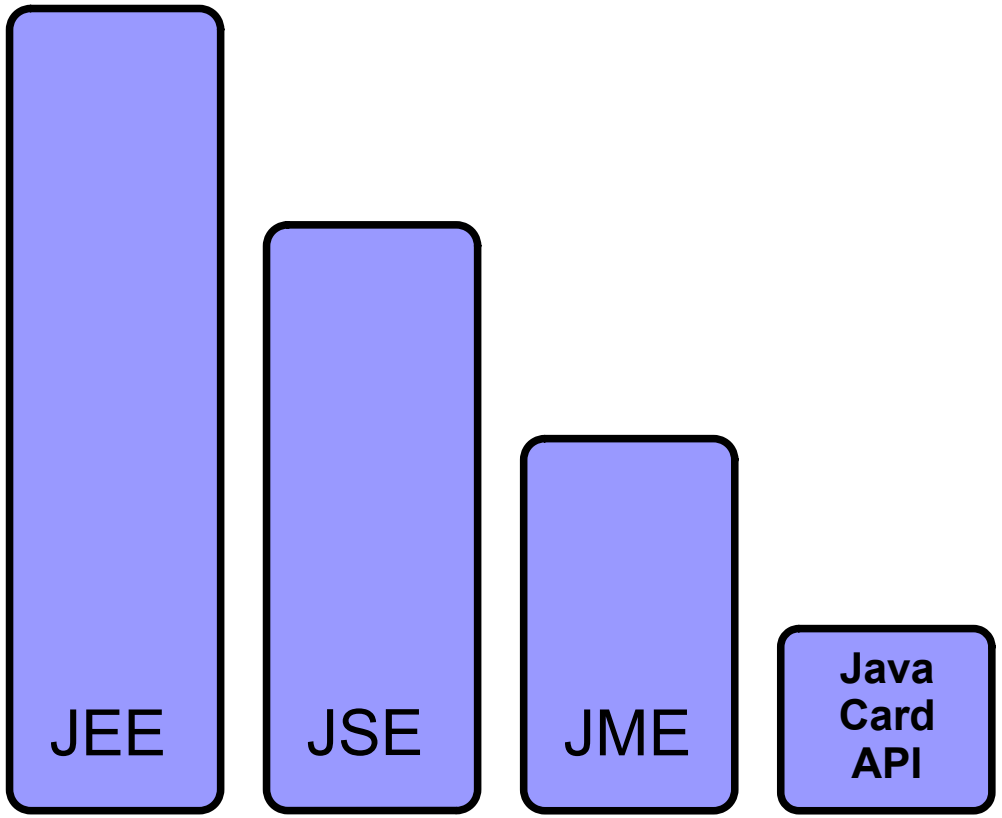
Java Micro Edition

Přehled

- předchůdci
  - Personal Java (1997)
  - Embedded Java (1998)
- definice JME – přes JCP
  - JCP – Java Community Process
- JME není jeden balík SW
  - sada technologií a specifikací
  - definuje
    - konfigurace (*configuration*)
    - profily (*profiles*)
    - volitelné balíky (*optional packages*)

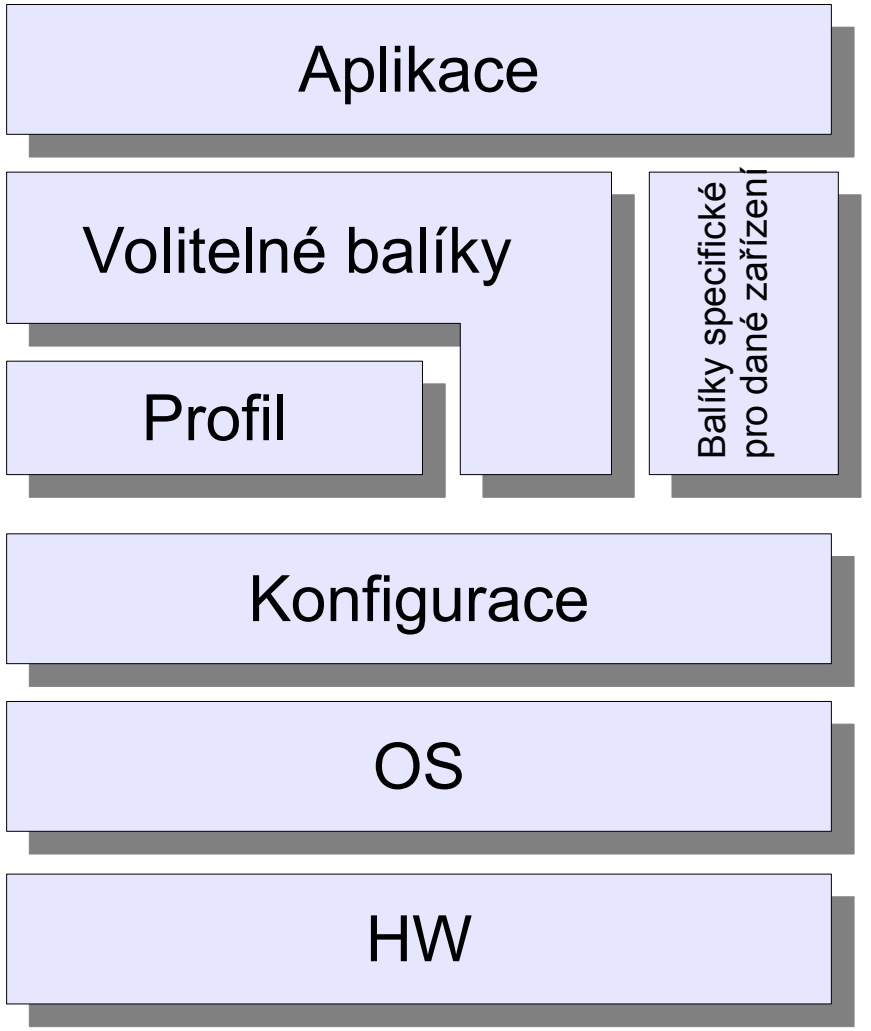
Java platform

- JSE – standard edition
- JEE – enterprise edition
- JME – micro edition



Architektura

- více vrstev
- **konfigurace**
  - specifikace VM
  - základní API
  - požadavky na zařízení (paměť, CPU,...)
- **profil**
  - API pro vytváření aplikací (pro specifická zařízení – m. telefon, PDA,...)
  - životní cyklus aplikace, GUI,...
- **volitelné balíky**
  - API pro specializované služby



Software

- Java ME SDK
  - http://www.oracle.com/technetwork/java/javame/

Přehled technologií

- JSR 30 – CLDC 1.0 – Connected, Limited Device Configuration
- JSR 139 – CLDC 1.1 – Connected, Limited Device Configuration 1.1
- JSR 36 – CDC – Connected Device Configuration
- JSR 218 – CDC 1.1 – Connected Device Configuration 1.1

- JSR 37 – MIDP 1.0 – Mobile Information Device Profile
- JSR 118 – MIDP 2.0 – Mobile Information Device Profile 2.0
- JSR 271 – MIDP 3.0 – Mobile Information Device Profile 3.0
- JSR 46 – FP – Foundation Profile
- JSR 129 – PBP – Personal Basis Profile
- JSR 62 – PP – Personal Profile

- JSR 82 – BTAPI – Java APIs for Bluetooth
- JSR 120 – WMA – Wireless Messaging API
- ...

Konfigurace

- základní specifikace
- určena pro širokou škálu zařízení s podobnými vlastnostmi
- definuje
  - požadavky na CPU, MEM, připojení k síti
  - vlastnosti VM
  - základní API (odvozené od JSE)
- konfigurace
  - CLDC – Connected, Limited Device Configuration
    - mobilní telefony, PDA,...
  - CDC – Connected Device Configuration
    - PDA, navigační systémy, set-top boxy,...

Profily

- nad konfigurací
- přidává API pro tvorbu aplikací
  - definuje
    - životní cyklus aplikace
    - API pro tvorbu GUI
    - persitence dat
    - …
- nad CDLC
  - MIDP – Mobile Information Device Profile
- nad CDC
  - Foundation Profile
  - Personal Profile

CLDC 1.0

- nejmenší konfigurace
- pro malá zařízení s omezenými zdroji
- požadavky na HW
  - 16-bit nebo 32-bit procesor
  - 128 kB trvalé paměti, 32 kB operační paměti
  - zdroj – baterie
  - pomalé připojení k síti
- omezená VM
  - KVM (Kilo VM)

CLDC 1.0 – KVM

- žádné floating-point operace a typy
- není finalizace objektů
- omezená sada výjimek
- není
  - JNI
  - reflexe
  - uživatelsky definované classloadery
  - daemon vlákna a skupiny vláken
  - weak reference
- bezpečnostní model – *sandbox*
- dvě fáze verifikace kódu

CLDC 1.0 – KVM – verifikace

- normální verifikace byte-kódu – náročná na zdroje
  - velikost 50 kB, operační paměť až 100 kB
  - náročné na výkon CPU
- rozdělení na 2 části
  - předverifikace
    - probíhá po překladu
      - typicky ji provádí vývojář
    - ke každé třídě přidán StackMap atribut
    - odstraněny některé instrukce (skoky) a nahrazeny ekvivalentními
    - asi 5% zvětšení kódu třídy
  - verifikace
    - pouze lineární analýza kódu
    - rychlé, nenáročné
      - velikost verifikátoru ~ 10 kB, operační paměť < 100 B

CLDC 1.0 – API

- java.lang
  - Object, Class, Runtime, System, Thread, Runnable, String, StringBuffer, Throwable
  - Boolean, Byte, Short, Integer, Long, Character
  - Math
- java.util
  - Vector, Stack, Hashtable, Enumeration
  - Date, Calendar, TimeZone
  - Random
- java.io
  - InputStream, OutputStream, ByteArrayInputStream, ByteArrayOutputStream, DataInput, DataOutput, DataInputStream, DataOutputStream, Reader, Writer, InputStreamReader, OutputStreamWriter, PrintStream

CLDC 1.0 – API

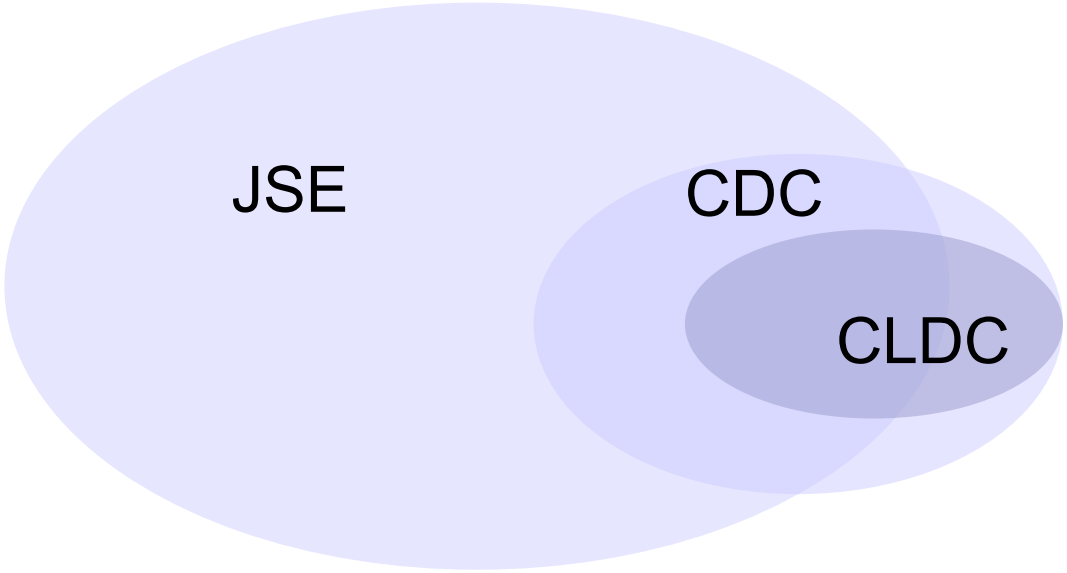
- Generic Connection Framework
  - javax.microedition.io
  - streamy
  - jednotná abstrakce pro různé druhy připojení
  - Connector.open("<protocol>:<address>;<parameters>")
  - př.:
    - Connector.open("http://www.foo.com");
    - Connector.open("socket://129.144.111.222:9000");
    - Connector.open("comm:0;baudrate=9600");
    - Connector.open("datagram://129.144.111.333");
    - Connector.open("file:/foo.dat");
  - na úrovni konfigurace není žádná implementace

CLDC 1.1

- podpora floating-point operací
- weak references
- vylepšeny třídy Date, Calendar, TimeZone
- vlákna mají jména
- minimální požadovaná paměť 192 kB

CDC

- 32-bit procesor, 2 MB RAM, 2.5 MB ROM
- VM – plné možnosti JSE VM
- CDC je nadmnožina CLDC
- java.io, java.util.zip, java.util.jar, java.net, java.security



$CLDC \subseteq CDC$

CDC profily

- Foundation Profile
  - základní profil
  - žádné GUI
  - práce s textem, HTTP, sockety
  - java.math
  - java.util.zip, java.util.jar
  - certifikáty, šifrování
- Personal Basis Profile
  - nad FP, podmnožina PP
  - část AWT, podpora JavaBeans
  - aplikace – Xlet
  - RMI komunikace
- Personal Profile
  - podobné JSE
  - kompletní AWT

MIDP

- Mobile Information Device Profile
- nad CLDC
- je v mobilních telefonech
- HW požadavky (MIDP 1.0)
  - display min. 96x54x1
  - aspect ratio 1:1
  - klávesnice nebo touch screen
  - 128 kB permanentní paměti
  - 8 kB permanentní paměti pro data aplikací
  - 32 kB operační paměti
  - obousměrné připojení k síti
- HW požadavky (MIDP 2.0)
  - 256 kB permanentní paměti
  - 128 kB operační paměti
  - zvuk

MIDP 1.0

- aplikace – MIDlet
- podpora pro GUI
- podpora pro komunikaci po síti (GCF)
  - HTTP
- ukládání dat aplikací
  - Record Management Storage (RMS)
- over the air (OTA)
  - způsob, jak uložit aplikaci do telefonu
- balíky
  - javax.microedition.midlet
  - javax.microedition.lcdgui
  - javax.microedition.rms

MIDP 2.0

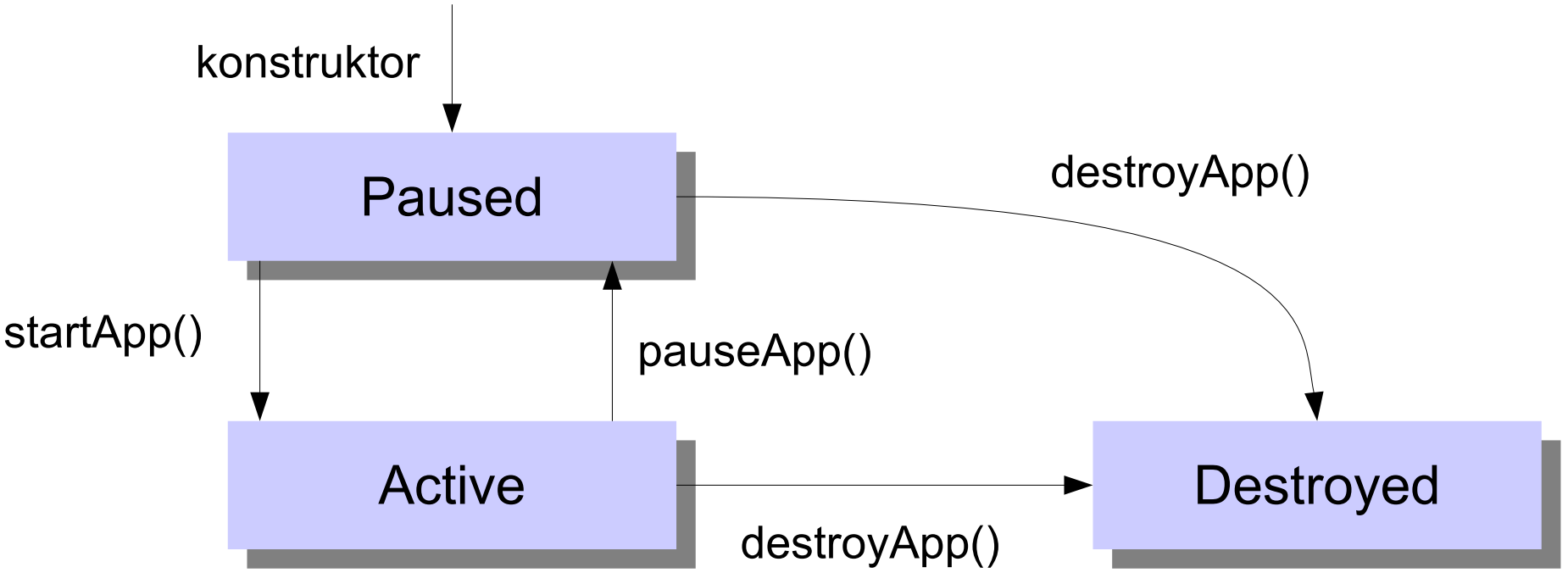
- lepší podpora sítí
  - HTTPS, TCP a UDP sockety
- podpora multimedií
  - Mobile Media API (MMAPI)
- podpora pro vytváření her
  - GameCanvas, Layers, Sprites
- certifikáty,...
- vylepšené GUI
- push registry
  - spouštění MIDletů na základě příchozích spojení
- úložiště lze sdílet mezi více aplikacemi

MIDP 3.0

- JSR 271
  - prosinec 2009
- běh více MIDletů současně a jejich komunikace
- podpora IPv6
- LIBlets
  - sdílené knihovny

MIDlet

- aplikace pro MIDP
- obdoba appletu
- potomci od javax.microedition.midlet.MIDlet
- životní cyklus aplikace



Metody MIDletu

- startApp()
  - volá se při přechodu do *ACTIVE* stavu
  - předefinovávaná programátorem
- pauseApp()
  - volá se při přechodu do *PAUSED* stavu
  - předefinovávaná programátorem
- destroyApp(boolean unconditional)
  - volá se při přechodu do *DESTROYED* stavu
  - pokud je parametr *false*, midlet může odmítnout skončit
  - předefinovávaná programátorem
- notifyDestroyed()
  - zavoláním se midlet ukončí (destroyApp se nevolá)

Metody MIDletu (pokrač.)

- notifyPaused()
  - zavoláním midlet chce přejít do stavu *PAUSED*
  - metoda pauseApp se nevolá
    - obdoba jak notifyDestroyed
- resumeRequest()
  - opak notifyPaused
  - midlet chce ze stavu *PAUSED* do *ACTIVE*
  - lze volat např. z časovače nebo z vlákna na pozadí

MIDlet – implementace

```
public class Main extends MIDlet {
  public Main() {
  }

  public void startApp() {
    Displayable current = Display.getDisplay(this).getCurrent();
    if (current == null) {
      HelloScreen helloScreen = new HelloScreen(this);
      Display.getDisplay(this).setCurrent(helloScreen);
    }
  }

  public void pauseApp() {  }

  public void destroyApp(boolean b) {  }

  void exitRequested() {
    destroyApp(false);
    notifyDestroyed();
  }
}
```

UI MIDletu

- zobrazení pouze jednoho okna v jednu chvíli
  - více oken – přepínání

```
Display.getDisplay(this).setCurrent(helloScreen);
```

- běží-li současně více MIDletů, pouze jeden z nich má přístup k displej

Distribuce midletů

- 2 soubory
  - JAR archiv – kód aplikace
  - JAD – Java Archive Descriptor
    - formát
      - jméno-atributu: hodnota-atributy
    - stejné informace musejí být i v manifestu JAR archivu
- příklad JAD

```
MIDlet-Name: HelloWorld
MIDlet-Version: 0.0.1
MIDlet-Vendor: PH
MIDlet-Jar-URL: HelloWorld.jar
MIDlet-Jar-Size: 1949
MIDlet-1: HelloWorld,,cz.cuni.mff.java.helloworld.Main
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
```

Distribuce midletů (pokrač.)

- v jednom balíku lze mít více midletů

  MIDlet-1: HelloWorld,,cz.cuni.mff.java.helloworld.Main
  MIDlet-2: HelloWorld2,,cz.cuni.mff.java.helloworld.Main2
  MIDlet-3: HelloWorld3,,cz.cuni.mff.java.helloworld.Main3
- do deskriptoru lze dát uživatelské atributy
  - lze je získat z aplikace
    - MIDlet.getAppProperty(String key)

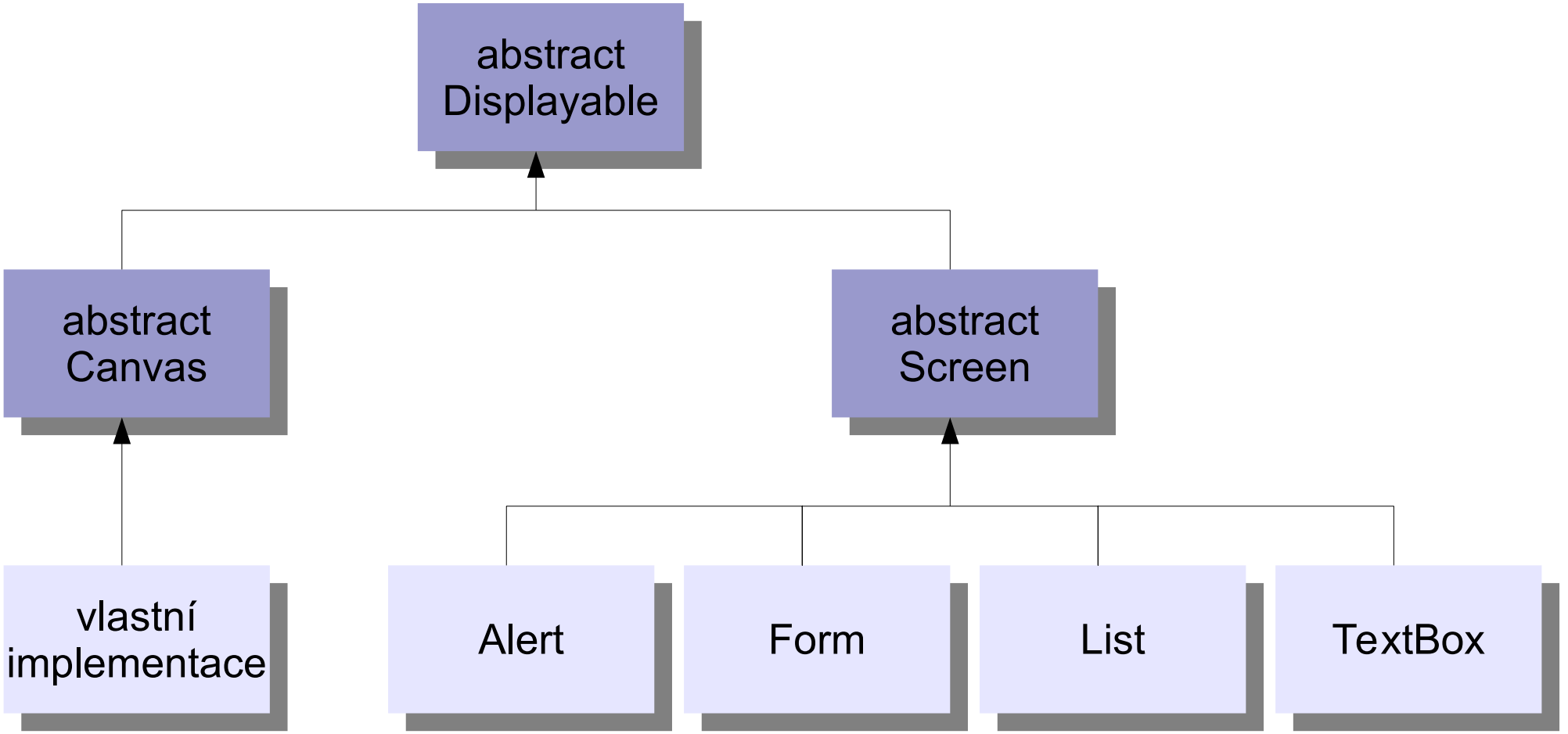
Record Management Store

- ukládání polí bytů
  - není to filesystem
- pro každý midlet vlastní úložiště
  - MIDP 2.0 – úložiště lze sdílet
- operace jsou atomické
- uložená data jsou perzistentní
- při vymazání midletu ze zařízení se smažou i záznamy
- balík javax.microedition.rms
  - třída RecordStore
    - openRecordStore()
    - addRecord()
    - getRecord()

GUI

- balík javax.microedition.lcdui
- nízkoúrovňové
  - Canvas
    - kreslení na display
    - obsluha stisků kláves/dotyků
- vysokoúrovňové
  - nezávislé na typu zařízení
  - nelze ovlivnit nízkoúrovňové vlastnosti
    - fonty, atd.
  - přenositelné

GUI


abstract
Displayable
abstract
Canvas
abstract
Screen
vlastní
implementace
Alert
Form
List
TextBox

GUI – MIDP 2.0

- javax.microedition.lcdui.game
  - GameCanvas
    - dědí od Canvas
    - umožňuje
      - dotazovat se na stav kláves
      - off-screen bufer
  - Layer
    - abstraktní třída pro viditelné elementy hry
    - potomci
      - Sprite
      - TiledLayer
  - LayerManager
    - správce viditelných elementů

GUI – MIDP 2.0

- javax.microedition.media
  - přehrávání multimédií
  - třída Manager
    - statické metody
    - void **playTone**(int note, int duration, int volume)
    - String[] **getSupportedContentTypes**(String protocol)
    - String[] **getSupportedProtocols**(String content_type)
    - Player **createPlayer**(String locator)
    - Player **createPlayer**(InputStream stream, String type)

Volitelné balíky

- rozšiřují profily
- definovány na základě JCP
- zvlášť pro CLDC nebo CDC (nebo pro oba)

- Wireless Messaging API (WMA) JSR 120, JSR 205
- JME Web Services APIs (WSA) JSR 172
- Bluetooth API JSR-82

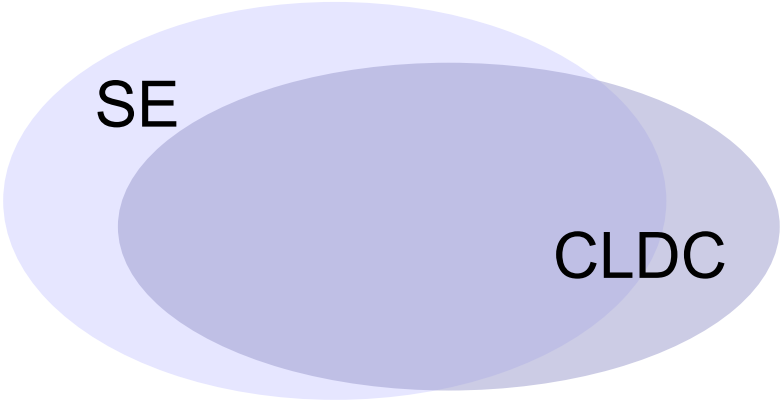
- JME RMI Optional Package (RMI OP) JSR 66
- JDBC Optional Package for CDC/Foundation Profile API JSR 169

Java ME 8

- 2014
- snaha o sjednoceni ME a SE
- CLDC 8
- MEEP 8
  - ME Embedded Profile 8

CLDC 8

- CLDC 8 – „extended strict subset of SE 8“



- VM odpovídá
  Java VM specification pro SE 7
  - bez
    - InvokeDynamic instrukce
    - reflexe a runtime annotací
- jazyk „skoro“ jako Java 8
  - bez
    - lambda funkcí
    - reflexe
    - serializace
    - JNI
    - vlastních classloaderů
    - ...

CLDC 8

- verifikace
  - verze bytekódu 51+ (JDK 7+)
    - bez předverifikace
  - verze bytekódu 48 a starší (JDK 1.4)
    - nutná předverifikace
- vylepšený Generic Connection Framework
  - podpora více protokolů
  - IP multicast
  - více možností nastavování parametrů protocolů
    - ConnectionOption
  - vyhledávání „access pointů“
    - 3GPP, CDMA, Wi-Fi,...
- podpora ServiceLoaderů

MEEP 8

- Java ME Embedded Profile (MEEP) 8
- nad CLDC 8
- profily
  - minimal
    - základní API, aplikační model
    - minimum – 128 kB RAM & 1 MB Flash
  - standard
    - služby, multitasking,...
    - minimum – 512 kB RAM & 2 MB Flash
  - full
    - kompletní API
    - minimum – 2 MB RAM & 4 MB Flash

MEEP 8

- balíčky
  - povinné
    - javax.microedition.midlet
  - volitelné
    - javax.microedition.swm
    - javax.microedition.cellular
    - javax.microedition.event
    - javax.microedition.power
    - javax.microedition.io
    - javax.microedition.lui
    - javax.microedition.key
    - javax.microedition.media
    - javax.microedition.rms

MEEP 8

- aplikace
  - MIDlets (IMlets), LIBlets
  - javax.microedition.midlet.MIDlet
    - notifyPaused(), pauseApp(), resumeRequest() deprecated
- služby
  - ServiceLoader
  - „poskytoval“ a „konzument“ služby můžou být různých aplikacích

MEEP 8

- Device I/O API
  - přístup k různým zařízením
  - GPIO, I2C, SPI, UART,...

Java Embedded

- kompletní Java platforma
- několik variant
  - Java ME Embedded
  - Java ME Embedded Client
  - ...

Java ME Embedded

- založeno na MEEP a CLDC
- určeno pro micro-kontrolery apod.
- headless
  - žádné UI
- plaftormy
  - ARM
    - Raspberry Pi
  - STM32
  - ...
- < 1 MB RAM

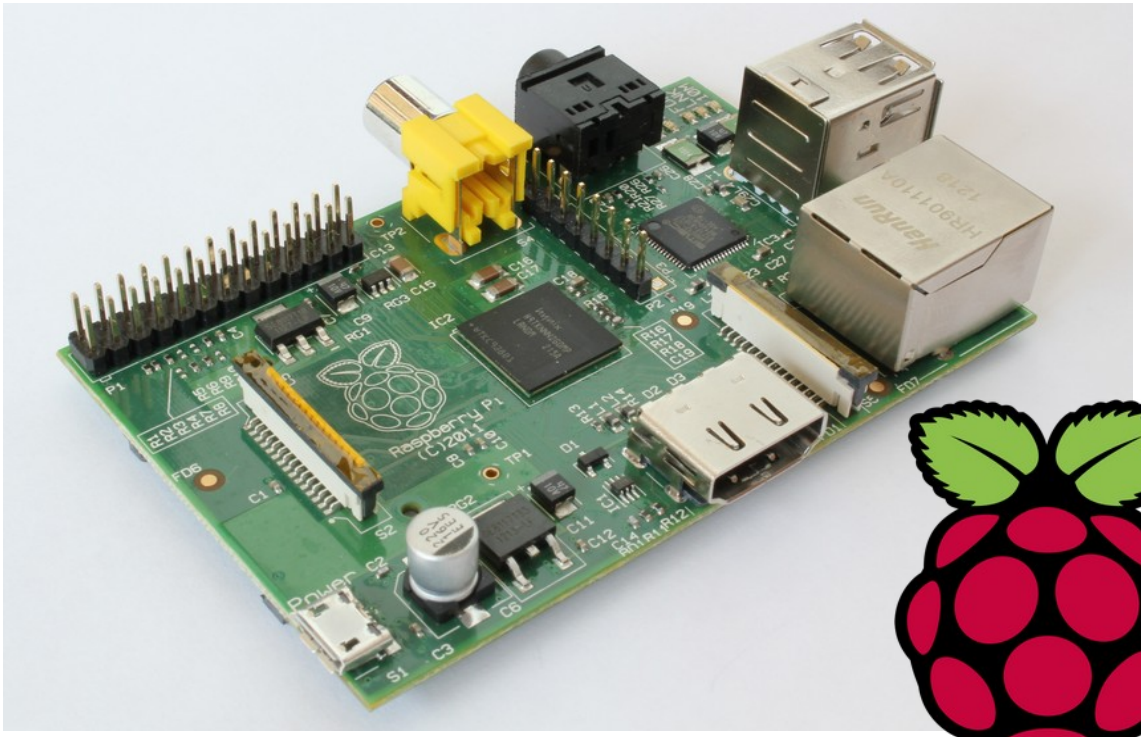
Java ME Embedded Client

- založeno na JME a CDC
- < 10 MB RAM

Pi4J

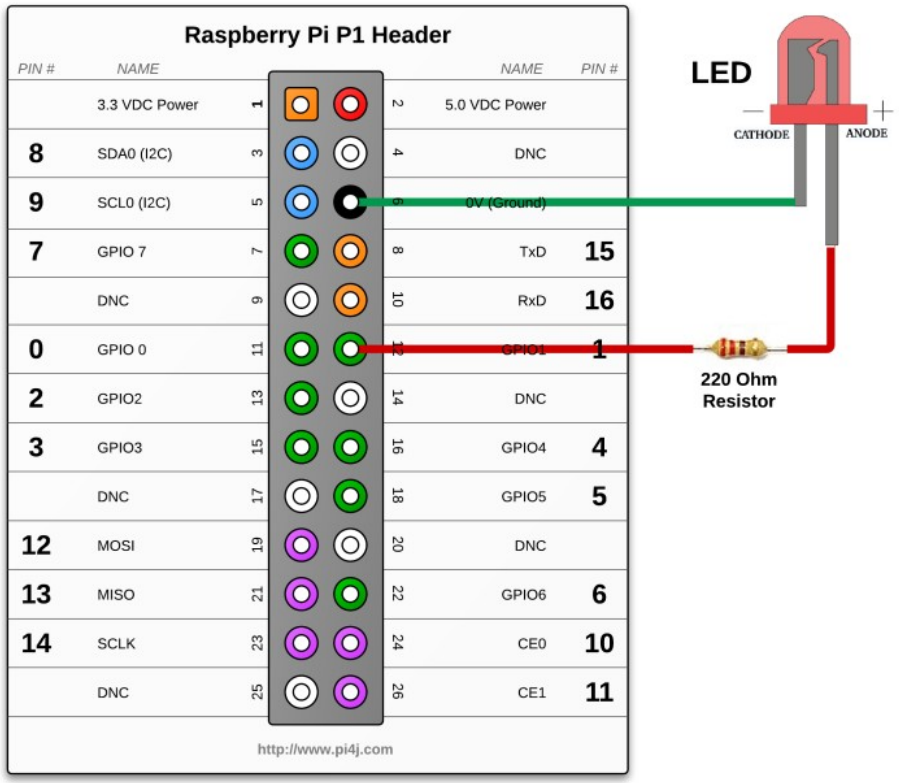
Pi4J

- http://pi4j.com/
- Raspberry Pi
- pro JSE
- GPIO, UART



Pi4J: příklad

```
final GpioController gpio = GpioFactory.getInstance();

final GpioPinDigitalOutput pin =
     gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01,
        "MyLED", PinState.HIGH);
pin.setShutdownOptions(true, PinState.LOW);

Thread.sleep(5000);

pin.low();

Thread.sleep(5000);

pin.pulse(1000, true);

gpio.shutdown();
```



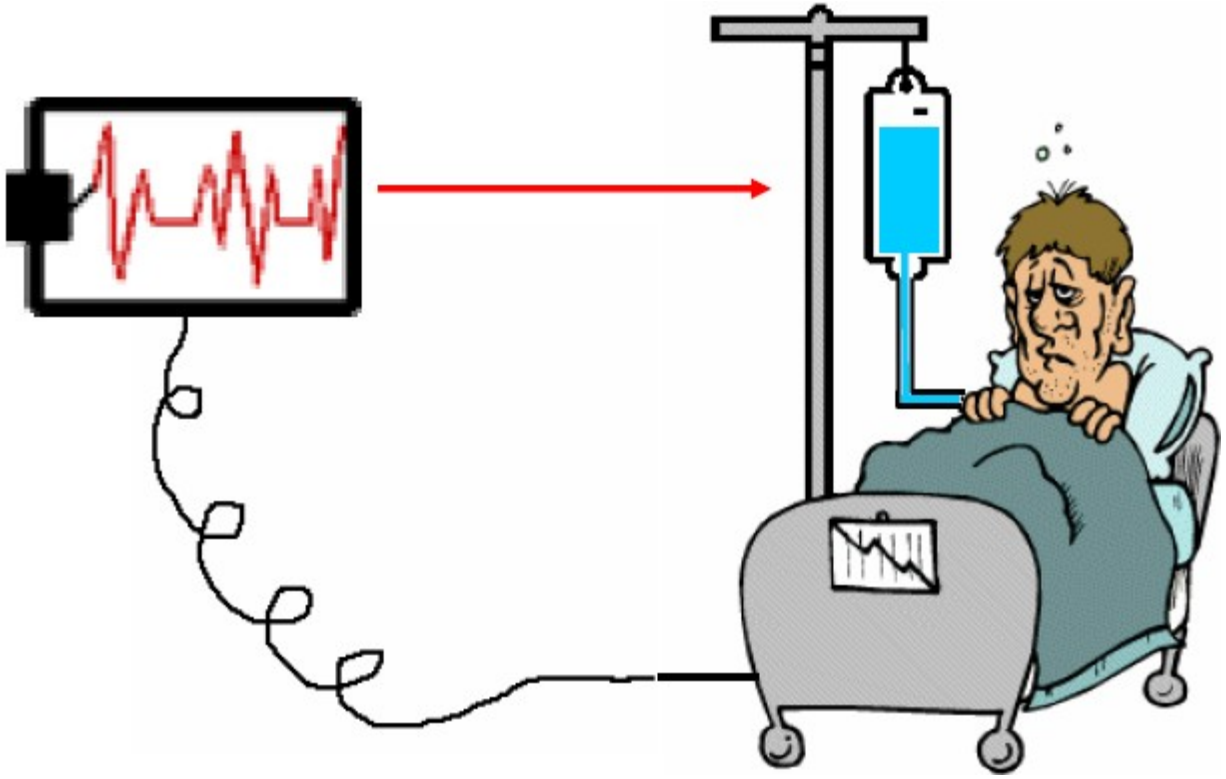
Real-Time Java

Real-time systém

- česky někdy jako „systém reálného času“
- ne-real-time systém
  - systém funguje správně pokud dává správné výsledky
- real-time systém
  - systém funguje správně pokud dává správné výsledky v požadovaném čase

Real-time systém

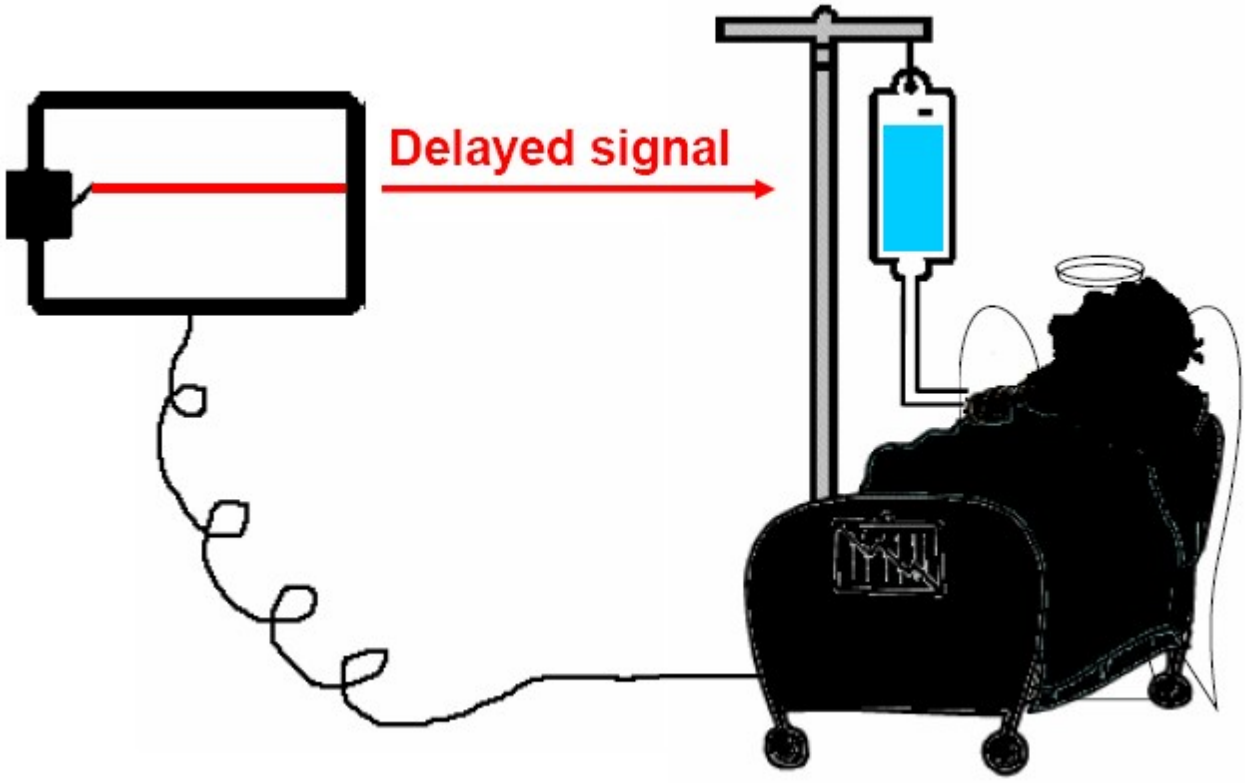
- příklad
  - lékařské zařízení musí detekovat změny v pacientově stavu a včas reagovat



zdroj obrázku Issovic, D.:Real-time systems, basic course

Real-time systém

- jinak...

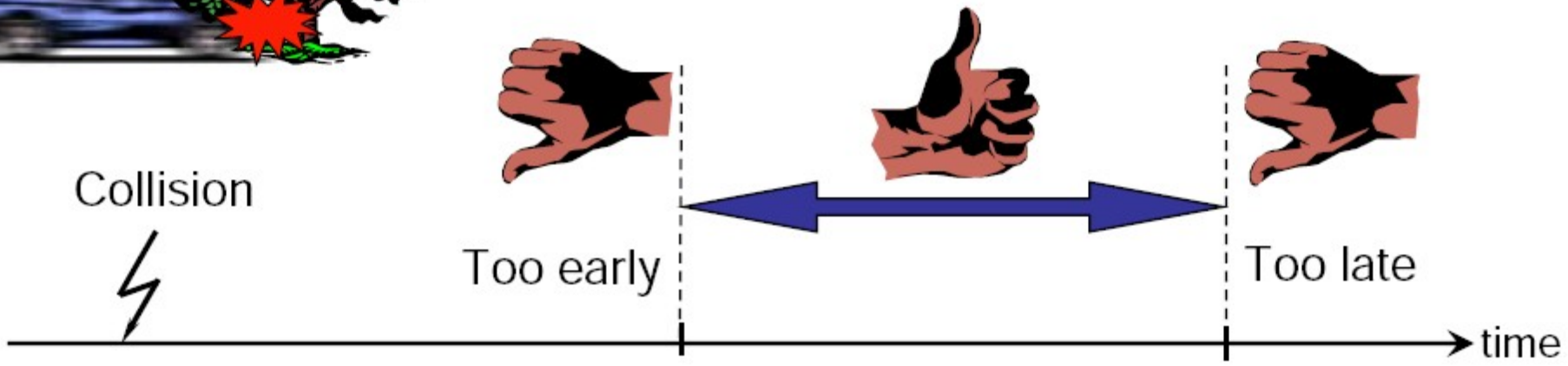


zdroj obrázku Issovic, D.:Real-time systems, basic course

Real-time systém

- příklad
  - airbag se nesmí nafouknout ani příliš brzo ani příliš pozdě

real-time ≠ rychle



zdroj obrázku Issovic, D.:Real-time systems, basic course

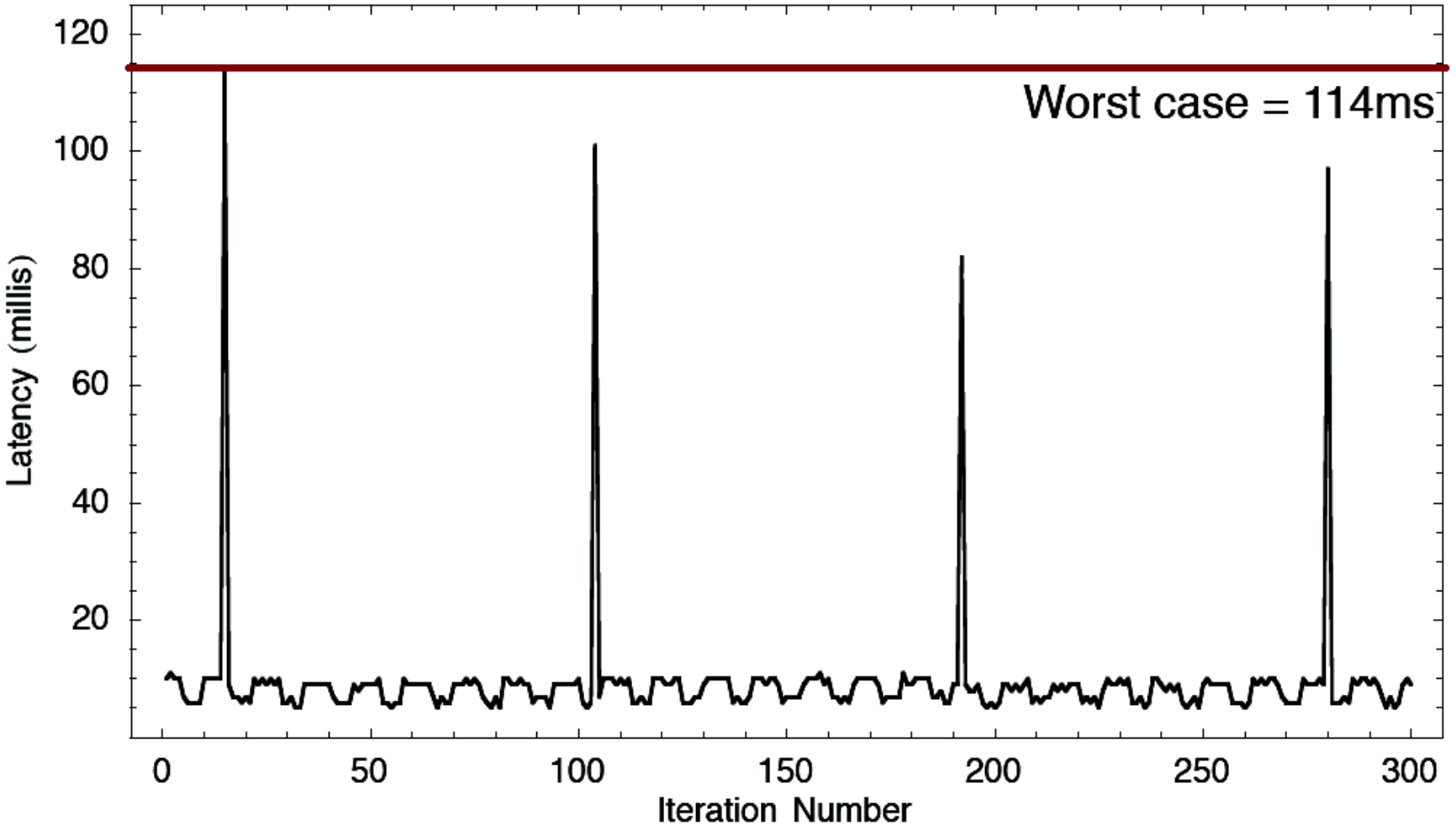
Real-time systém

- soft real-time
- hard real-time
- safety-critical

Java a RT

- Java
  - jednoduchá
  - široce používaná
  - množství knihoven
  - portabilní
- ale
  - není real-time plánování
  - není podpora pro periodické vykonávání
  - není podpora pro aperiodické události
  - problémy s GC
  - problémy s přímým přístupem do paměti
  - problémy s ovládáním zařízení
  - ...

Garbage collector



Real-time Specification for Java

- RTSJ
- 1999 – JSR-1
- žádné změny v syntaxi
- rozšíření Javy o
  - Thread Scheduling and Dispatching
  - Memory Management
  - Synchronization and Resource Sharing
  - Asynchronous Event Handling
  - Asynchronous Transfer of Control and Asynchronous Thread Termination
  - High resolution time
  - Physical and Raw Memory Access

RTSJ – plánování

- Fixed-priority round robin plánovač
  - lze přidat vlastní
- Minimálně 28 real-time priorit navíc k 10 základním

- Periodická vlákna
  - mohou začít v určeném čase
  - mají periodu a deadline

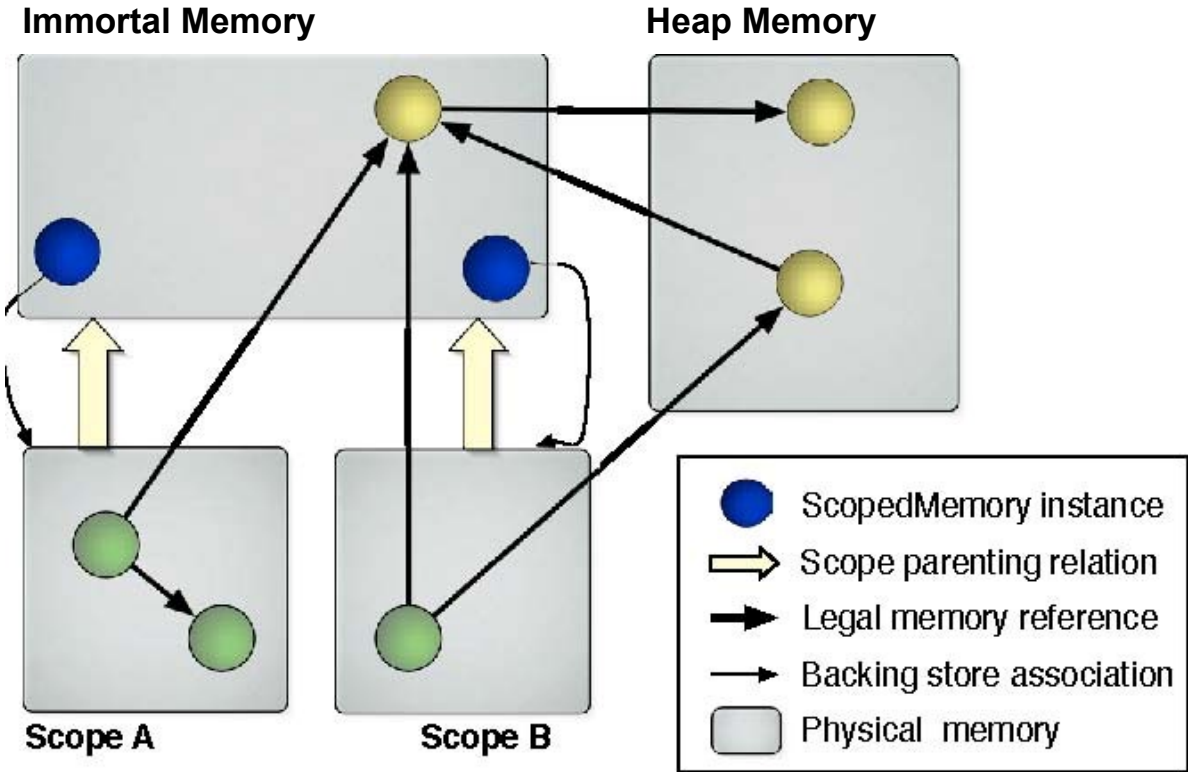
- Aperiodické události
  - plánovatelný objekt, která je vykonán jako reakce na událost

RTSJ – paměť

- NoHeapRealtimeThread
  - vlákno bez přístupu k haldě
  - nemůže být blokováno GC
- halda
  - jako normálně
- immortal paměť
  - nelze z ní objekty uvolňovat
  - pro globální data
- scoped paměť
  - regiony paměti
  - objekty uvolněny naráz pokud všechna vlákna opustí region
  - vhodné pro volání metod z std knihovny

RTSJ – paměť

- pravidla pro reference mezi objekty



RTSJ

- problémy
  - paměťové regiony jsou neintuitivní
  - změny v klasickém programovacím modelu s GC
  - přiřazení reference může selhat

- existují real-time garbage collectory

Ravenscar Java

- omezení RTJS
- inspirováno „Ravenscar for Ada“
- cíl
  - lepší analyzovatelnost a predikovatelnost

- příklad omezení
  - žádný GC

RTSJ

- RTSJ 2.0 – JSR 282
  - draft

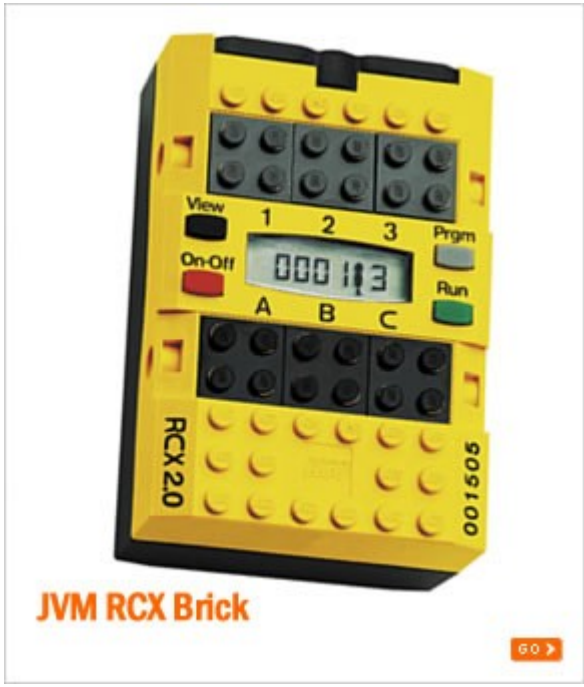
- Base Module
  - Schedulables
  - Events & Handlers
  - Priority Inheritance
  - Clock
  - MemoryArea
    - HeapMemory
    - ImmortalMemory
  - ...
- Device
  - Happenings
  - RawMemory
  - ISR (Option)
- Alternate Memory
  - physical
  - scoped
- POSIX
  - POSIX signals

JAVA

LeJOS

Přehled

- http://www.lejos.org/
- firmware pro LEGO Mindstorm
- obsahuje Java virtual machine
  tj. LEGO roboty lze programovat v Javě



Příklad

```
public static void main(String[] argv) {
  TouchSensor touchL = new TouchSensor(SensorPort.S4);
  TouchSensor touchR = new TouchSensor(SensorPort.S1);
  UltrasonicSensor sonar = new UltrasonicSensor(SensorPort.S2);

  Motor.A.forward();
  Motor.C.forward();
  LCD.drawString("Press ESC to quit", 0, 0);
  while (true) {
    if (Button.ESCAPE.isPressed()) { System.exit(0);   }
    if (touchL.isPressed() || touchR.isPressed() || (sonar.getDistance() <
                                                                    40)) {
      Motor.A.stop(); Motor.C.stop();
      sleep(1000);
      Motor.A.backward(); Motor.C.backward();
      sleep(1000);
      Motor.A.forward(); Motor.C.backward();
      sleep(1000);
      Motor.A.stop(); Motor.C.stop();
      sleep(1000);
      Motor.A.forward(); Motor.C.forward();
      }
    }

  }
```

LeJOS

- mix Java SE a ME
- omezení
  - nejsou classloadery
  - malá velikost aplikací

  - po kompilaci se vytvoří binární obraz aplikace
    - nahraje se do „kostky“

    - nxjlink -v ClassWithMain -o App.nxj
    - nxjupload App.nxj