

JAVA

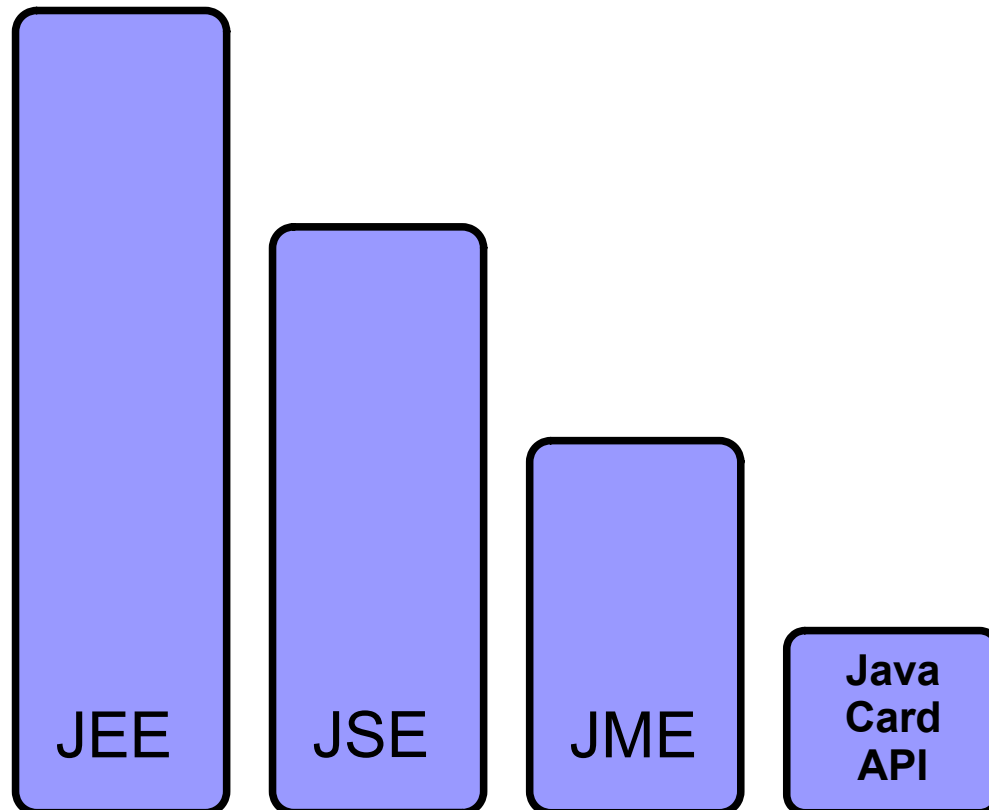
Java Micro Edition

Overview

- predecessors
 - Personal Java (1997)
 - Embedded Java (1998)
- JME definition – via JCP
 - JCP – Java Community Process
- JME is not a single SW package
 - a set of technologies and specifications
 - defines
 - *configuration*
 - *profiles*
 - *optional packages*

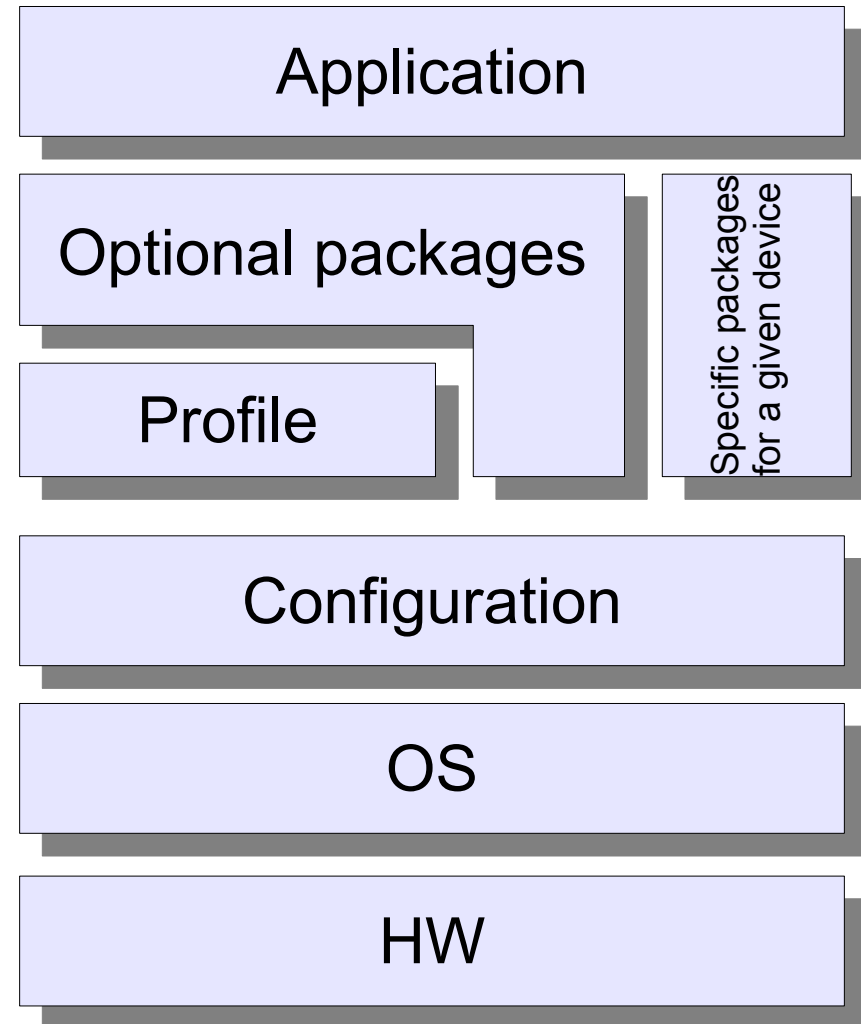
Java platform

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



Architecture

- several layers
- **configuration**
 - VM specification
 - core API
 - requirements on device (memory, CPU,...)
- **profile**
 - API for application creation (for specific devices – mob. phone, PDA,...)
 - application lifecycle, GUI,...
- **optional packages**
 - APIs for specialized services



Software

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

Technology overview

- 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
- ...

Configuration

- core specification
- intended for a large family of devices with similar features
- defines
 - requirements on CPU, MEM, net connectivity
 - features of VM
 - core API (derived from JSE)
- configurations
 - CLDC – Connected, Limited Device Configuration
 - mobile phones, PDA,...
 - CDC – Connected Device Configuration
 - PDA, navigation systems, set-top boxes,...

Profile

- over a configuration
- adds API for application creation
 - defines
 - application lifecycle
 - API for GUI
 - data persistence
 - ...
- over CDLC
 - MIDP – Mobile Information Device Profile
- over CDC
 - Foundation Profile
 - Personal Profile

CLDC 1.0

- the smallest configuration
- for small devices with limited resources
- HW requirements
 - 16-bit or 32-bit processor
 - 128 kB permanent memory, 32 kB operating memory
 - energy source – battery
 - slow connection to network
- limited VM
 - KVM (Kilo VM)

CLDC 1.0 – KVM

- no floating-point operations and types
- no object finalization
- limited set of exceptions
- no
 - JNI
 - reflection
 - user defined classloaders
 - daemon threads and thread groups
 - weak references
- security model – *sandbox*
- two phases of code verifications

CLDC 1.0 – KVM – verification

- regular byte-code verification – resource demanding
 - size 50 kB, operation memory up to 100 kB
 - CPU performance demanding
- divided to two parts
 - preverification
 - during development
 - typically performed by a developer
 - the StackMap field added to every class
 - some instructions (jumps) replaced by equivalent ones
 - size of a class increased by approx. 5%
 - verifications
 - only linear analysis
 - fast, nondemanding
 - verifier size ~ 10 kB, operating memory < 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

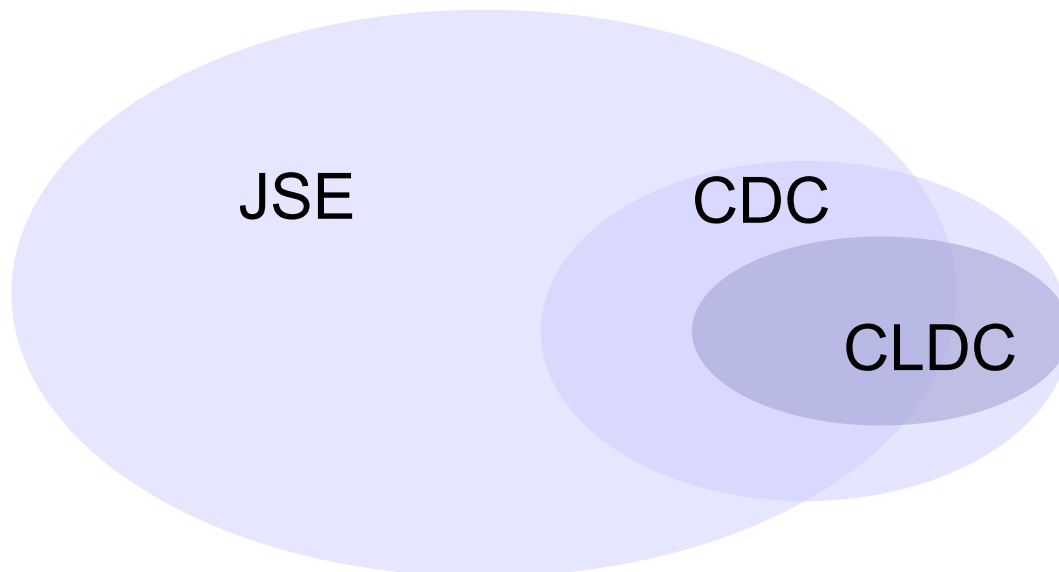
- Generic Connection Framework
 - javax.microedition.io
 - streams
 - a common abstraction for different kinds of connections
 - `Connector.open("<protocol>:<address>;<parameters>")`
 - e.g.:
 - `Connector.open("http://www.foo.com");`
 - `Connector.open("socket://129.144.111.222:9000");`
 - `Connector.open("comm:0;baudrate=9600");`
 - `Connector.open("datagram://129.144.111.333");`
 - `Connector.open("file:/foo.dat");`
 - no implementation at the configuration level

CLDC 1.1

- support of floating-point operations
- weak references
- enhanced classes Date, Calendar, TimeZone
- threads has names
- minimal required memory 192 kB

CDC

- 32-bit processor, 2 MB RAM, 2.5 MB ROM
- VM – complete features of JSE VM
- CDC is superset of CLDC
- `java.io`, `java.util.zip`, `java.util.jar`, `java.net`, `java.security`



$CLDC \subseteq CDC$

CDC profiles

- Foundation Profile
 - core profile
 - no GUI
 - text manipulation, HTTP, sockets
 - java.math
 - java.util.zip, java.util.jar
 - certificates, encryption
- Personal Basis Profile
 - over FP, subset of PP
 - part of AWT, JavaBeans support
 - application – Xlet
 - RMI communication
- Personal Profile
 - similar to JSE
 - complete AWT

MIDP

- Mobile Information Device Profile
- over CLDC
- for mobile phones
- HW requirements (MIDP 1.0)
 - display min. 96x54x1
 - aspect ratio 1:1
 - keyboard or touch screen
 - 128 kB permanent memory
 - 8 kB permanent memory for applications data
 - 32 kB operating memory
 - duplex connection to network
- HW requirements (MIDP 2.0)
 - 256 kB permanent memory
 - 128 kB operating memory
 - sound

MIDP 1.0

- application – MIDlet
- support for GUI
- support for network communication (GCF)
 - HTTP
- persistent application data
 - Record Management Storage (RMS)
- over the air (OTA)
 - a way to install application to a device
- packages
 - `javax.microedition.midlet`
 - `javax.microedition.lcdgui`
 - `javax.microedition.rms`

MIDP 2.0

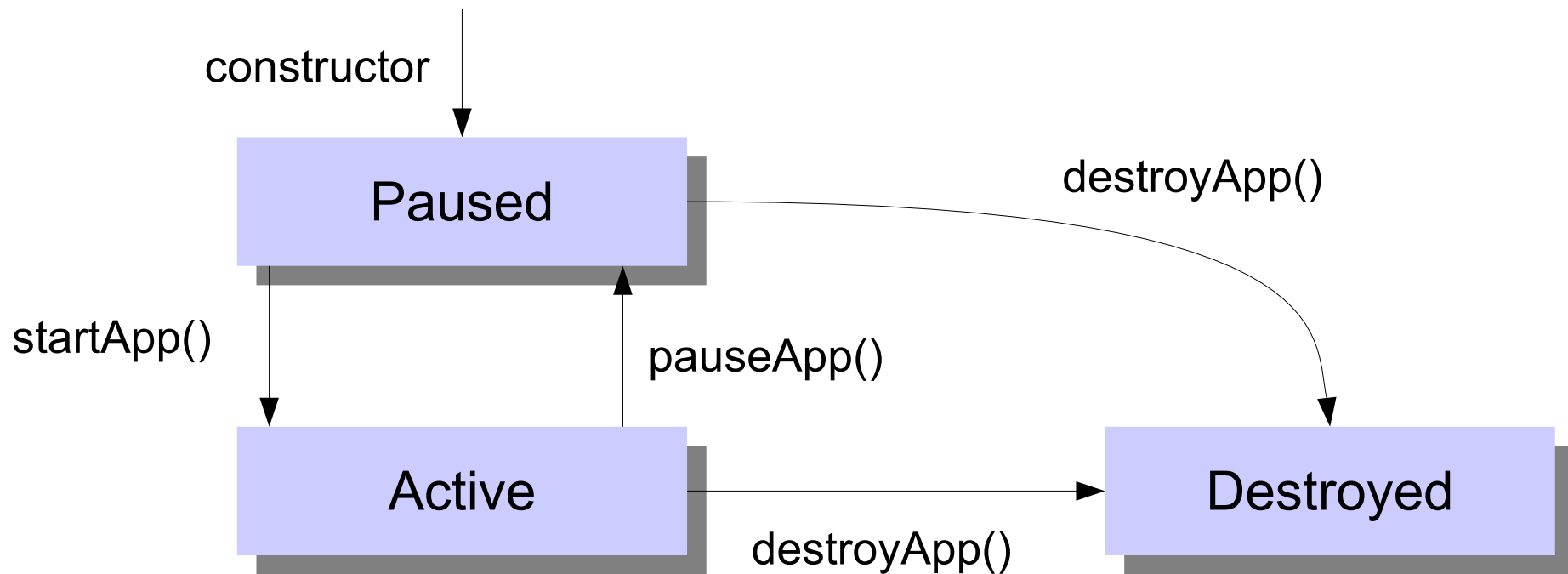
- better support of network
 - HTTPS, TCP and UDP sockets
- multimedia support
 - Mobile Media API (MMAPI)
- support for game creation
 - GameCanvas, Layers, Sprites
- certificates,...
- enhanced GUI
- push registry
 - launching MIDlets as a reaction to an incoming connection
- storage can be shared among several applications

MIDP 3.0

- JSR 271
 - December 2009
- parallel execution of several MIDlets and their communication
- support of IPv6
- LIBlets
 - shared libraries

MIDlet

- an application for MIDP
- similar to applets
- extends `javax.microedition.midlet.MIDlet`
- application lifecycle



Methods of MIDlet

- startApp()
 - called when when the *ACTIVE* state is entered
 - intended to be overridden
- pauseApp()
 - called when when the *PAUSED* state is entered
 - intended to be overridden
- destroyApp(boolean unconditional)
 - called when when the *DESTROYED* state is entered
 - if the parameter is *false*, the midlet can refuse to be destroyed
 - intended to be overridden
- notifyDestroyed()
 - terminates the midlet (destroyApp is not called)

Methods of MIDlet (cont.)

- notifyPaused()
 - the midlet wants to enter the *PAUSED* state
 - the pauseApp is not called
 - similar to notifyDestroyed
- resumeRequest()
 - opposite to notifyPaused
 - the midlet wants from the *PAUSED* state to *ACTIVE*
 - can be called e.g. from a timer or a background thread

MIDlet – implementation

```
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();
    }
}
```


MIDlet UI

- a single window can be shown at a single moment
 - several windows – switching

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

- if several MIDlets run concurrently, only one of them can access the display

MIDlet distribution

- 2 files
 - JAR archive – application code
 - JAD – Java Archive Descriptor
 - format
 - attribute-name: attribute-value
 - the same information must be also in the JAR manifest
- a JAD example

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

MIDlet distribution (cont.)

- several midlets can be in a single package
 - 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
- the descriptor can contain user-defined attributes
 - can be obtained from the application
 - MIDlet.getAppProperty(String key)

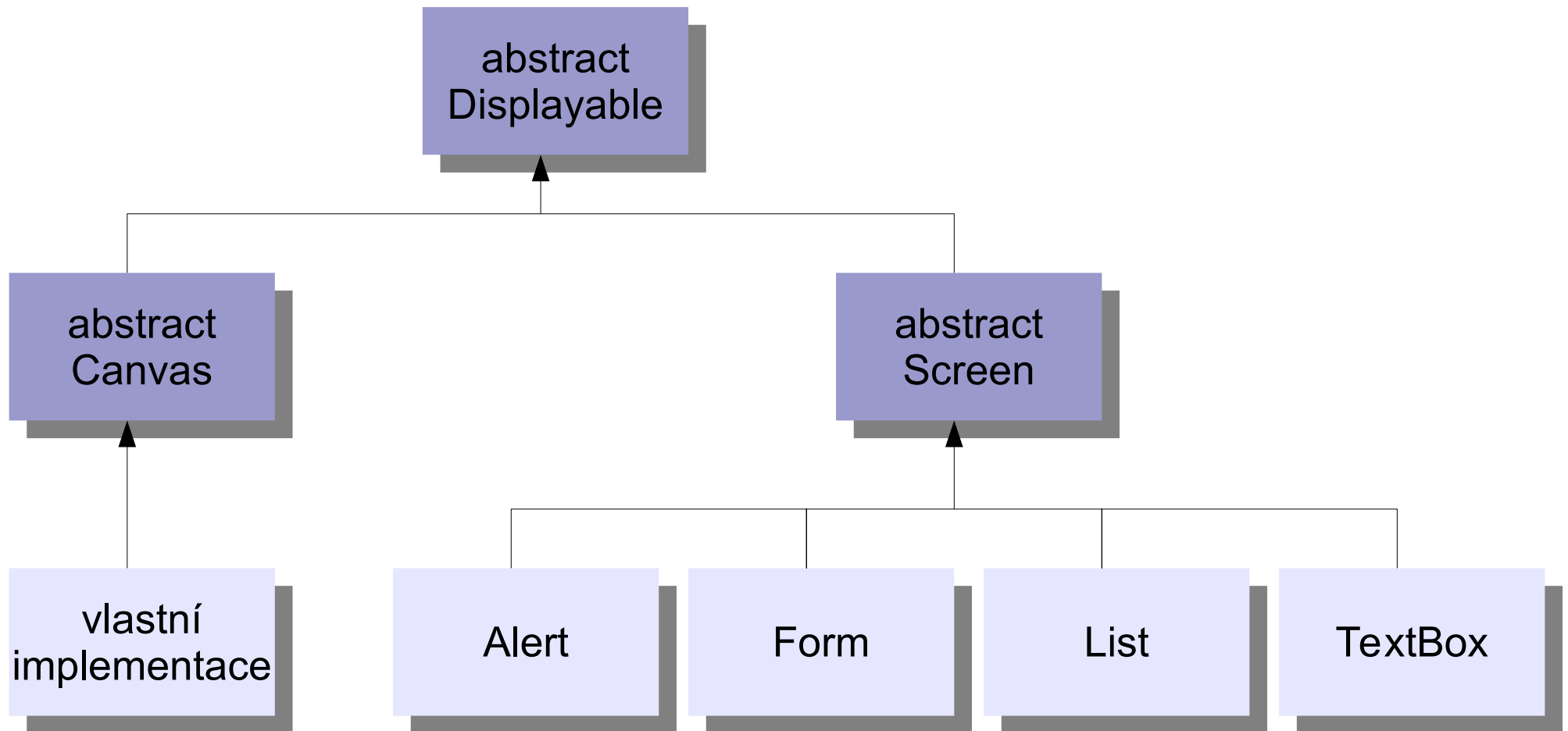
Record Management Store

- storing byte arrays
 - it is not a filesystem
- each midlet has own storage
 - MIDP 2.0 – storages can be shared
- operations are atomic
- stored data are persistent
- if the midlet is removed from a device, its storage is also deleted
- the `javax.microedition.rms` package
 - the `RecordStore` class
 - `openRecordStore()`
 - `addRecord()`
 - `getRecord()`

GUI

- the javax.microedition.lcdui package
- low-level
 - Canvas
 - drawing to display
 - handling keyboard/touch events
- high-level
 - device independent
 - low-level features cannot be influenced
 - fonts, etc.
 - portable

GUI



GUI – MIDP 2.0

- `javax.microedition.lcdui.game`
 - `GameCanvas`
 - extends `Canvas`
 - allows for
 - querying keys states
 - off-screen buffer
 - `Layer`
 - the abstract class for visible elements of a game
 - children
 - `Sprite`
 - `TiledLayer`
 - `LayerManager`
 - the manager of the visible elements

GUI – MIDP 2.0

- javax.microedition.media
 - playing multimedia
 - the Manager class
 - static methods
 - 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)

Optional packages

- extend profiles
- defined based on JCP
- separately for CLDC or CDC (or for both)

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

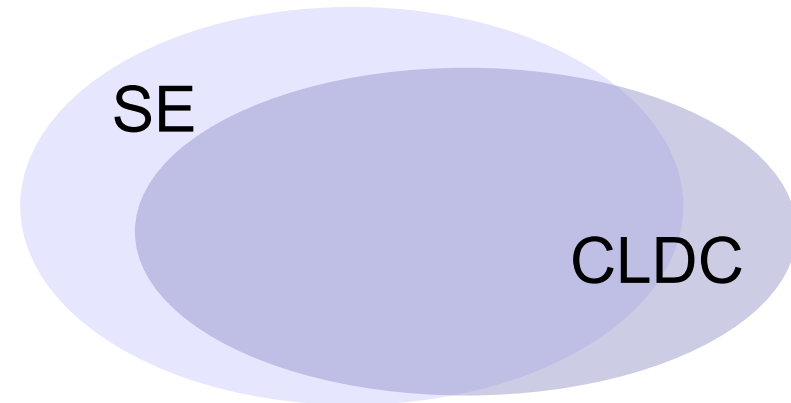
- JME RMI Optional Package (RMI OP) JSR 66
- JDBC Optional Package for CDC/Foundation Profile API JSR 169

Java ME 8

- 2014
- goal – unifying ME and SE
- CLDC 8
- MEEP 8
 - ME Embedded Profile 8

CLDC 8

- CLDC 8 – extended strict subset of SE 8
- VM supports
Java VM specification for SE 7
 - without
 - the InvokeDynamic instruction
 - reflection and runtime annotations
- language almost as Java 8
 - without
 - lambda functions
 - reflection
 - serialization
 - JNI
 - user-defined classloaders
 - ...



CLDC 8

- verification
 - bytecode versions 51+ (JDK 7+)
 - without preverification
 - bytecode versions 48 and older (JDK 1.4)
 - mandatory preverification
- enhanced Generic Connection Framework
 - supporting more protocols
 - IP multicast
 - specific options for protocols
 - ConnectionOption
 - listing “access points”
 - 3GPP, CDMA, Wi-Fi,...
- supporting ServiceLoader

MEEP 8

- Java ME Embedded Profile (MEEP) 8
- built on CLDC 8
- profiles
 - minimal
 - core API, application model
 - minimum – 128 kB RAM & 1 MB Flash
 - standard
 - services, multitasking, ...
 - minimum – 512 kB RAM & 2 MB Flash
 - full
 - complete API
 - minimum – 2 MB RAM & 4 MB Flash

MEEP 8

- packages
 - mandatory
 - javax.microedition.midlet
 - optional
 - 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

- applications
 - MIDlets (IMlets), LIBlets
 - javax.microedition.midlet.MIDlet
 - notifyPaused(), pauseApp(), resumeRequest() deprecated
- services
 - ServiceLoader
 - service provider and consumer can be in different applications

MEEP 8

- Device I/O API
 - accessing devices
 - GPIO, I2C, SPI, UART,...

Java Embedded

- a complete Java platform
- several variants
 - Java ME Embedded
 - Java ME Embedded Client
 - ...

Java ME Embedded

- based on MEEP and CLDC
- intended for microcontrollers, etc.
- headless
 - no UI
- platforms
 - ARM
 - Raspberry Pi
 - STM32
 - ...
- < 1 MB RAM

Java ME Embedded Client

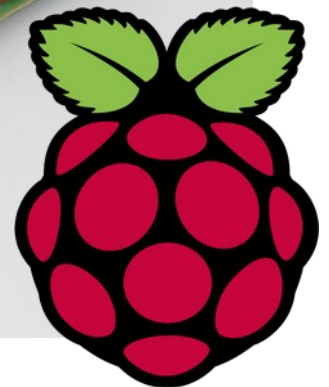
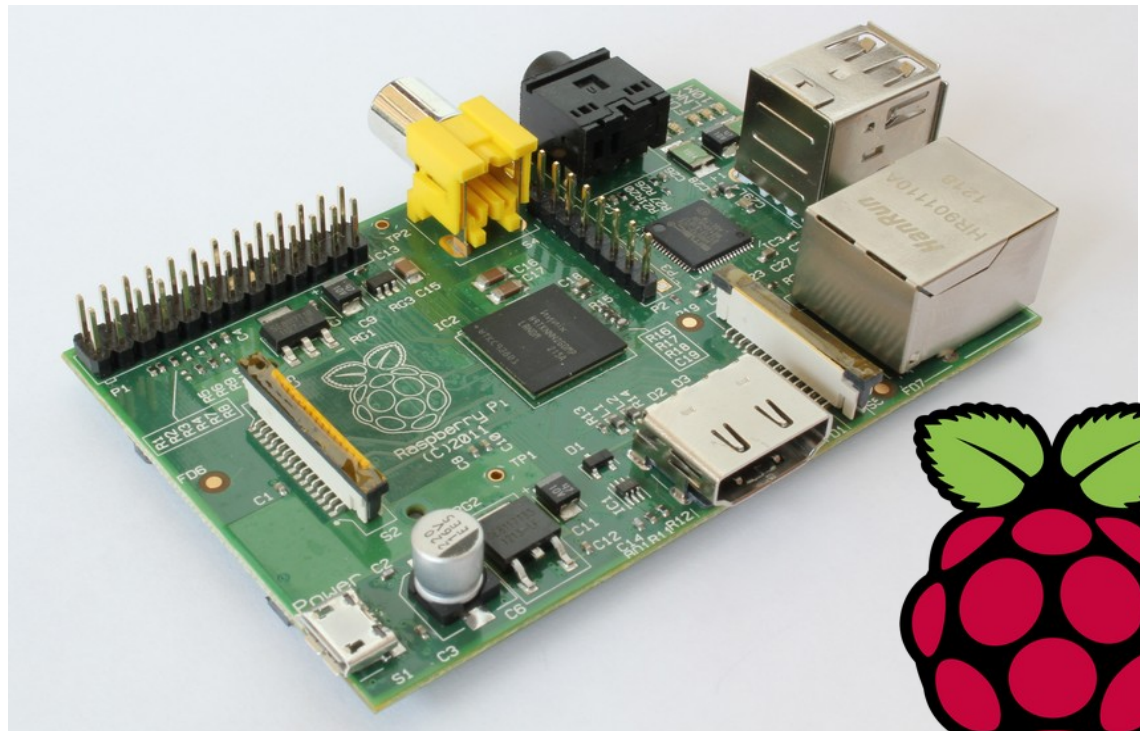
- based on JME and CDC
- < 10 MB RAM

JAVA

Pi4J

Pi4J

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



Pi4J: example

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

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

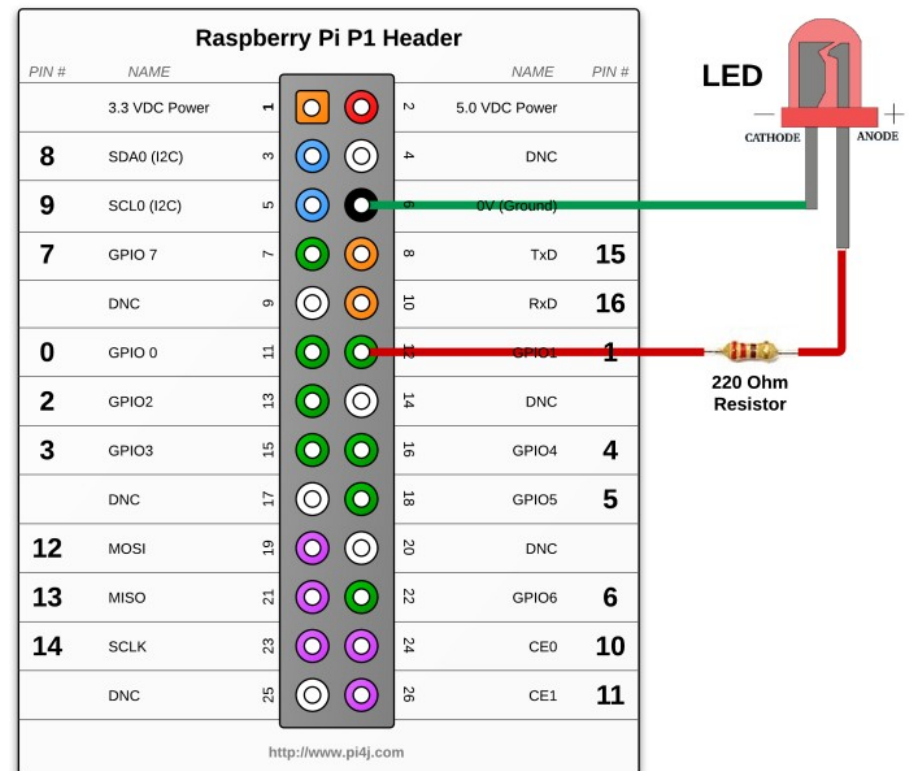
Thread.sleep(5000);

pin.low();

Thread.sleep(5000);

pin.pulse(1000, true);

gpio.shutdown();
```



JAVA

Real-Time Java

Real-time system

- non-real-time system
 - a system behaves correctly if produces correct results
- real-time system
 - a system behaves correctly if produces correct results at required time

Real-time system

- example
 - a medical device has to detect changes of patient state and react on time

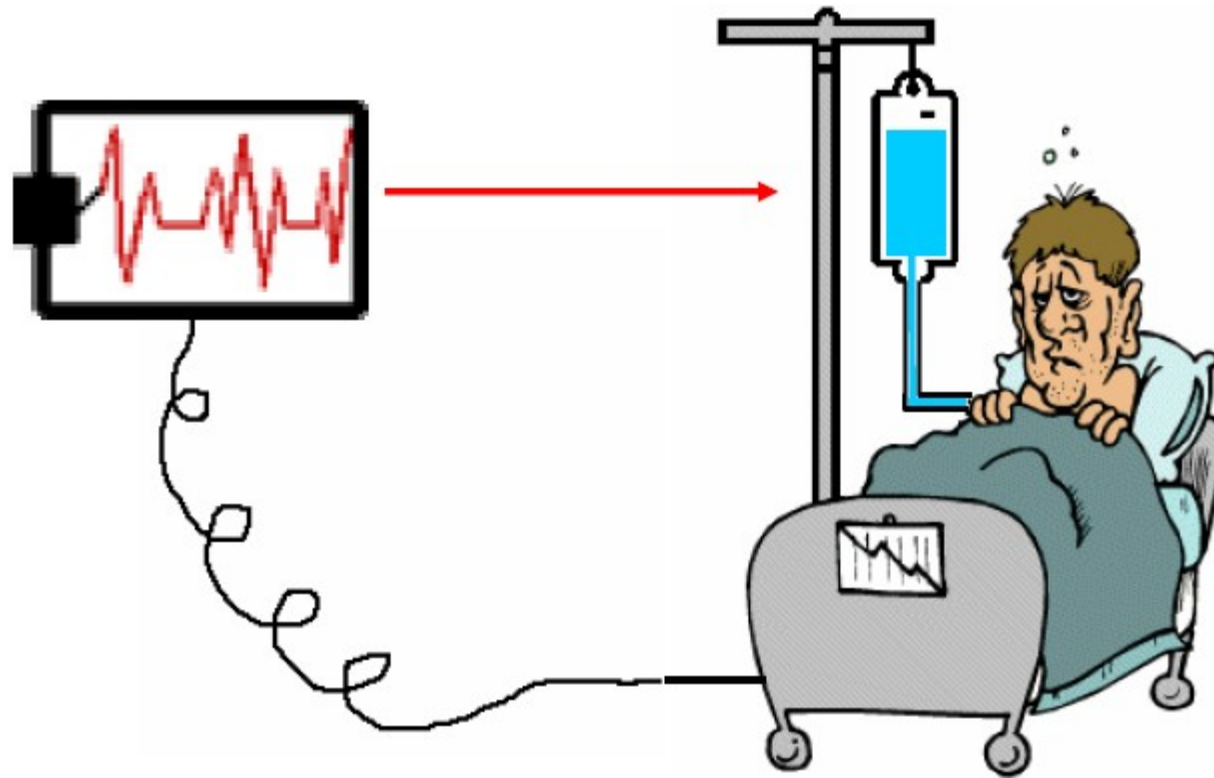


image source Issovic, D.:Real-time systems, basic course

Real-time system

- or...

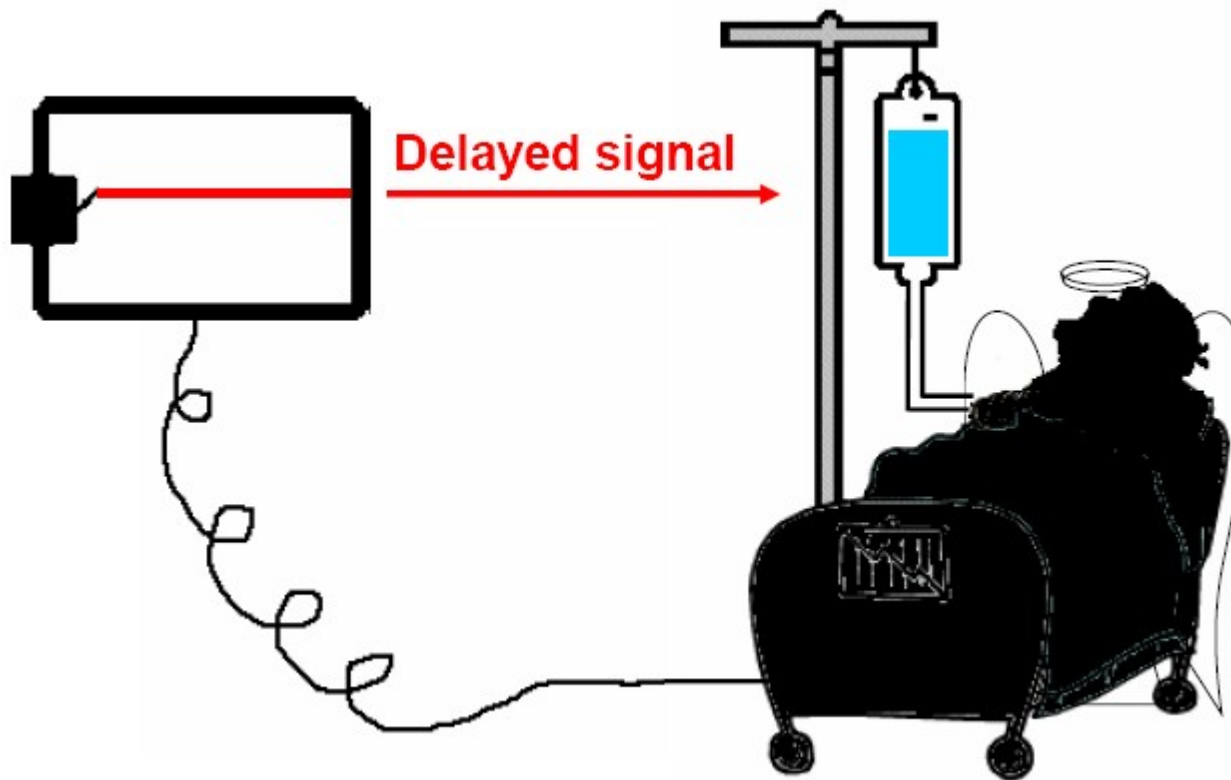


image source Issovic, D.:Real-time systems, basic course

Real-time system

- example
 - the airbag cannot inflate too early or too late

real-time \neq quickly

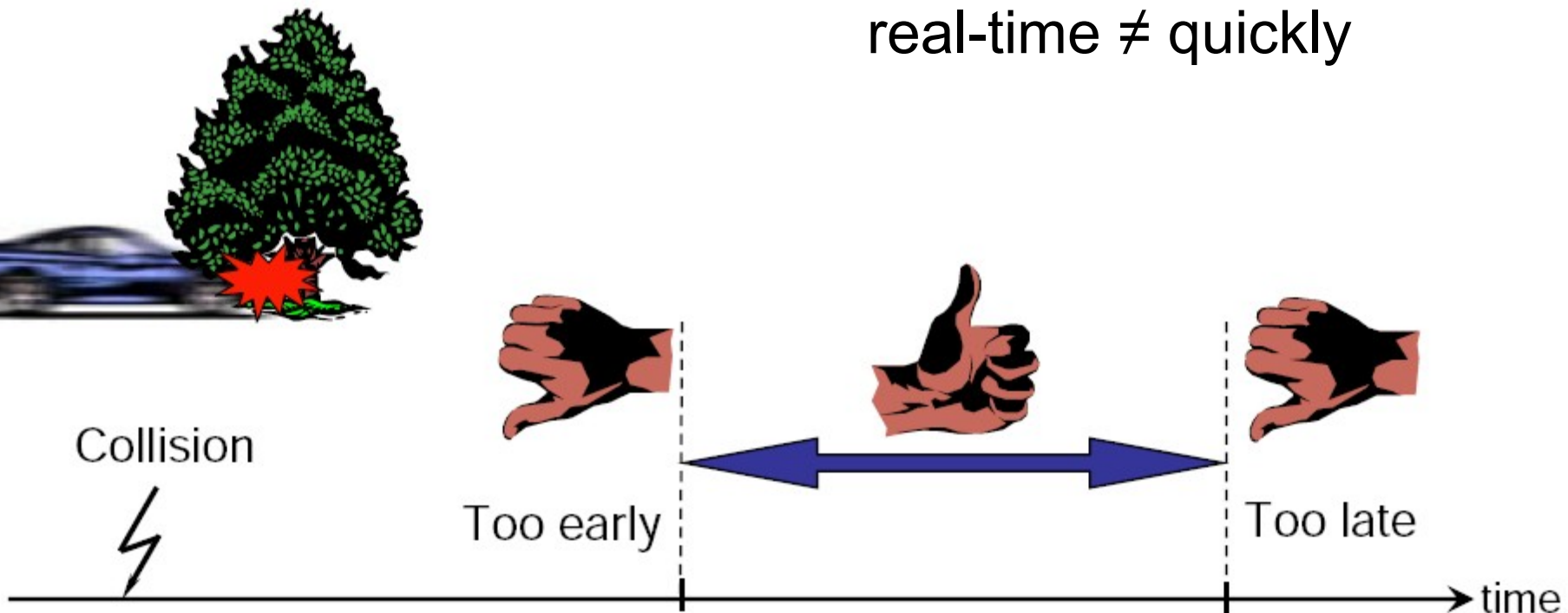


image source Issovic, D.:Real-time systems, basic course

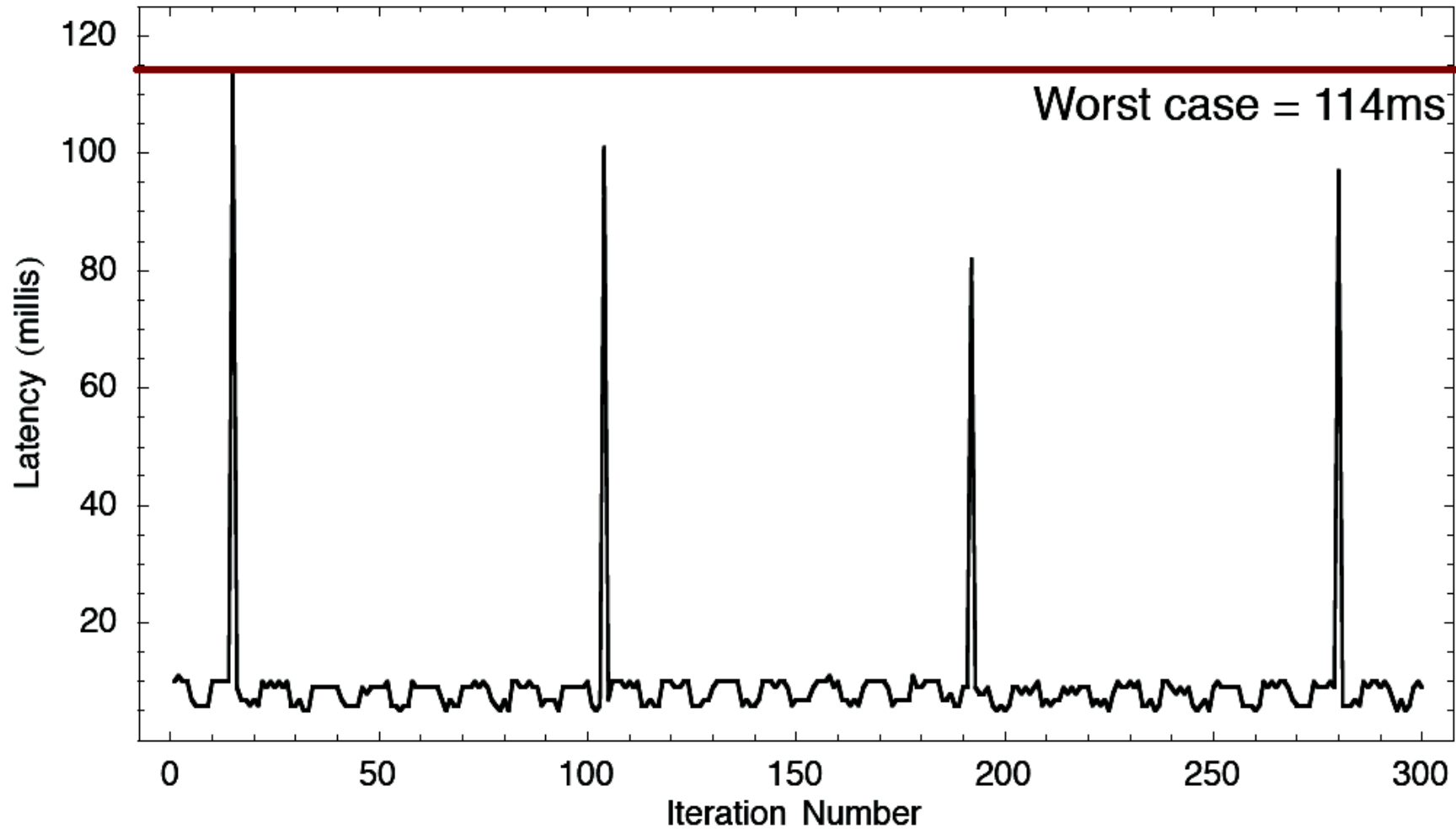
Real-time system

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

Java and RT

- Java
 - simple
 - widely used
 - many libraries
 - portable
- but
 - no real-time scheduling
 - no support for periodic execution
 - no support for aperiodic events
 - GC issues
 - issues with direct access to memory
 - issues with managing devices
 - ...

Garbage collector



Real-time Specification for Java

- RTSJ
- 1999 – JSR-1
- no changes in syntax
- it extends Java by
 - 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 – scheduling

- Fixed-priority round robin scheduler
 - own one can be added
- At least 28 real-time priorities (in addition to 10 common ones)

- Periodic threads
 - can start at specific time
 - have period and deadline

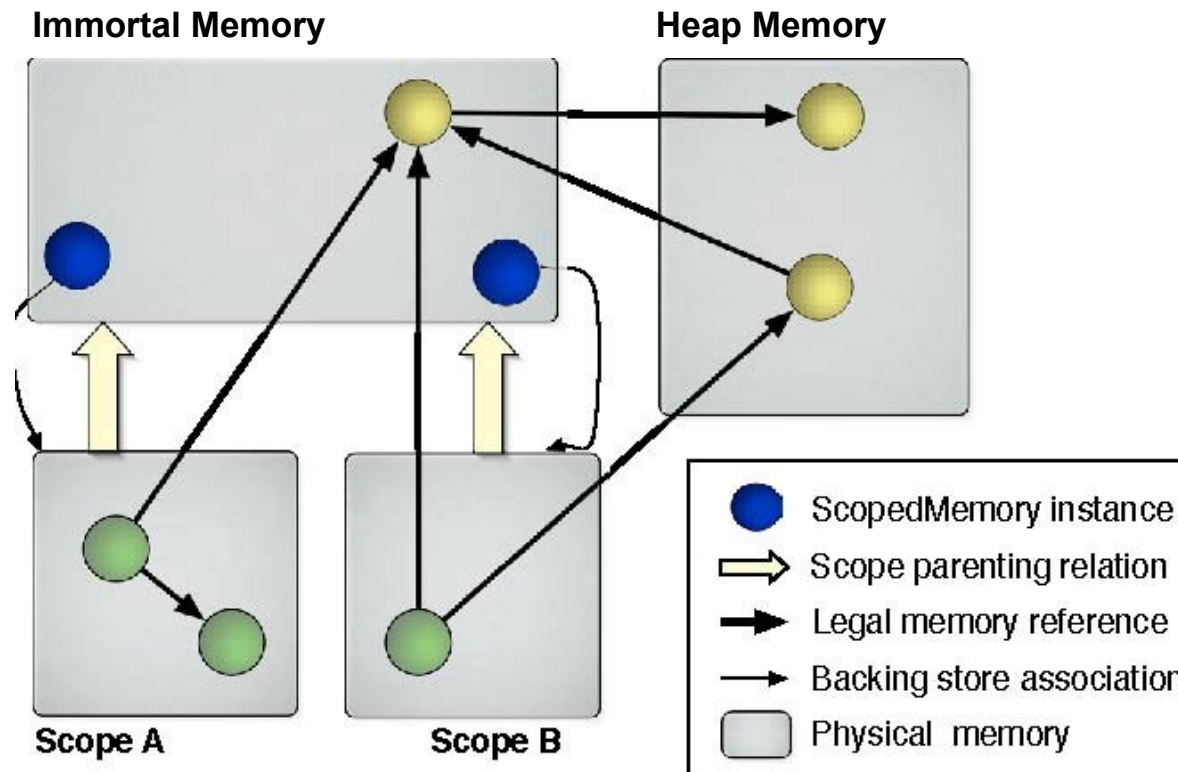
- Aperiodic events
 - a schedulable object, which is executed as a reaction to an event

RTSJ – memory

- NoHeapRealtimeThread
 - a thread without heap access
 - cannot be blocked by GC
- heap
 - as usual
- immortal memory
 - objects in the im. memory cannot be freed
 - for global data
- scoped memory
 - memory regions
 - objects freed at once when all threads leave the region
 - suitable for calling methods from the std library

RTSJ – memory

- rules for references between objects



RTSJ

- problems
 - memory regions are not intuitive
 - change of the classical programming model with GC
 - assigning a reference can fail
- there are real-time garbage collectors

Ravenscar Java

- restriction of RTJS
- inspired by “Ravenscar for Ada”
- goal
 - better analyzability and predictability
- an example of the restriction
 - no GC

RTSJ

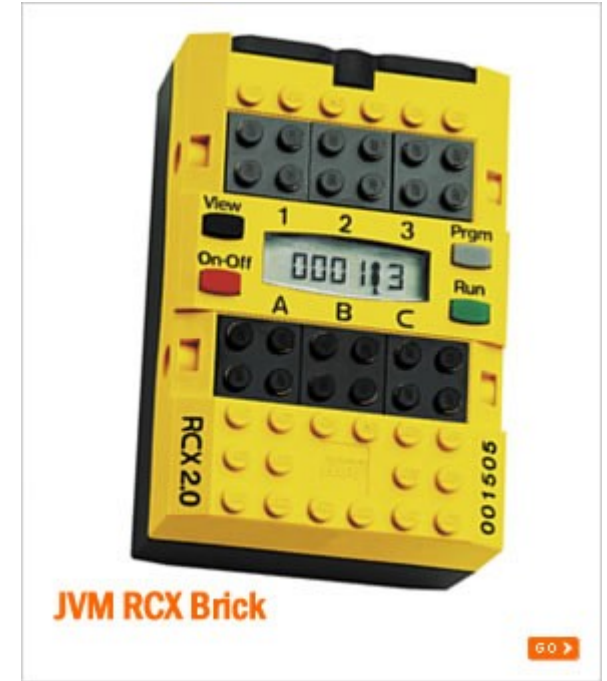
- RTSJ 2.0 – JSR 282
 - draft
- Base Module
 - Schedulables
 - Events & Handlers
 - Priority Inheritance
 - Clock
 - MemoryArea
 - HeapMemory
 - ImmortalMemory
 - ...
- Device
 - Happenings
 - RawMemory
 - ISR (Option)
- Alternate Memory
 - physical
 - scoped
- POSIX
 - POSIX signals

JAVA

LeJOS

Overview

- <http://www.lejos.org/>
- a firmware for LEGO Mindstorm
- contains a Java virtual machine
i.e. LEGO robots can be programmed in Java



Example

```
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

- Java 7
- mix Java SE a ME
- limitations
 - no classloaders
 - small size of applications
- after compilation, a binary image of the application is created
 - it is loaded to the “brick”
 - `nxjlink -v ClassWithMain -o App.nxj`
 - `nxjupload App.nxj`



Slides version AJ12.en.2020.01

This slides are licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).