předání řízení (přidělení CPU) se stará JVM Scheduler.

- Nastavení priority **`setPriority()`**
- Zjištění priority **`getPriority()`**
- Hodnoty priority
  - **`MAX_PRIORITY  - 10`**
  - **`MIN_PRIORITY  -  1`**
  - **`NORM_PRIORITY -  5`**

Priorita vláken

Pravidla plánovače:

- Běží vždy to, co má z běhuschopných vláken nejvyšší prioritu
- Je-li více vláken se stejnou prioritou, je řízení postupně předáváno všem v náhodném pořadí, předání **`yield()`**
- Vlákno s nižší prioritou mohou získat řízení, jen když vlákna s vyšší prioritou se dostanou do neběhuschopného stavu, vlákno s vyšší prioritou nelze přinutit k předání řízení standardním mechanismem plánování, jen zásahem zvenku
- Pokud se do běhuschopného stavu dostane vlákno s vyšší prioritou, je běžící vlákno okamžitě přinuceno předat řízení ve prospěch tohoto vlákna (preemtivní plánování)

Priorita vláken - příklad

- Čekání na vstup - typický případ, běžící vlákno je přerušeno vláknem vyšší priority
- Vlákno je automaticky uvedeno do stavu neběhuschopné, když čeká na vstup/výstup
- Vstup/výstup má nejvyšší prioritu, kdykoli přeruší, když je možnost předávání dat
- Není třeba další synchronizace, spolupráce je asynchronní

Pomocná třída Vstup

```
class Vstup extends Thread {
  public boolean hotovo = false;
  public void run() {
    byte[] pole = new byte[10];
    Thread.currentThread().setPriority(MAX_PRIORITY);
    while (hotovo == false) {
      try {
        System.in.read(pole);
        if (pole[0] == 'K') {
          hotovo = true;
        }
      }
      catch (IOException e) {
        System.out.println("Chyba vstu
        hotovo = true;
      }
    }
  }
}
```

nastavení nejvyšší priority

Čte nějaké hodnoty ze vstupu a ukládá je do pole b.

Priorita vláken - čekání na vstup

```
public class Vlakno7 extends Thread {
  public void run() {
    long i = 0;
    while (true) {
      System.out.println(i++);
    }
  }
  public static void main(String[] args) {
    Vstup vlVstup = new Vstup();
    vlVstup.start();
    Vlakno7 vlVypis = new Vlakno7();
// démon – sám skončí s posledním nedémonickým vláknem
    vlVypis.setDaemon(true); // musíme nastavit před startem
    vlVypis.start();
  }
}
```

nejvyšší priorita

Priorita vláken - nastavování priorit

```
class MojeVlakno5 implements Runnable {
  private boolean spusteno = true;
  private String jmeno;

  public MojeVlakno5(String tName){
      jmeno = tName;
  }

  public void run(){
      while (spusteno){
         System.out.println(jmeno + " běží");
      }
  }

  public void stop(){
      spusteno = false;
      System.out.println(jmeno + " zastaveno");
  }
}
```

Priorita vláken - nastavování priorit

```
public class DemoVlakno5 {
   public static void main(String[] args) {
    Thread.currentThread().setPriority(10);

    MojeVlakno5 lowPriority = new MojeVlakno5("nízkou prioritou");
    Thread tLow = new Thread (lowPriority);
    tLow.setPriority(3);
    tLow.start();

    MojeVlakno5 highPriority = new MojeVlakno5(7,"vysokou prioritou");
    Thread tHigh = new Thread (highPriority);
    tHigh.setPriority(7);
    tHigh.start();

    try {
        Thread.sleep(10000);
    } catch (InterruptedException e) {
        System.out.println("Hlavní vlákno probuzeno");
    }
    lowPriority.stop(); // nastavení prom. spusteno na false
    highPriority.stop();
   }
}
```

```
Vlakno s nízkou prioritou spuštěno
Vlakno s vysokou prioritou spuštěno
Vlakno s vysokou prioritou běží
Vlakno s nízkou prioritou běží
Vlakno s nízkou prioritou zastaveno
Vlakno s vysokou prioritou zastaveno
```

Předávání řízení

```
public class DemoVlakno6 {
    public static void main(String[] args) {
     MojeVlakno vlakno1 = new MojeVlakno("1");
     Thread  t1 = new Thread(vlakno1);
     t1.start();
…
System.out.println("Stav vlákna isAlive");
System.out.println("Vlakno1: " + t1.isAlive());
…
        try{
 System.out.println("Vlakno: join");
 t1.join();
…
        }catch(InterruptedException e){
 System.out.println("Výjimka: Hlavní vlákno přerušeno");
        }
 System.out.println("Stav vlákna isAlive");
 System.out.println("Vlakno1: " + t1.isAlive());
…
 System.out.println("Výjimka: Hlavní vlákno ukončeno");
    }
}
```

Testování, zda vláno ještě běží

Vlákna čekají na ukončení vlákna t1

Testování, zda vláno ještě běží

Předávání řízení

```
class MojeVlakno implements Runnable{
 String jmenoVlakna;

 MojeVlakno (String vlaknoName){
  JmenoVlakna = vlaknoName;
 }

 public void run(){
  try{
    System.out.println("Vlákno: " + jmenoVlakna);
    Thread.sleep(2000);
  }catch(InterruptedException e){
    System.out.println("Výjimka:" + jmenoVlakna + "probuzeno");
  }
    System.out.println("Ukončení vlákna" + jmenoVlakna);
 }
}
```

Předávání řízení

```
Stav vlákna isAlive
Vlakno1: true
Vlakno2: true
Vlakno3: true
Vlákno: 3
Vlákno: 1
Vlakno4: true
Vlakno: join
Vlákno: 2
Vlákno: 4
Ukončení vlákna1
Ukončení vlákna2
Ukončení vlákna4
Ukončení vlákna3
Stav vlákna isAlive
Vlakno1: false
Vlakno2: false
Vlakno3: false
Vlakno4: false
Výjimka: Hlavní vlákno ukončeno
```

Synchronizace činnosti vláken

- Vlákna spolupracují, problém sdílení datového prostoru
  - Problém nedeterminovanosti přístupu ke společným prostorům
    - Řešení pomocí proměnné třídy nevhodné!!
- Možné řešení je tzv. monitor (kritická sekce)
  - objekt, který vláknu zpřístupní zdroj,
  - v daném okamžiku aktivně používat monitor jen jedno vlákno
  - „pro daný časový interval vlákno vlastní monitor“, monitor smí vlastnit jen jedno vlákno
  - vlákno běží, jen když vlastní monitor, jinak čeká
- Všechny objekty v Javě „mají“ monitor
  - Vlákno vstoupí do monitoru voláním metody s přívlastkem `synchronized` (lze i příkaz) - tuto metodu nazýváme synchronizovanou
  - O vláknu, které úspěšně jako první zavolá synchronizovanou metodu říkáme, že je „uvnitř“ metody a má k dispozici všechny zdroje používanými touto metodou
  - Jiné vlákno, které volá synchronizovanou metodu čeká, dokud aktivní vlákno synchronizovanou metodu neopustí (nebo uvolní zámek pomocí wait)

Synchronizace činnosti vláken - příklad

- Výpis jména v závorkách, tři fáze
  - Levá závorka
  - Vlastní text
  - Pravá závorka

(((XaverHonzaJana)
)
)

((Xaver(Honza)
Jana)
)

(Honza)
(Xaver)
(Jana)

Synchronizace vláken

```
class Zavorky {

   synchronized void zobrazit(String s){
      System.out.print ("(");
      try{
          Thread.sleep(1000);
      } catch (InterruptedException e){
          System.out.println("Probuzeno");
      }

      System.out.print(s);

      try {
          Thread.sleep(1000);
      } catch(InterruptedException e){
          System.out.println(„Probuzeno");
      }
      System.out.print(")");
   }
}
```

Zajistí synchronizaci

Synchronizace vláken

```
class MyThread extends Thread {
   private String s1;
   private Zavorky z1;

   public MyThread(Zavorky z2, String s2){
       z1 = z2;
       s1 = s2;
   }

   public void run(){
   // synchronizovaný blok – synchr. bude probíhat podle
   // monitoru objektu z1
       synchronized(z1){
          z1.zobrazit(s1);
       }
   }}
```

Synchronizace vláken

```
public class DemoVlakno2 {
    public static void main(String[] args) {
      Zavorky z3= new Zavorky();
      MyThread name1 = new MyThread(z3, "Honza");
      MyThread name2 = new MyThread(z3, "Jana");
      MyThread name3 = new MyThread(z3, "Xaver");
      name1.start(); name2.start(); name3.start();

      try {
         name1.v.join();
         name2.v.join();
         name3.v.join();
      } catch (InterruptedException v){
         System.out.println("Preruseno");
      }
    }
}
```

```
public void run(){
  synchronized(z1){
   z1.zobrazit(s1);
  }
}
```

Komunikace mezi vlákny

- Ovládání synchronizované metody umožní ovládat komunikaci mezi vlákny, děje se voláním metod:

`wait()` zastaví vlákno do doby pokynu metodou **`notify()`**resp. **`notifyAll()`** nebo zastaví vlákno na určenou dobu, tím se uvolní monitor pro ostatní vlákna

`notify()` probouzí pozastavené vlákno metodou **`wait()`,** není určeno které, a převzalo monitor

`notifyAll()` probouzí všechna vlákna metodou **`wait()`,** monitoru se zmocní vlákno s nejvyšší prioritou

Komunikace mezi vlákny, Box příklad

```
class Box {
    private int hodnotaVBoxu; //nepřístupná zvenčí
    boolean obsazeno = false;

    synchronized int vyjmout(){
     if(!obsazeno){ // box je prázdný
       try{
          wait(); // čekej dokud ho Prodavač nenaplní
       } catch (InterruptedException v){
    System.out.println("Vyjmout: InterruptedException");
       }
    }
    obsazeno = false; // obsah boxu vyprázdněn
    System.out.println("Vyjmout:" + hodnotaVBoxu);
    notify(); // vzbuď Prodavače, může naplnit box
    return hodnotaVBoxu;
}
```

Komunikace mezi vlákny, Box příklad

```
synchronized void vlozit(int hodnotaVBoxu){
  if(obsazeno){ // box je plný
    try{
      wait(); // čekej dokud ho Zákazník nevyprázdní
    } catch (InterruptedException v){
     System.out.println("Vlozit: InterruptedException");
    }
  }

  this.hodnotaVBoxu = hodnotaVBoxu;
  obsazeno = true; // box naplněn
  System.out.println("Vlozit:" + hodnotaVBoxu);
  notify(); // vzbuď Zákazníka, může vybrat obsah boxu
  }
}
```

Komunikace mezi vlákny, příklad fronty

```
class Prodavac implements Runnable {
    Box b;
    Prodavac(Box b){
        this.b=b;
    }
    public void run(){
        for(int i=0; i<5; i++){
            b.vlozit(i);
        }
    }
}

class Zakaznik implements Runnable {
    Box b;
    Zakaznik(Box b){
        this.b=b;
    }
    public void run(){
        for(int i=0; i<5; i++){
            b.vyjmout();
        }
    }
}
```

Komunikace mezi vlákny, příklad fronty

```
public class DemoVlakno3 {
    public static void main(String[] args) {
        Box box = new Box();
        Prodavac p = new Prodavac(box);
        Thread tProdavac = new Thread(p, "Prodavac");
        tProdavac.start();

        Zakaznik z =  new Zakaznik(box);
        Thread tZakaznik = new Thread(z, "Zakaznik");
        tZakaznik.start();
    }
}
```

Dvě vlákna (tProdavac, tZakaznik) společně pracují s jedním objektem (box). V jednom okamžiku může být do boxu buď vkládáno (Prodavač) nebo z boxu vybíráno (Zákazník).

Komunikace mezi vlákny, příklad fronty

Prodavac – vkládá do boxu (vložit)

obsazeno == false

1) **„vložit do boxu“**
2) **obsazeno = true**
3) **notify()** zakazníka
4) **wait()**

obsazeno == true

1) **wait()**

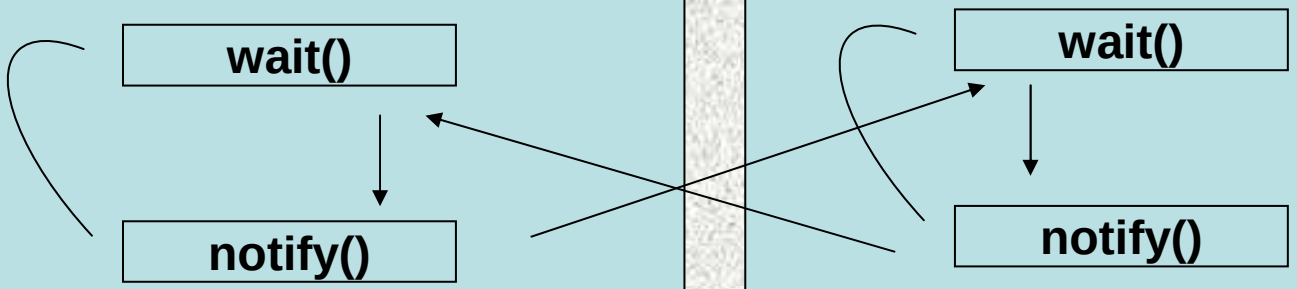
Zakaznik – odebírá z boxu(vyjmout)

obsazeno == true

1) **„vyjmout z boxu“**
2) **obsazeno = false**
3) **notify()** prodavače
4) **wait()**

obsazeno == false

1) **wait()**



Synchronizace vláken, netrpělivé čekání

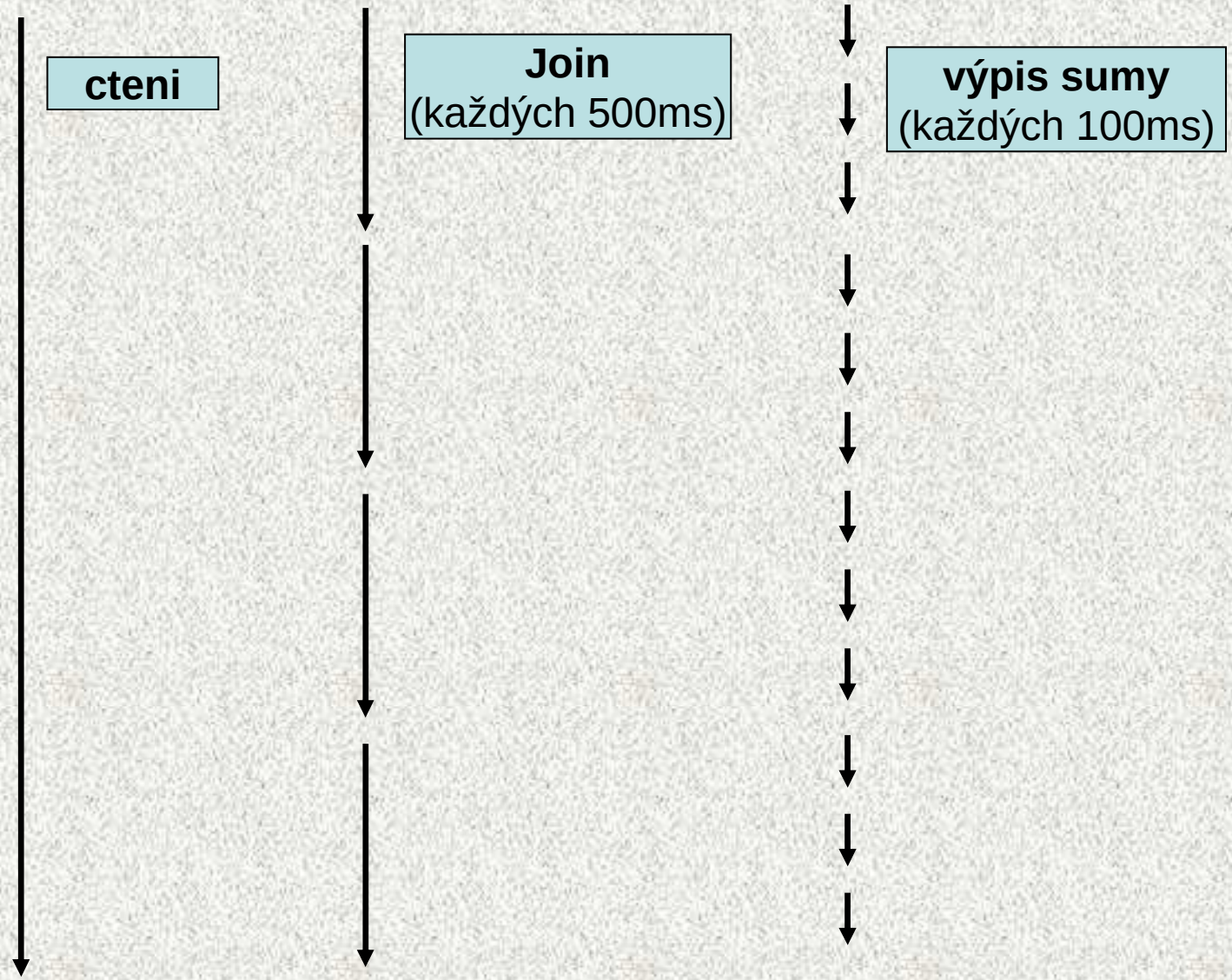
```
public class Vlakno9 extends Thread {
  public void run() {
    while (ReadVl.hotovo == false) {
      System.out.println(ReadVl.suma);
      try {
        Thread.sleep(100);  // 100 milisekund
      }
      catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
    System.out.println(ReadVl.suma);
  }
```

Synchronizace vláken, netrpělivé čekání

```
public static void main(String[] argv) throws
 InterruptedException {
   int sance = 500;
   long zac = System.currentTimeMillis();
   ReadVl vlCteni = new ReadVl("data.txt");
   vlCteni.start();
   Vlakno9 vlVypis = new Vlakno9();
   vlVypis.start();
   while (vlVypis.isAlive() == true) {
     vlVypis.join(sance);   // dá šanci, pak skončí
     System.out.println("\t Doba šance = " + sance);
   }

   long kon = System.currentTimeMillis();
   System.out.println("Konec: " + (kon - zac));
 }
}
```

Synchronizace vláken, netrpělivé čekání



Synchronizace vláken,předčasné ukončení

```
public class Vlakno10 extends Thread {
  public void run() {
    while (ReadVl.hotovo == false) {
      System.out.print(ReadVl.suma + "\r");
      yield();
      if (interrupted() == true) {
        return;
      }
    }
    System.out.println(ReadVl.suma);
  }
```

Pro zájemce

Synchronizace vláken,předčasné ukončení

```
public static void main(String[] args) throws
                                        InterruptedException {
  long zac = System.currentTimeMillis();
  ReadVl vlCteni = new ReadVl("data.txt");
  vlCteni.start();
  Vlakno10 vlVypis = new Vlakno10();
  vlVypis.start();
  vlVypis.join(1000);
  if (vlVypis.isAlive() == true) {
    System.out.println("\t Vyprsel ti cas - koncis!");
    vlVypis.interrupt();
  }
  long kon = System.currentTimeMillis();
  System.out.println("Konec: " + (kon - zac));
 }
}
```

Pro zájemce

Synchronizace vláken, předčasné probuzení

```
public class Vlakno11 extends Thread {
  public void run() {
    while (ReadVl.hotovo == false) {
      System.out.print(ReadVl.suma + "\r");
      try {
        Thread.sleep(7000);  // 100 milisekund
      }
      catch (InterruptedException e) {
        System.out.println("Predcasne vzbuzen");
        break;
      }
    }
    System.out.println(ReadVl.suma);
  }
```

Pro zájemce

Synchronizace vláken, předčasné probuzení

```
public static void main(String[] argv) throws
 InterruptedException {
   long zac = System.currentTimeMillis();
   ReadVl vlCteni = new ReadVl("data.txt");
   vlCteni.start();
   Vlakno11 vlVypis = new Vlakno11();
   vlVypis.start();
   vlVypis.join(1000);
   if (vlVypis.isAlive() == true) {
     System.out.println("\t Vyprsel ti cas - koncis!");
     vlVypis.interrupt();
   }

   long kon = System.currentTimeMillis();
   System.out.println("Konec: " + (kon - zac));
  }
}
```

Pro zájemce

Pozastavení a obnovení činnosti vláken

```
class MojeVlakno implements Runnable {
    boolean zastaven;

    MojeVlakno() {
        zastaven = false;
    }

    public void run() {
        try {
            for (int i = 0; i < 10; i++) {
                System.out.println("Vlakno" + i);
                Thread.sleep(2000);
                synchronized (this) {
                    while (zastaven) {
                        wait();
                    }
                }
            }
        } catch (InterruptedException e) {
            System.out.println("Vlakno probuzeno");
        }
        System.out.println("Vlakno ukonceno");
    }
```

Pro zájemce

Pozastavení a obnovení činnosti vláken

```
    void zastavitChod() {
        zastaven = true;
    }

    synchronized void obnovitChod() {
        zastaven = false;
        notify();
    }
}
```

Pro zájemce

Pozastavení a obnovení činnosti vláken

```
public class DemoVlakno4 {
    public static void main(String[] args) {
        MojeVlakno mv1 = new MojeVlakno();
        Thread t1 = new Thread(mv1, "Moje vlákno");
        t1.start();
        try {
            Thread.sleep(1000); t1.zastavitChod();
            System.out.println("Vlakno, chod zastaven");
            Thread.sleep(1000); t1.obnovitChod();
            System.out.println("Vlakno, chod obnoven");
            Thread.sleep(2000); t1.zastavitChod();
            System.out.println("Vlakno, chod zastaven");
            Thread.sleep(4500); t1.obnovitChod();
            System.out.println("Vlakno, chod obnoven");
        } catch (InterruptedException v) {}
        try {
            t1.join();
        } catch (InterruptedException v) {
            System.out.println("Hlavni vlankno preruseno");
        }}}
```

Pro zájemce