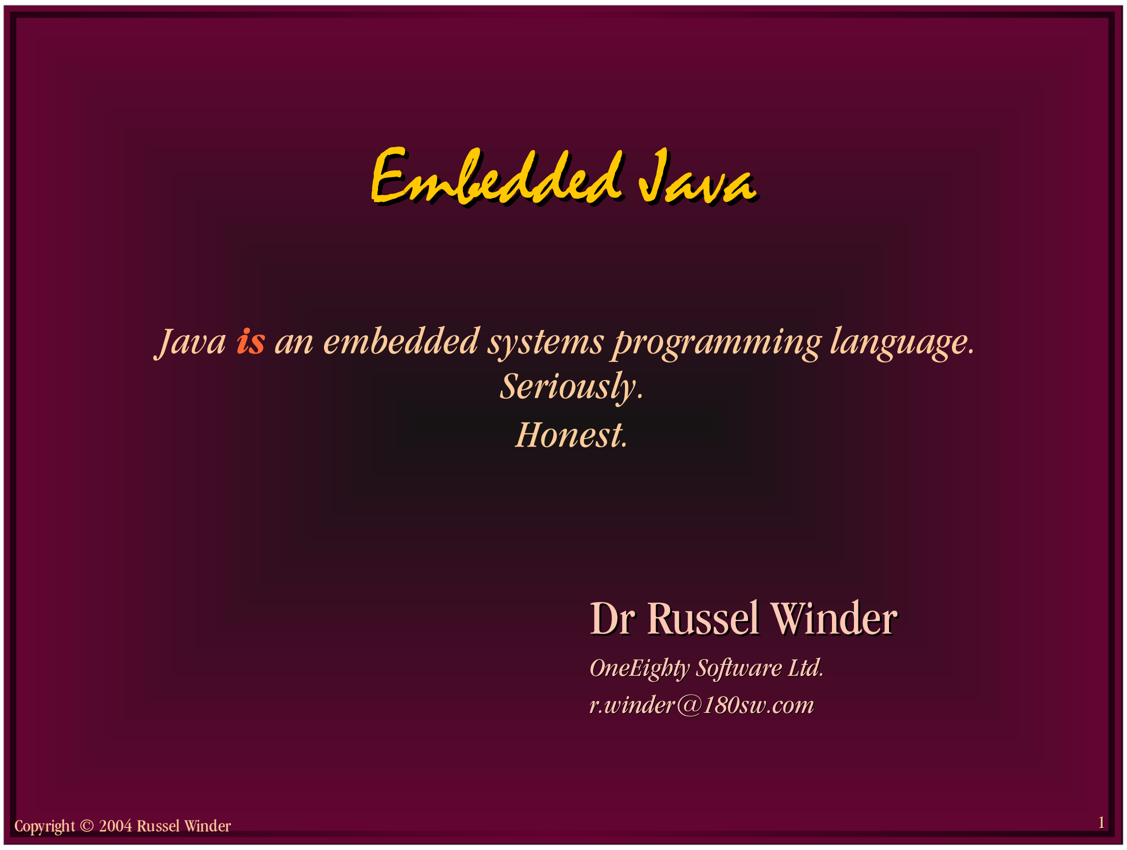
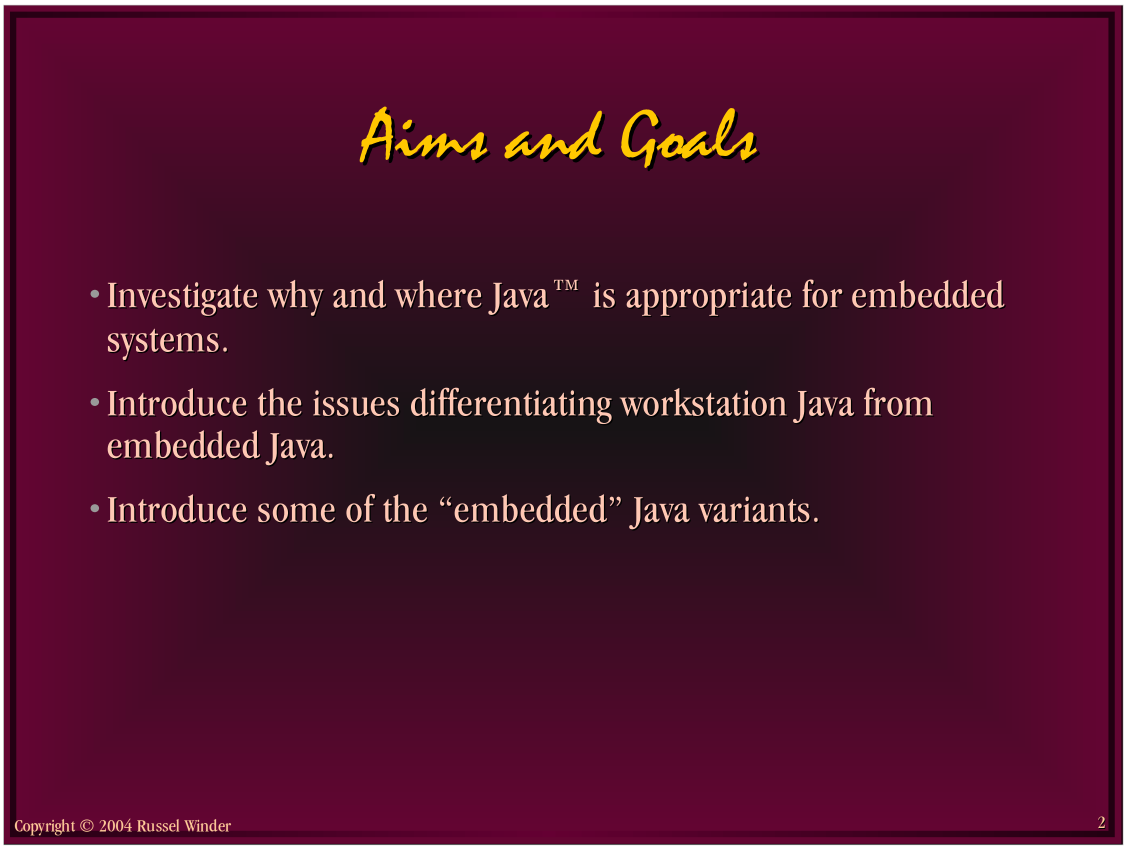


Embedded Java

*Java **is** an embedded systems programming language.*
Seriously.
Honest.

Dr Russel Winder
OneEighty Software Ltd.
r.winder@180sw.com

Copyright © 2004 Russel Winder 1



Aims and Goals

- Investigate why and where Java™ is appropriate for embedded systems.
- Introduce the issues differentiating workstation Java from embedded Java.
- Introduce some of the “embedded” Java variants.

Copyright © 2004 Russel Winder 2

The basic idea is to present some information and then to foment some discussion about the ideas and possibilities.

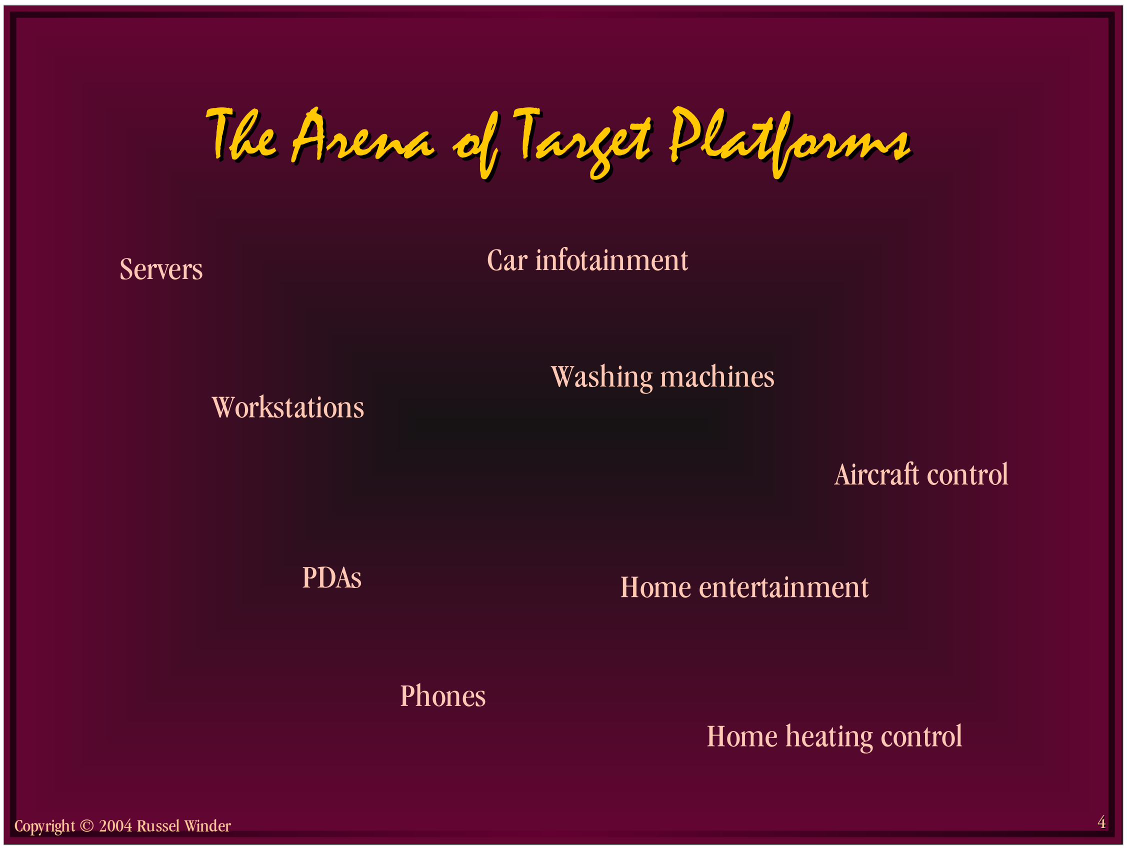
Sun's lawyers do not accept the use of the term Java as a noun since to do so would ruin, from a legal perspective, their trademark. So I should say Java programming language or Java programming system but all the “acceptable” labels are such an awful mouthful. Of course the irony is that everyone, including on many occasions Sun, uses Java as a noun without problem since context distinguishes the programming language from the island from the coffee. The point is that I readily acknowledge Sun's input into the Java Platform.

At this point I have to slip in a quick plug for OneEighty Software's principal product ORIGIN-J™. So here is the marketing slide.

ORIGIN - J
Infinite opportunities

OK so now that is over, I have witnesses (names will be taken) to say I performed the plug.

I shall now try to avoid any obvious marketing as this is not the time or place. If anyone wants details of ORIGIN-J there should be fliers somewhere around the place or please chat with me after the session.



The Arena of Target Platforms

Servers

Car infotainment

Washing machines

Workstations

Aircraft control

PDAs

Home entertainment

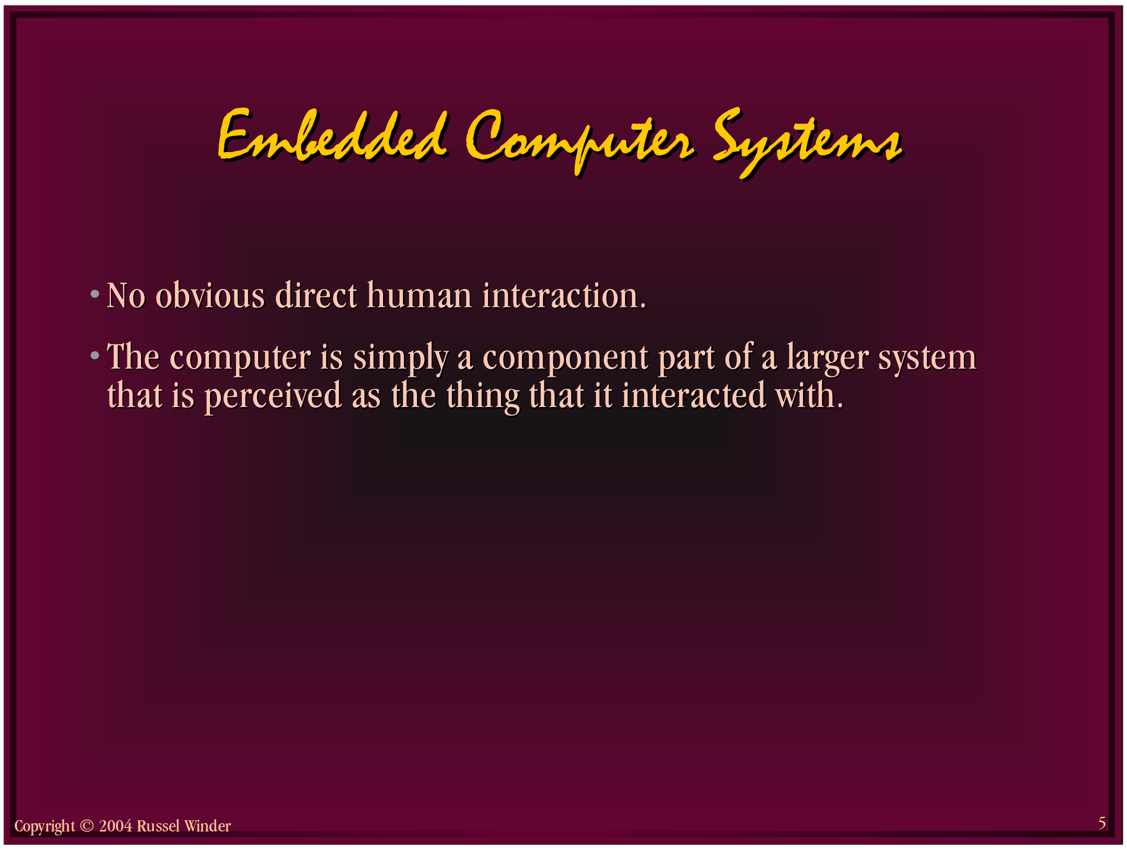
Phones

Home heating control

Copyright © 2004 Russel Winder 4

This is just a small selection of categories of computer system. Some — servers, workstations, etc. — are obviously computers some — aircraft control systems — are clearly not per se, though increasingly everyone knows there are actually computers in these non-computer devices.

So which of these are embedded and what defines embedded?



Embedded Computer Systems

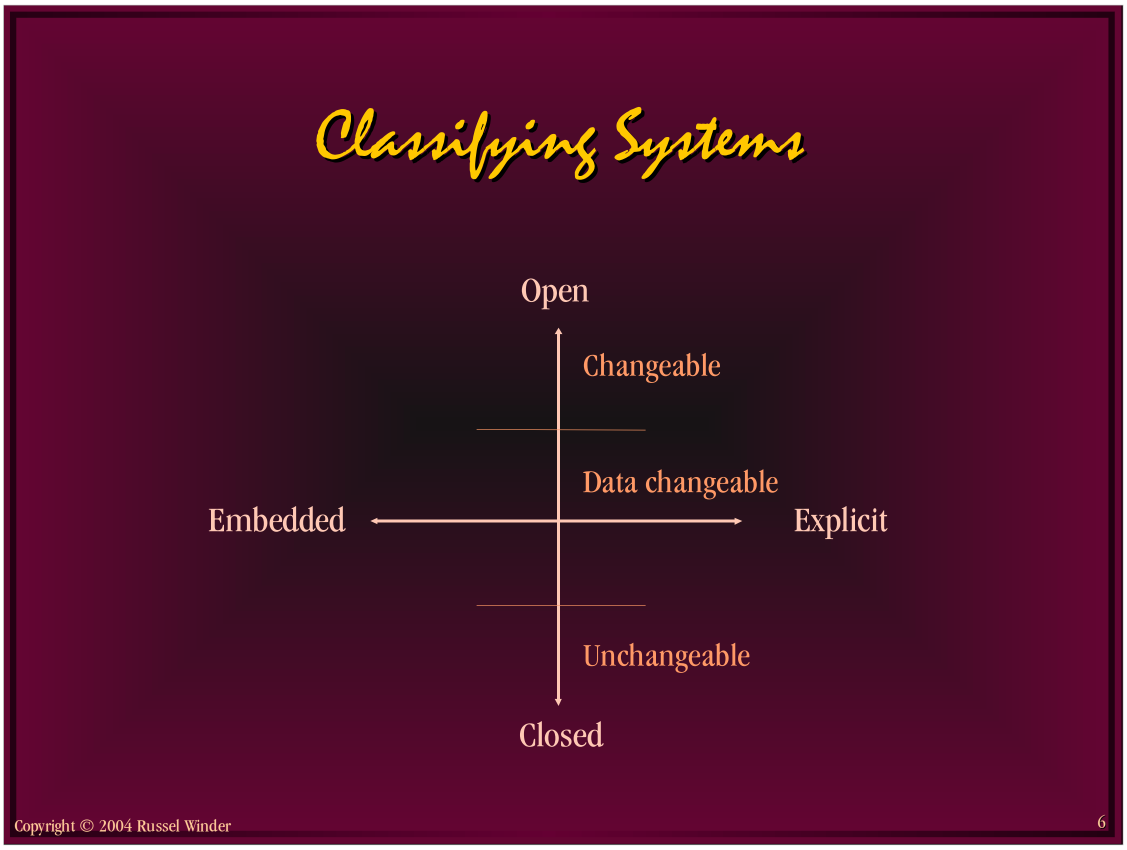
- No obvious direct human interaction.
- The computer is simply a component part of a larger system that is perceived as the thing that it interacted with.

Copyright © 2004 Russel Winder 5

It could be argued that this is a very HCI-centric “definition”. It is and I make no apologies for this as I think the purpose and perception of the computer system is what really matters.

Actually it is probably best to leave the definition at this since for every 10 computing people there are at least 12 definitions of embedded system.

However, it is worth deconstructing things a little more so as to find some of the dimensions and so be able to separate out good uses of Java in embedded systems.



Classifying Systems

Copyright © 2004 Russel Winder 6

Embedded vs. Explicit

A three way relationship between the computer system, user interface and user.

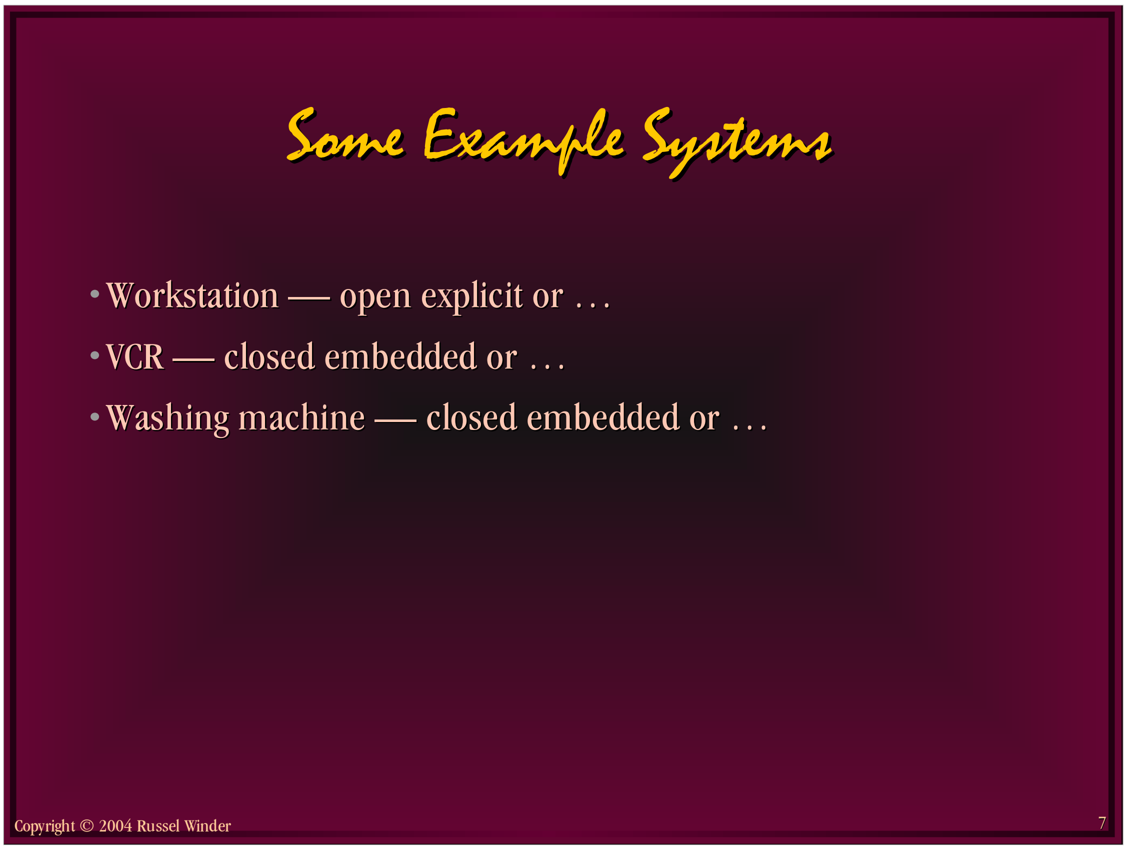
Open vs. Closed

Description of whether and how the behaviour of the system can be changed — can the data or code controlling the system be modified? Three categories:

Unchangeable — totally closed. The code and internal data cannot be changed; system is a "black box" which transforms incoming data into outgoing data.

Data changeable — neither closed nor open. The system can alter internal data tables to adapt to changing requirements.

Changeable — totally open. Both code and data can be changed to adapt to changing requirements.



Some Example Systems

- Workstation — open explicit or . . .
- VCR — closed embedded or . . .
- Washing machine — closed embedded or . . .

Copyright © 2004 Russel Winder 7

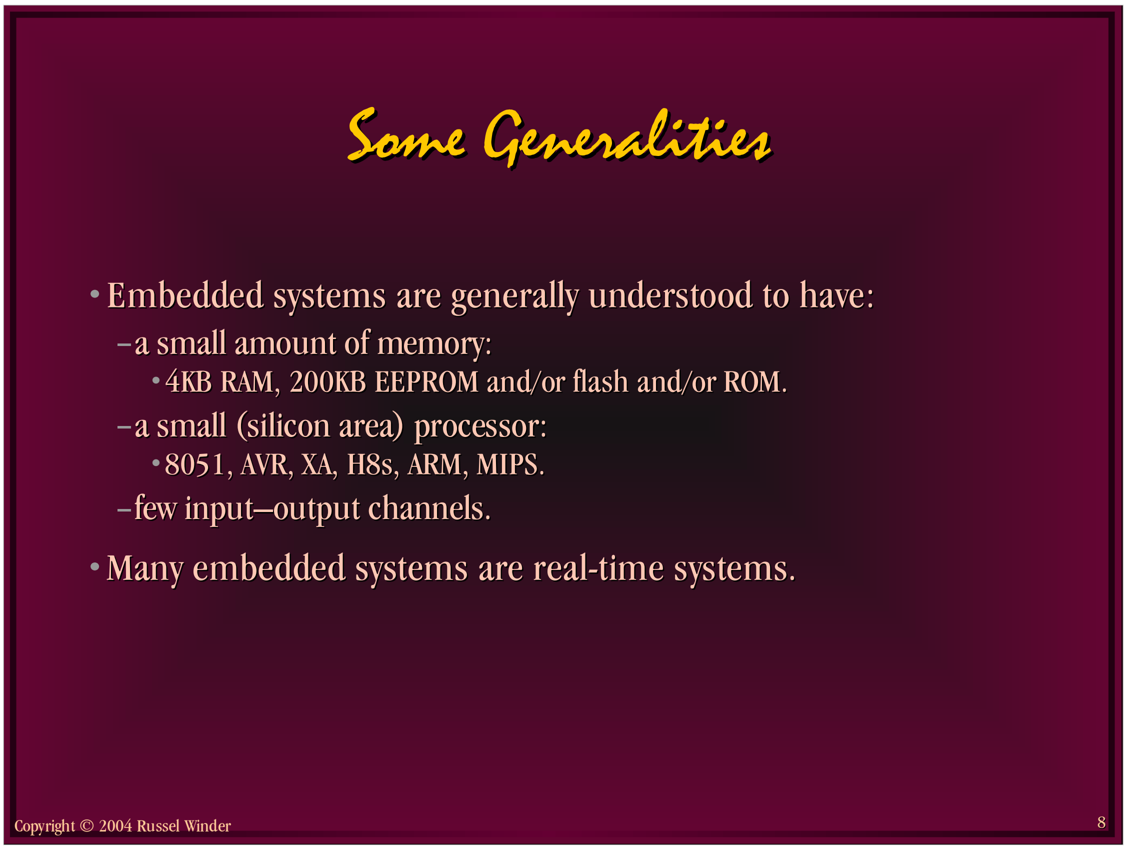
It may seem obvious where systems lie on the scale but it is not always as some systems can be perceived differently:

Workstations is clearly a computer and is open.

VCR is generally not perceived as a computer system but everyone knows they can be programmed which implies data changeable.

A washing machine is also programmed during use but should a washing machine do its own programming given that we only want to use a washing machine to do washing?

I will actually use smart cards as an example later on as this is an area of embedded systems development I am very familiar with.



Some Generalities

- Embedded systems are generally understood to have:
  - a small amount of memory:
    - 4KB RAM, 200KB EEPROM and/or flash and/or ROM.
  - a small (silicon area) processor:
    - 8051, AVR, XA, H8s, ARM, MIPS.
  - few input–output channels.
- Many embedded systems are real-time systems.

Copyright © 2004 Russel Winder 8

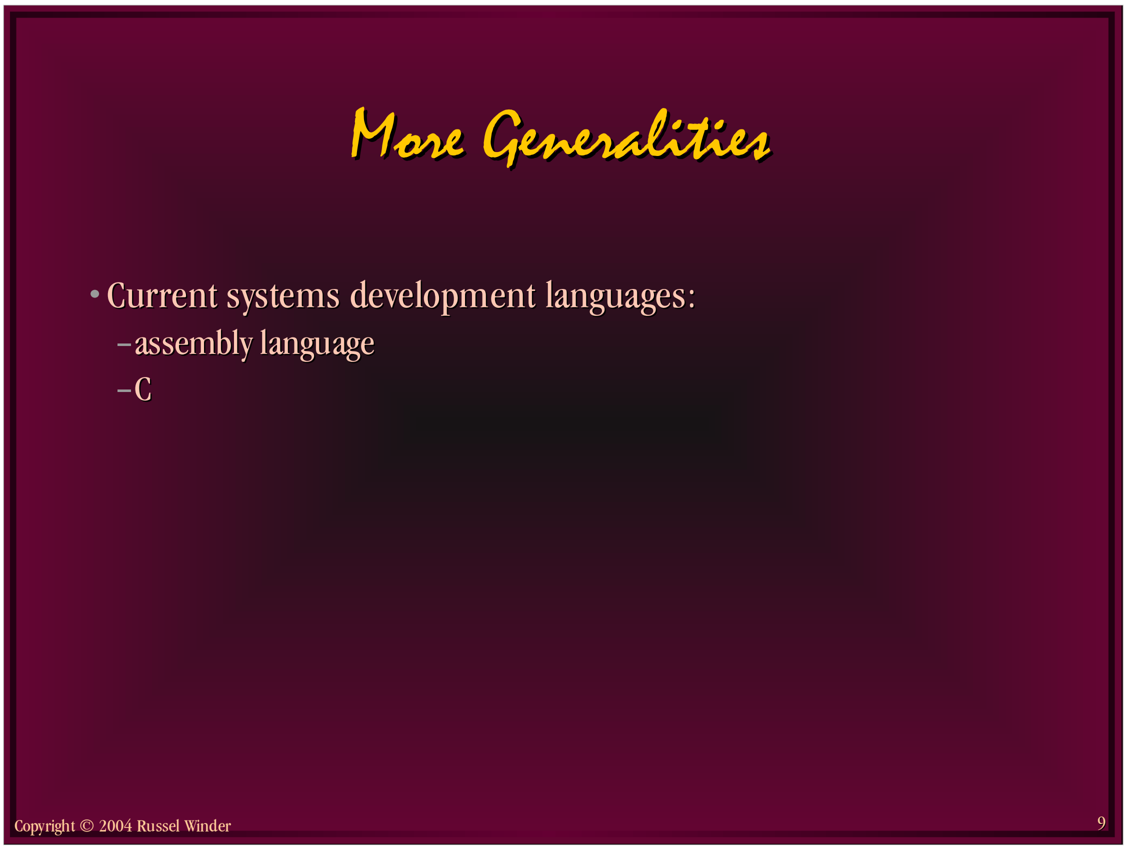
8051 and AVR are 8-bit, XA and H8s are 16-bit, ARM and MIPS are 32-bit.

Price is generally the primary driver.

Bizarrely, 32-bit processors can be as cheap as 8-bit processors due to different manufacturing techniques. Memory and memory buses are the primary costs, especially RAM.

For any general case there will always be exceptions but in the main the above is not unreasonable as a measure of the sort of system being dealt with under the banner "Embedded System".

Many embedded systems are real-time systems in that they are there simply to respond to events (interrupts). For me a real-time system is one that if it does not respond to a stimulus in a given period then it has failed.



More Generalities

- Current systems development languages:
  - assembly language
  - C

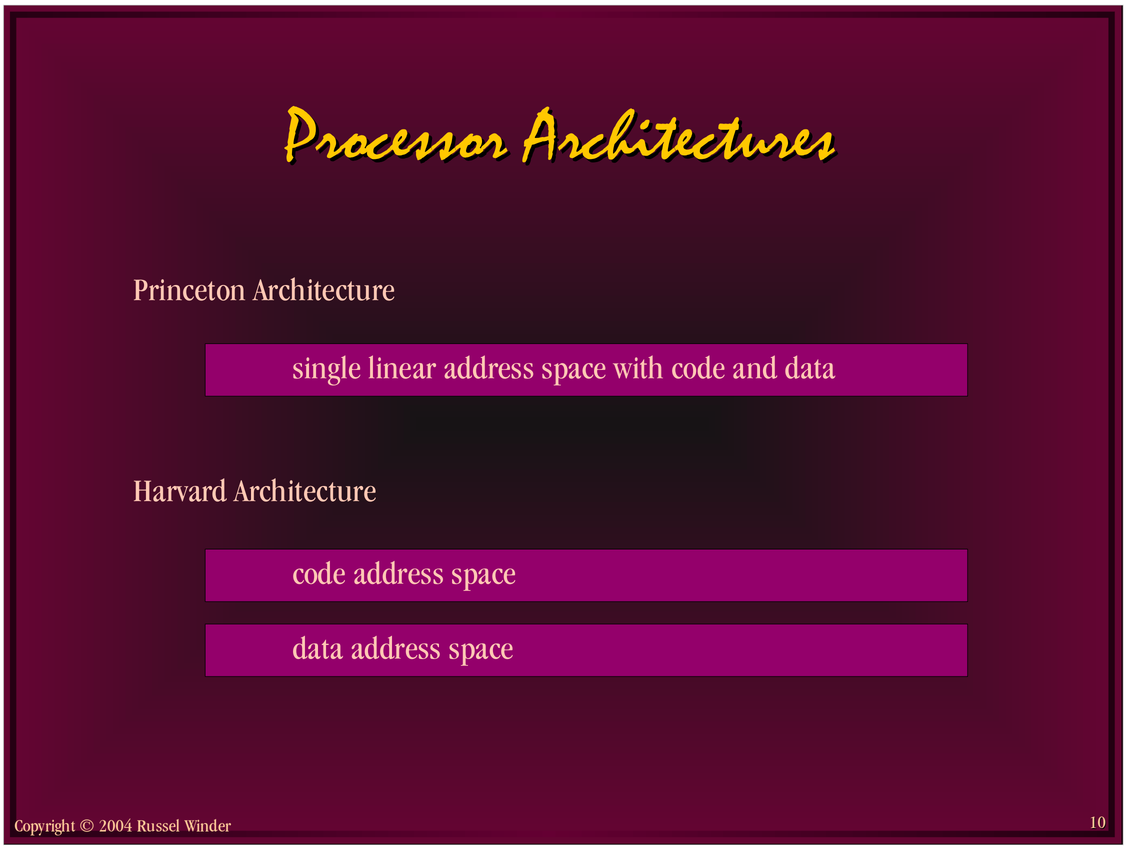
Copyright © 2004 Russel Winder 9

Until relatively recently it was assumed that assembly language was the only way to deal with embedded systems and in some cases this remains true. cf. 4-bit processors in cheaper washing machines.

C, in its guise as an high-level and/or portable assembly language, is really ruling the roost. Often this is C89 not C99. Often (cf. 8051) there are necessary language extensions. More on this on a later slide.

The C model is often a very unsuitable computational model for dealing with highly restricted EEPROM / flash / ROM based systems. Also there are serious problems with Havard architecture processors — as opposed to Princeton (von Neuman) architecture processors.

OneEighty Software's ORIGIN™ technology is a computational model with a high-level/portable assembly language system and translator build system designed to get round the problems with C.



Processor Architectures

Princeton Architecture

single linear address space with code and data

Harvard Architecture

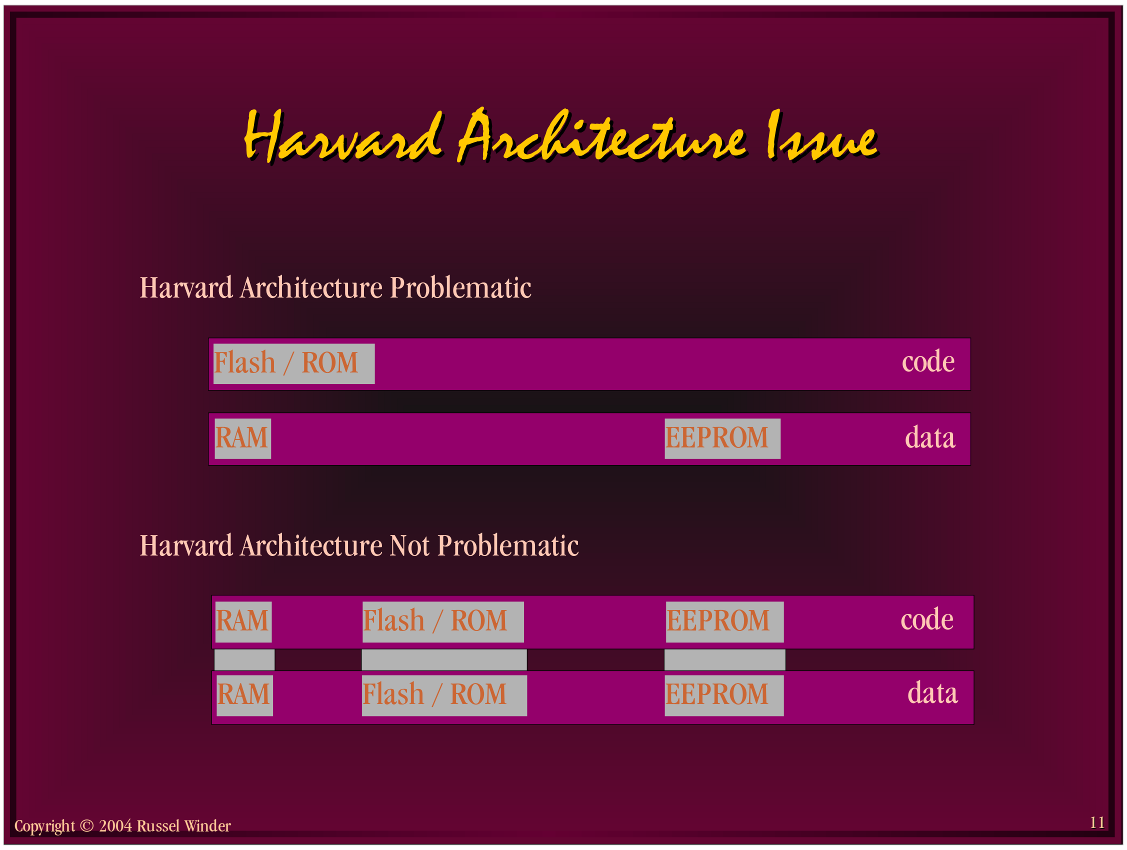
code address space

data address space

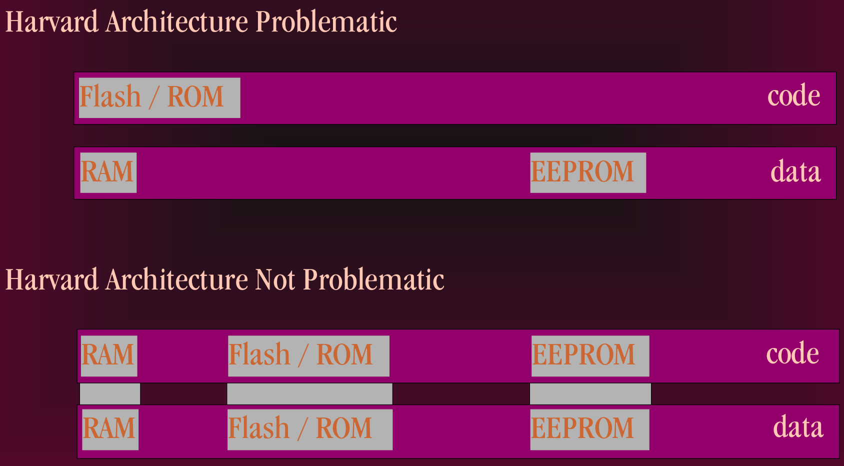
Copyright © 2004 Russel Winder 10

The computational model of C is really a model based on Princeton Architecture. It is often (cf. AVR processor) rammed (!) onto a Harvard Architecture but with a huge loss of certain aspect of expressivity.

Also note that the RAM / EEPROM / Flash / ROM physical memory structure underneath the address space causes serious problems sometimes for the C computational model.



Harvard Architecture Issue



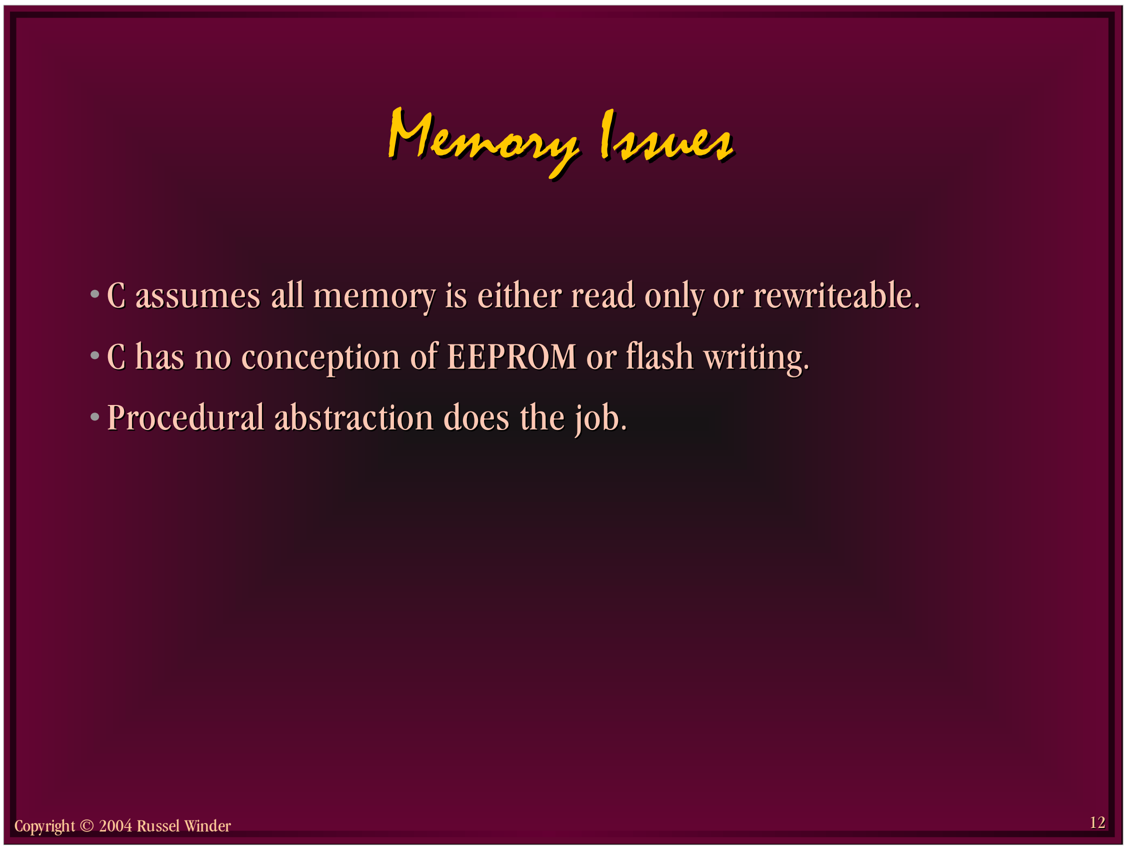
Copyright © 2004 Russel Winder

11

For many C systems there is no issue. Code is immutable and cannot be accessed as data so there is no problem with putting code in Flash / ROM that is only accessible as code. However `const` initialized data should also go in Flash / ROM but is then not accessible as data. The low level memory and addressing architecture is no longer easily mapped to by the C computational model.

Actually many C systems actually copy this data to RAM anyway for addressing mode speed purposes but this is an hack.

Ensuring that all physical memory appears in both code and data address spaces is not difficult and allows the C computational model to function properly since it is basically Harvard Architecture physically but Princeton Architecture abstractly.



Memory Issues

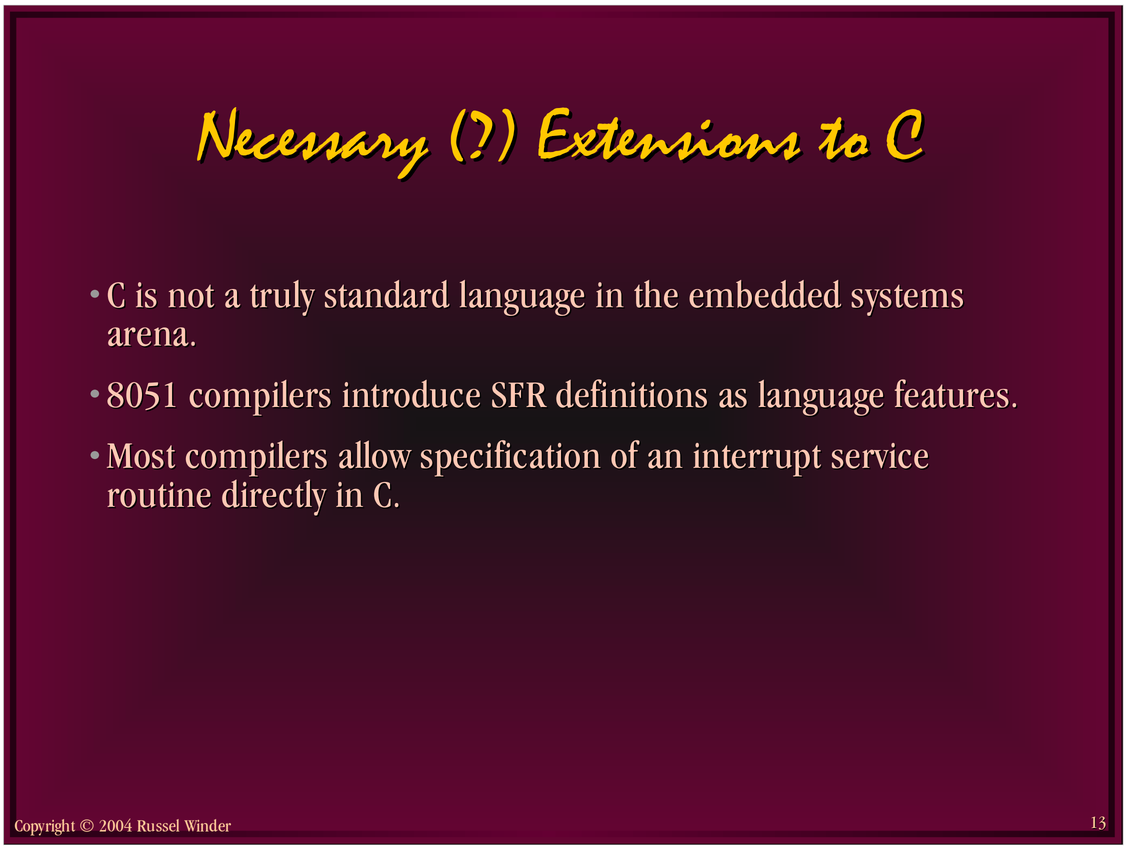
- C assumes all memory is either read only or rewriteable.
- C has no conception of EEPROM or flash writing.
- Procedural abstraction does the job.

Copyright © 2004 Russel Winder 12

C has no conception of a variety of memory that is easily readable but has a complex writing strategy.

For EEPROM or flash writing, data is transferred to a set of latches and then a write initiated and then the processor is paused until an end of write interrupt occurs.

Such writing is page based which seriously affects the measures of performance. Writing EEPROM or flash is slow (1-10ms). However, writing of a page (typically 128 bytes) is as fast as writing a single byte.



Necessary (?) Extensions to C

- C is not a truly standard language in the embedded systems arena.
- 8051 compilers introduce SFR definitions as language features.
- Most compilers allow specification of an interrupt service routine directly in C.

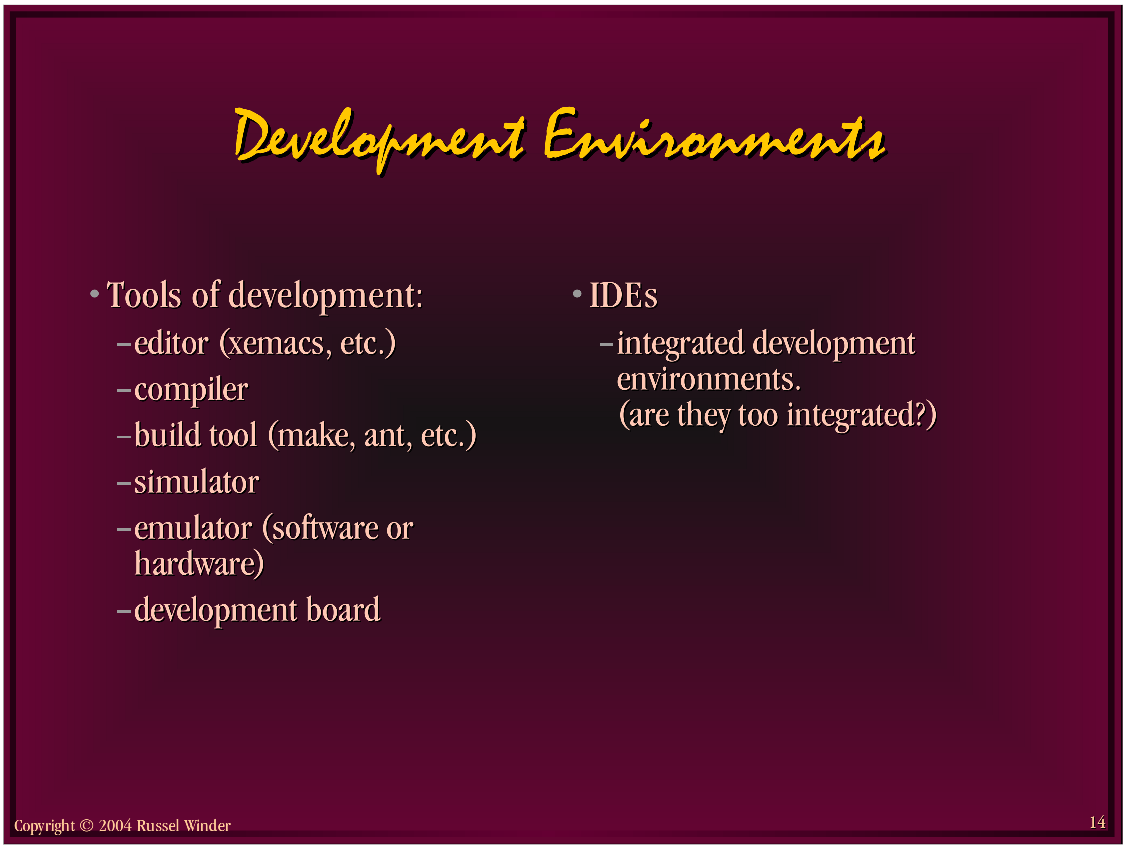
Copyright © 2004 Russel Winder 13

C99 (or more likely C89) compilers for embedded systems add extra, invariably target specific, features. These two are the classics.

8051 SFRs are generally added as direct language features not handled by C constructs. This is usually to get at the assembler instructions directly.

Standard C always uses return and cannot gernerate return from interrupt. Hence the need to bounce ISRs through assembler procedures. Most C compilers now allow annotation of ISRs so as to get at return from interrupt instruction.

Should also note, talking of 8051 that handling of bit data in hardware is effectively impossible in standard C.



Development Environments

- Tools of development:
  - editor (xemacs, etc.)
  - compiler
  - build tool (make, ant, etc.)
  - simulator
  - emulator (software or hardware)
  - development board
- IDEs
  - integrated development environments.
    (are they too integrated?)

Copyright © 2004 Russel Winder 14

<flame>

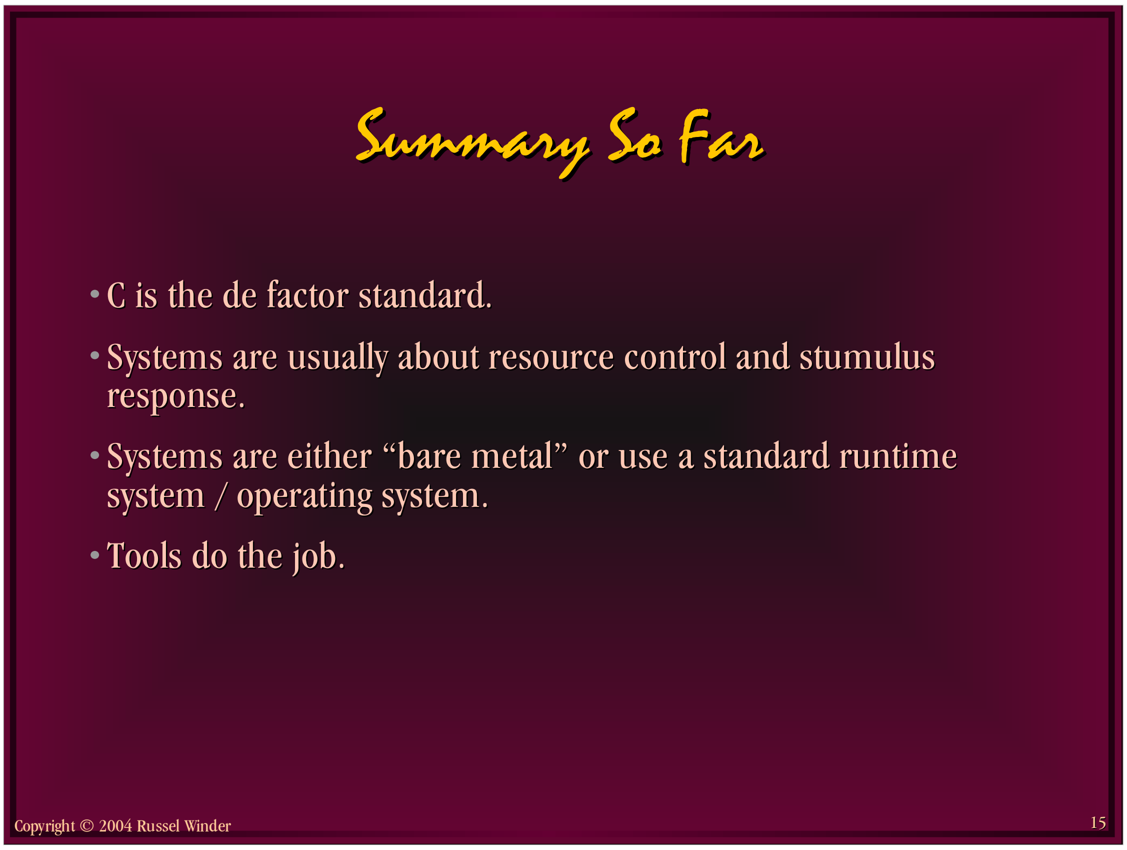
Most tool vendors think everyone uses Microsoft Windows. This is not true a very large number use Linux.

Most tool vendors think everyone will only use integrated development environments (IDEs). This is not true a very large number use make and command line compilers.

</flame>

Has to be said though Eclipse is probably making IDEs more attractive, especially for Java development. Also IntelliJ IDEA gets highly rated as well as Eclipse.

Come back to why this is an interesting aspect of development later.

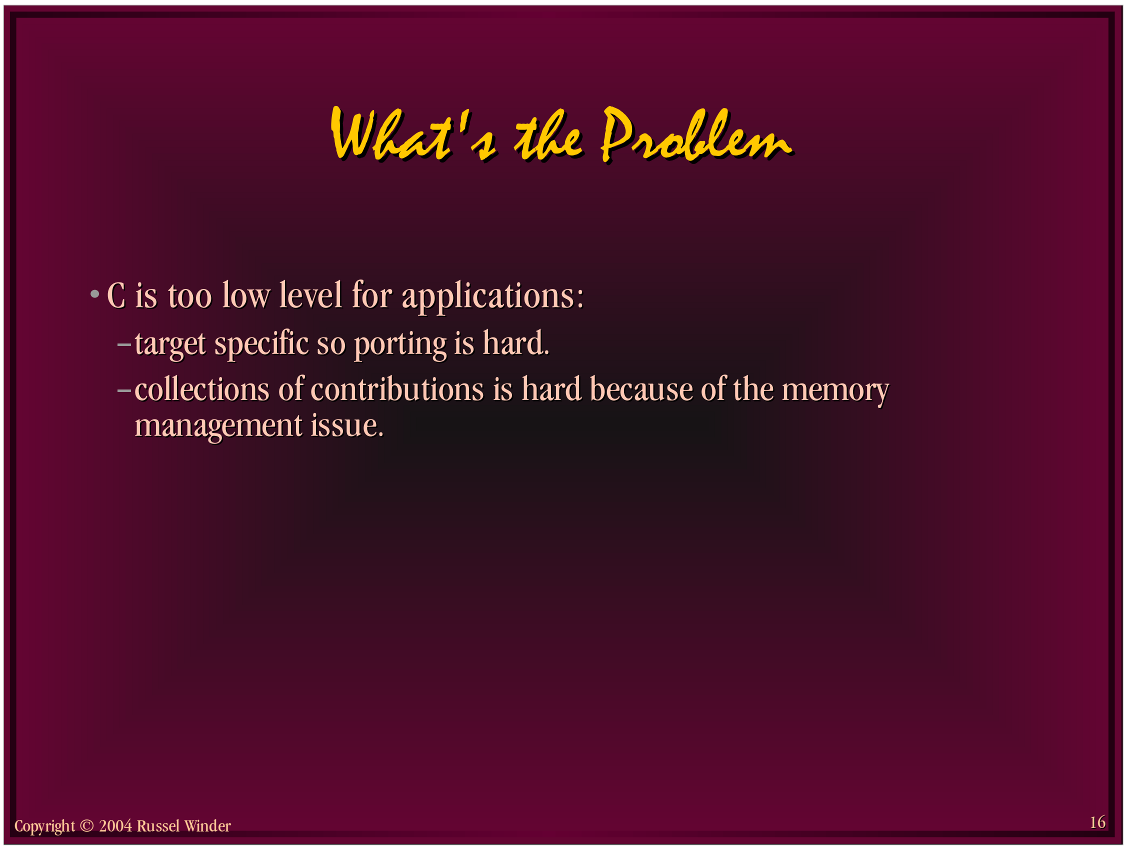


Summary So Far

- C is the de factor standard.
- Systems are usually about resource control and stumulus response.
- Systems are either “bare metal” or use a standard runtime system / operating system.
- Tools do the job.

Copyright © 2004 Russel Winder 15

So what is the problem, it seems that everything in the garden is fine.



What's the Problem

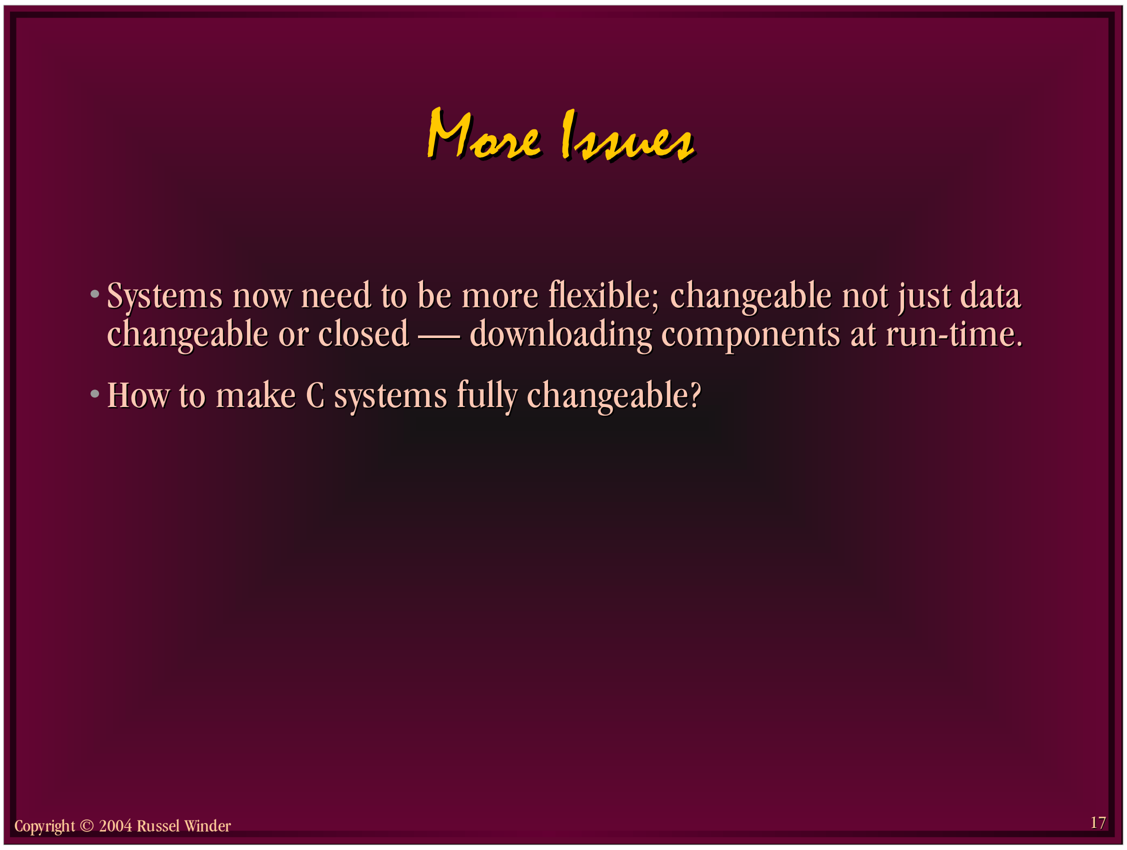
- C is too low level for applications:
  - target specific so porting is hard.
  - collections of contributions is hard because of the memory management issue.

Copyright © 2004 Russel Winder 16

Because of the target specific features to write most systems, porting is dreadful. Either that or, using the lowest comment denominator, C89, systems are very inefficient.

Creating a system by combining contributions from different development teams is a system integration nightmare. In particular testing for failures of memory management, i.e. pointer problems and memory leaks, is a real pain.

Vendors have products to help but still …



More Issues

- Systems now need to be more flexible; changeable not just data changeable or closed — downloading components at run-time.
- How to make C systems fully changeable?

Copyright © 2004 Russel Winder 17

Traditionally whole systems were defined and then created and were fixed: all embedded systems were closed or data changeable.

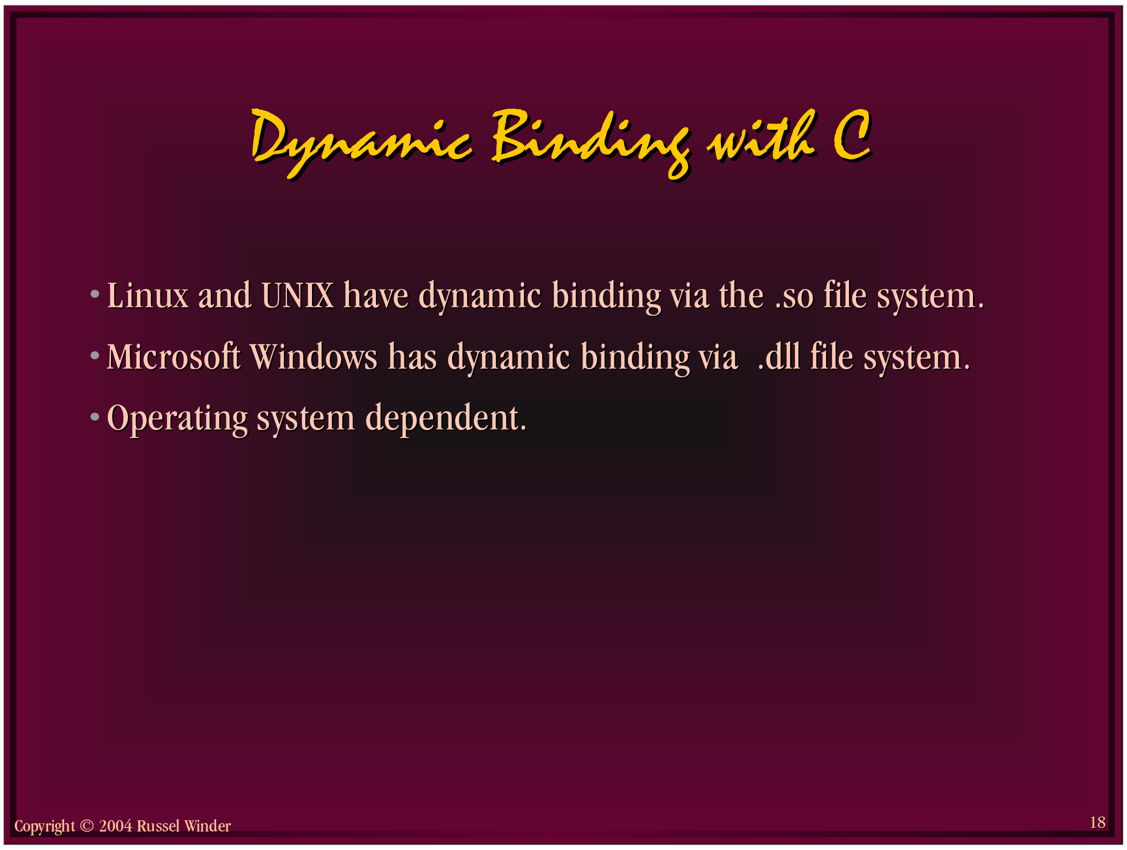
The use of ROM millitated in favour of this since a ROM mask had to be created.

Increased size of EEPROM with long data retention times (> 20 years) and the introduction of the use of flash has changed things.

Systems can now be extended / updated whilst in the field. This means that componenets of the running system are downloaded and installed during run time.

Assuming there is a communications system of course.

So in the area of network connected systems this is essential. Car or home infotainment, smart cards, etc.



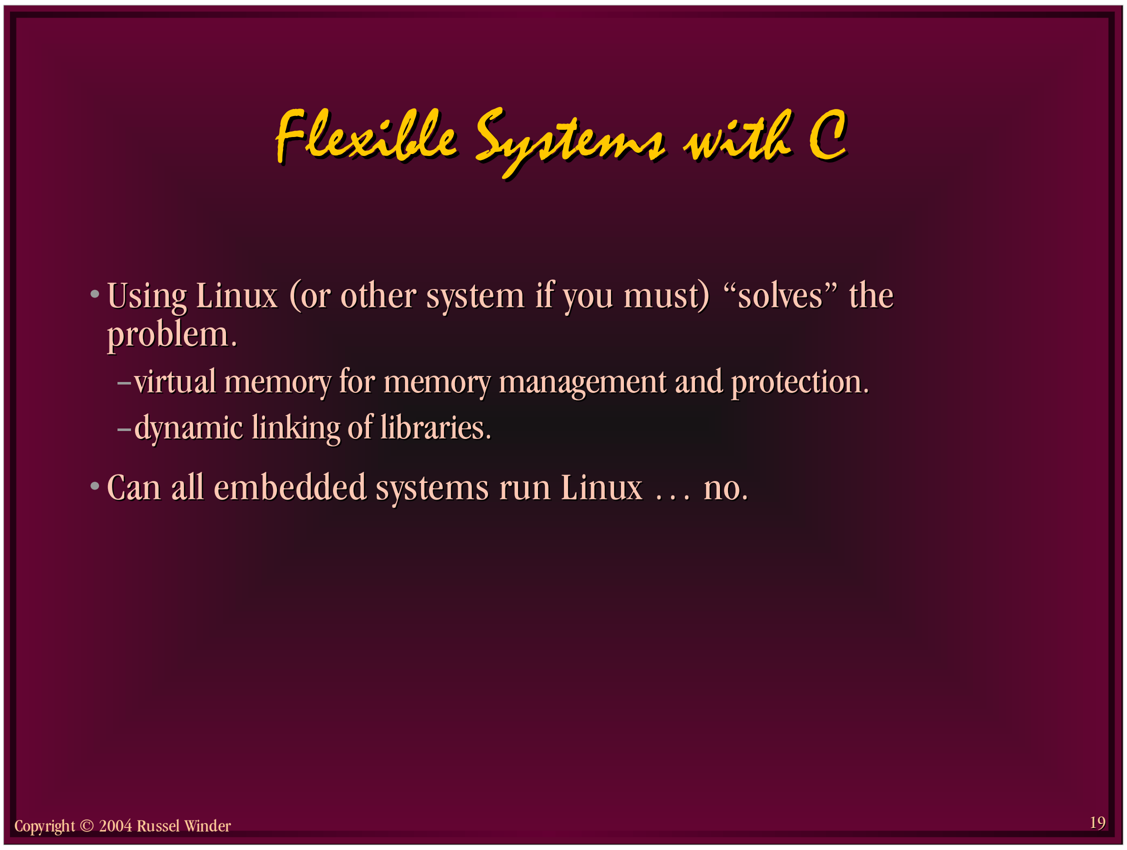
Dynamic Binding with C

- Linux and UNIX have dynamic binding via the .so file system.
- Microsoft Windows has dynamic binding via .dll file system.
- Operating system dependent.

Copyright © 2004 Russel Winder 18

Although really a statically bound language, with operating system and compiler support, C can be used to create dynamically loaded and bound systems.

The important thing here is that there is an API to be met and that the operatig system supports the features.



Flexible Systems with C

- Using Linux (or other system if you must) “solves” the problem.
  - virtual memory for memory management and protection.
  - dynamic linking of libraries.
- Can all embedded systems run Linux … no.

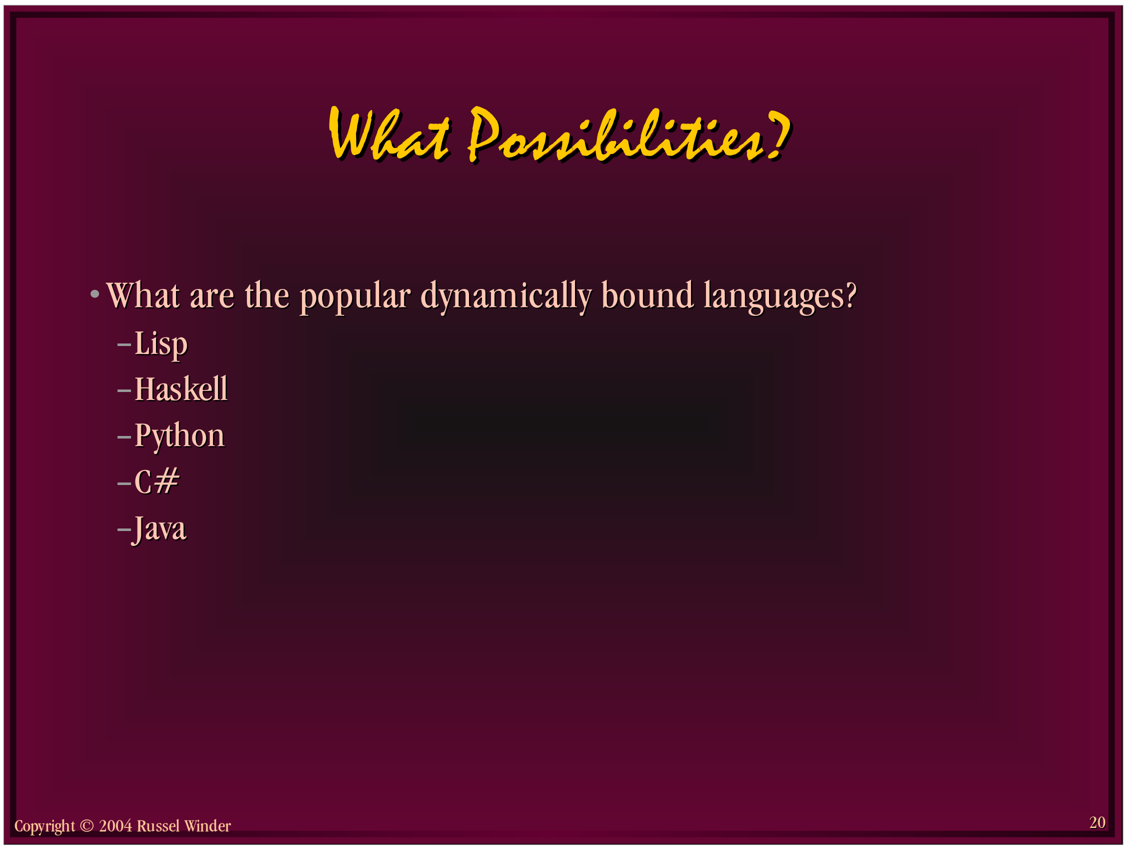
Copyright © 2004 Russel Winder 19

A lot of embedded systems do run Linux, e.g. μCLinux. For these systems C can be used to create dynamically bound systems. The use of virtual memory for processes also solves the danger of one process damaging another by inappropriate use of pointers.

Linux is relatively big though despite being small.

Standard real-time operating systems, e.g. μC/OS, generally assume a predefined system, i.e. that all tasks are known at compile time.

No standard way of doing dynamic loading and linking of C.



What Possibilities?

- What are the popular dynamically bound languages?
  - Lisp
  - Haskell
  - Python
  - C#
  - Java

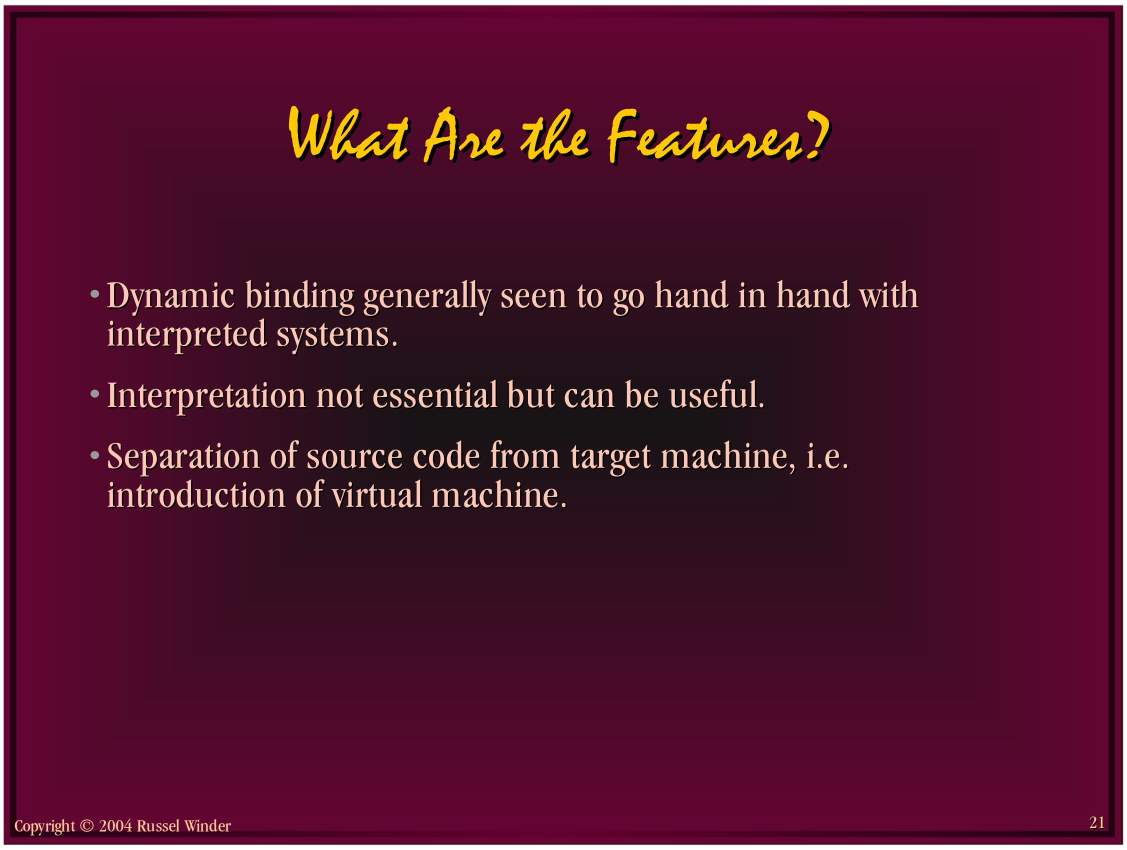
Copyright © 2004 Russel Winder 20

Lisp: An old language but still thriving. Common Lisp is huge though. However, it is certainly possible to consider use of Lisp for embedded systems since a small runtime system could be constructed. The barrier is trying to educate assembly language / C programmers into using Lisp.

Haskell: Functional programming for embedded systems. Probably a very good idea but getting assembly language and C programmers to use Haskell may be impossible. Issue would be dealing with interrupts.

Python, *C#* and *Java* are in many ways very similar. Basically Java was there first (not necessarily first to exist but first into the minds of embedded systems people).

Note the absence of *Perl*. I believe this to be an execute only language, it is impossible to read and damn near impossible to write. But it is very popular.



What Are the Features?

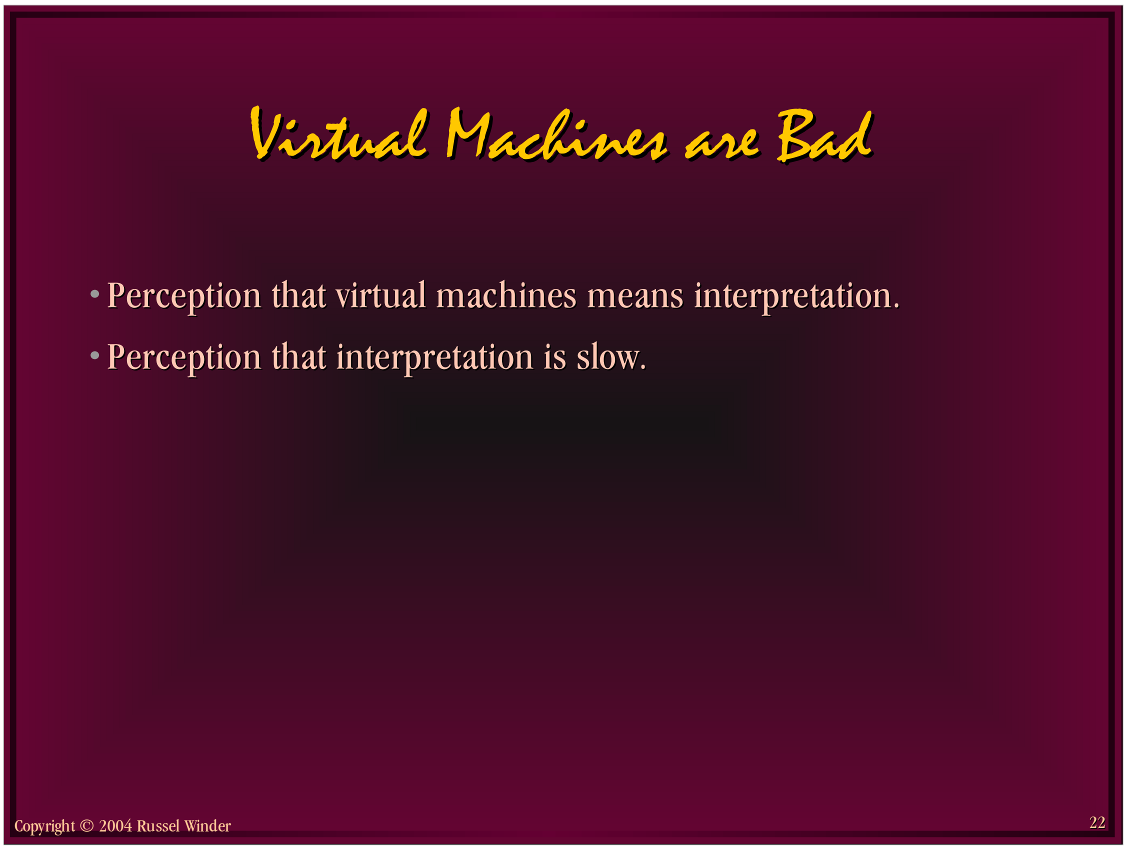
- Dynamic binding generally seen to go hand in hand with interpreted systems.
- Interpretation not essential but can be useful.
- Separation of source code from target machine, i.e. introduction of virtual machine.

Copyright © 2004 Russel Winder 21

The core issue here is that there is a runtime system of some sort that manages resources which supports the code supplied.

The supplied code is not the master controller of resources.

There is a separation of concerns, roles and responsibilities.

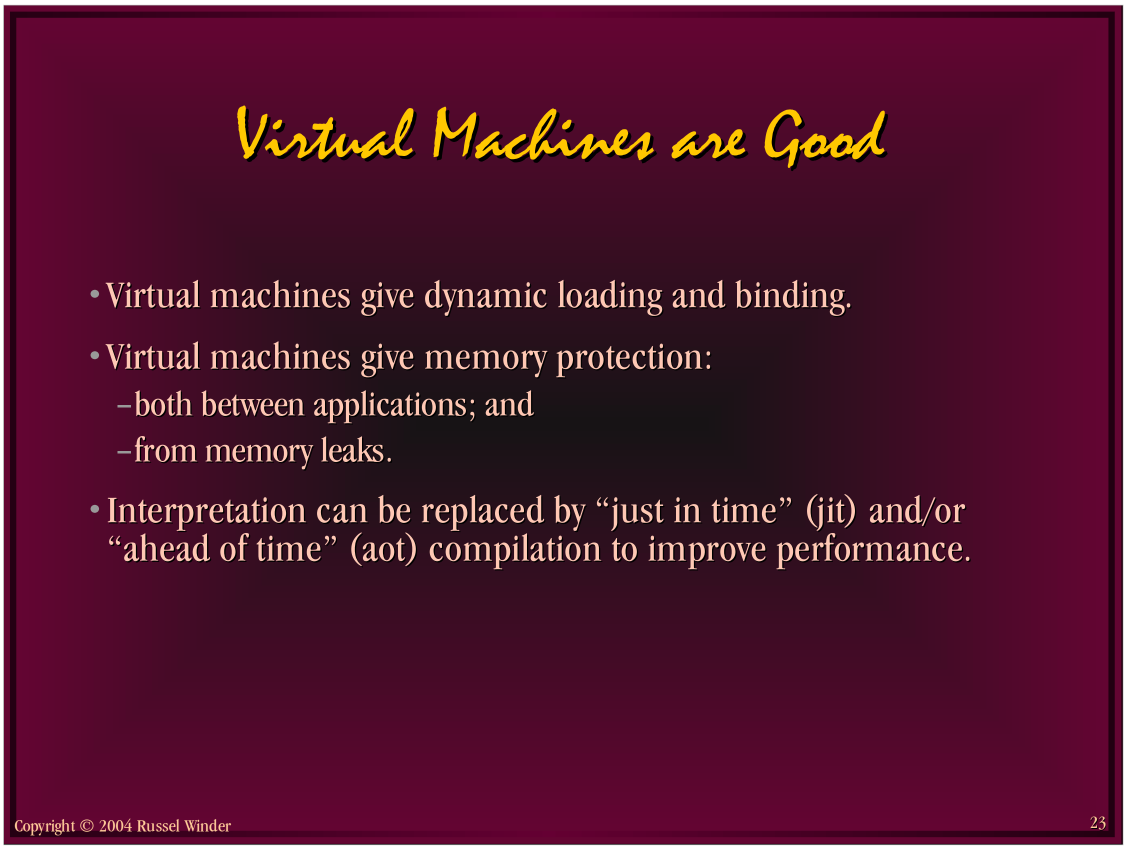


Virtual Machines are Bad

- Perception that virtual machines means interpretation.
- Perception that interpretation is slow.

Copyright © 2004 Russel Winder 22

J2SE has traditionally had a very slow start up time. Actual code execution is at a good speed, even when interpreted rather than using JIT or AOT execution. The problem was the loading of the infrastructure. With J2SE v1.5 the executable has all the infrastructure preloaded and so start up of Java application is far, far faster.



Virtual Machines are Good

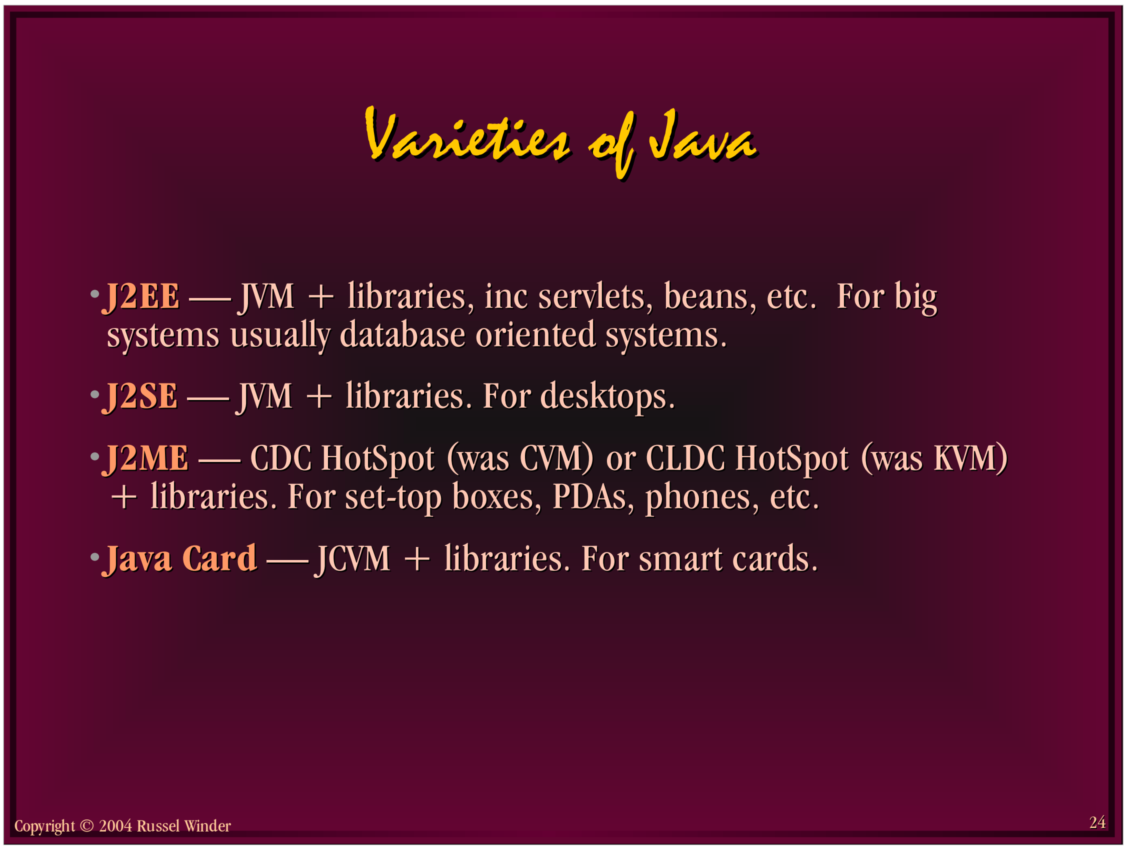
- Virtual machines give dynamic loading and binding.
- Virtual machines give memory protection:
  - both between applications; and
  - from memory leaks.
- Interpretation can be replaced by “just in time” (jit) and/or “ahead of time” (aot) compilation to improve performance.

Copyright © 2004 Russel Winder

23

The separation of resource management from application gives many many benefits. There can be no pointer problems as there are no unmanaged pointers just object references. This means no danger of memory manipulation out of bounds. Garbage collection means that if memory leaks do exist then they are less problematic.

Virtual machine concept gives a place where the “operating system” hooks to provide dynamic binding can be sited.



Varieties of Java

- **J2EE** — JVM + libraries, inc servlets, beans, etc. For big systems usually database oriented systems.
- **J2SE** — JVM + libraries. For desktops.
- **J2ME** — CDC HotSpot (was CVM) or CLDC HotSpot (was KVM) + libraries. For set-top boxes, PDAs, phones, etc.
- **Java Card** — JCVM + libraries. For smart cards.

Copyright © 2004 Russel Winder 24

J2EE is really just an extension of J2SE with extra libraries.

J2ME is a restriction of the J2SE (based on J2SE v1.1!). KVM (on it's way out) is just a restriction of the JVM.

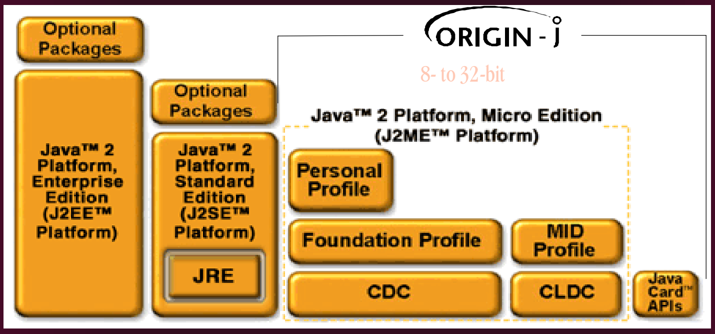
Java Card is a system invented by Schlumberger that has been coerced into being a form of Java. The bytecodes of the JCVM are not the bytecodes of the JVM. Java Card requires CAP files not class files for downloading. CAP files are internally linked files, think JEFF (J execution file format, a partially linked multi-class file format created by the J Consortium).

RTJ is really an extension of J2SE for handling real time concepts. It is a Java realization of all the feature normally associated with a real-time kernel.

Gosh that sounds rather useful for Embedded Systems. Or is it?

I will focus on J2ME and Java Card as I am interested in systems with serious resource constraints.

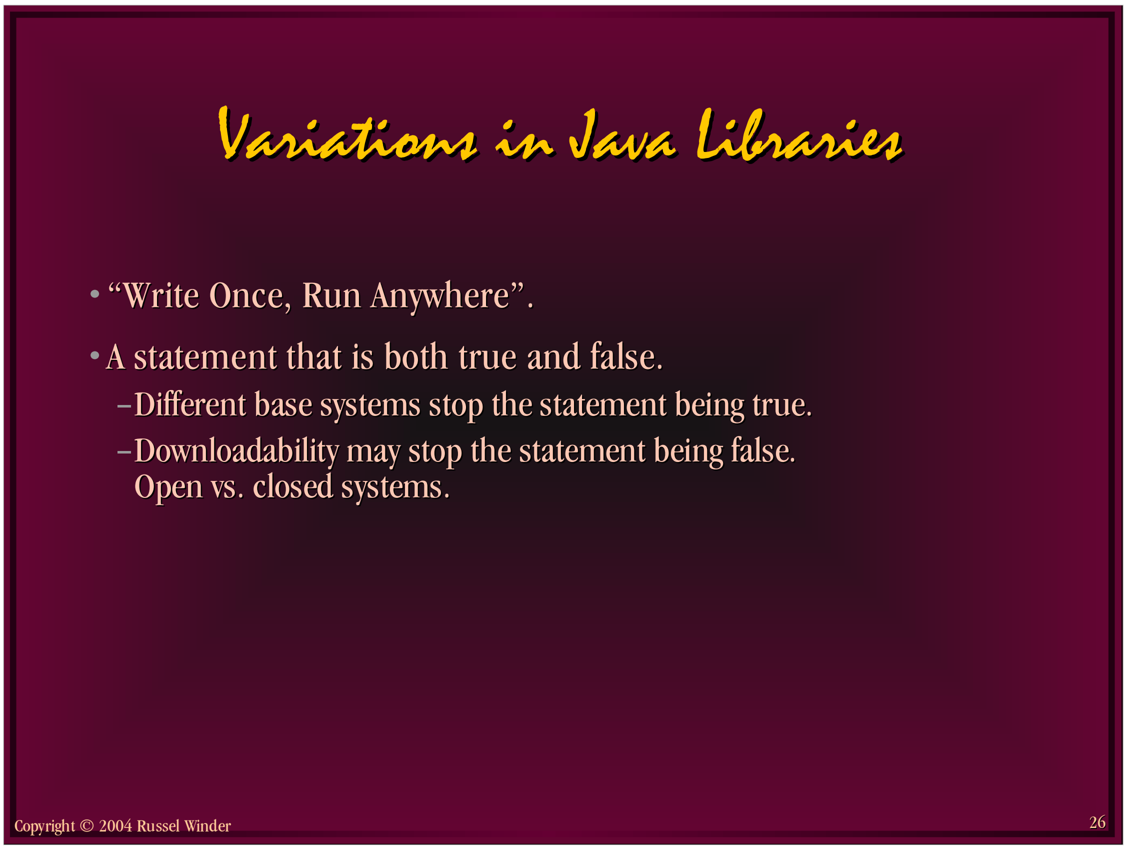
The Varieties of Java


Optional Packages
Java™ 2 Platform, Enterprise Edition (J2EE™ Platform)
Optional Packages
Java™ 2 Platform, Standard Edition (J2SE™ Platform)
JRE
ORIGIN - j
8- to 32-bit
Java™ 2 Platform, Micro Edition (J2ME™ Platform)
Personal Profile
Foundation Profile
MID Profile
CDC
CLDC
Java Card™ APIs

Enterprise Systems	Desktop Systems	Mobile Devices & Embedded Systems	Smart Cards
32- to 64-bit	32-bit	16- to 32-bit	8- to 32-bit

Copyright © 2004 Russel Winder

Not sure where the original diagram comes from, it is probably a Sun diagram.



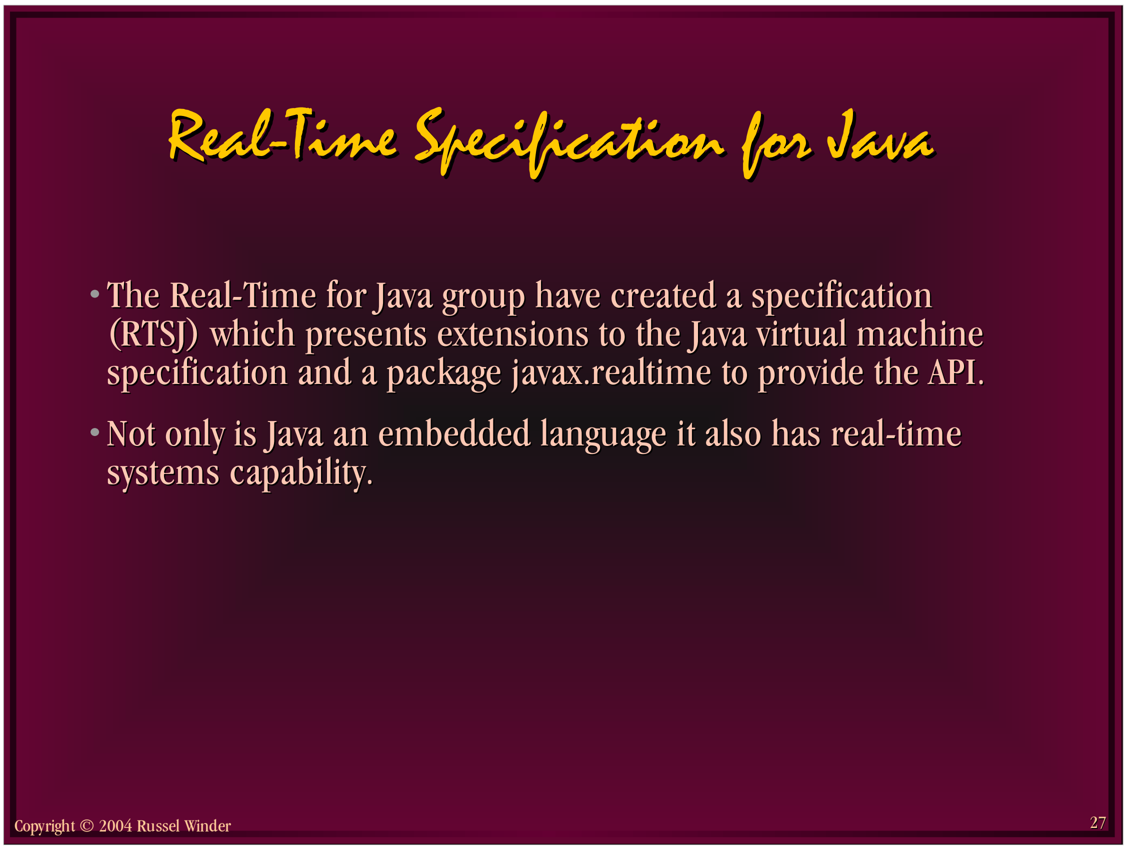
Variations in Java Libraries

- “Write Once, Run Anywhere”.
- A statement that is both true and false.
  - Different base systems stop the statement being true.
  - Downloadability may stop the statement being false. Open vs. closed systems.

Copyright © 2004 Russel Winder

The issue here is that the virtual machine concept gives the run anywhere idea but only if the support / system classes are present. This requires standardized runtime environments and/or the ability to necessary load classes on demand.

As we shall see the WORA concept is not really true but can be in segments of the overall Java field.



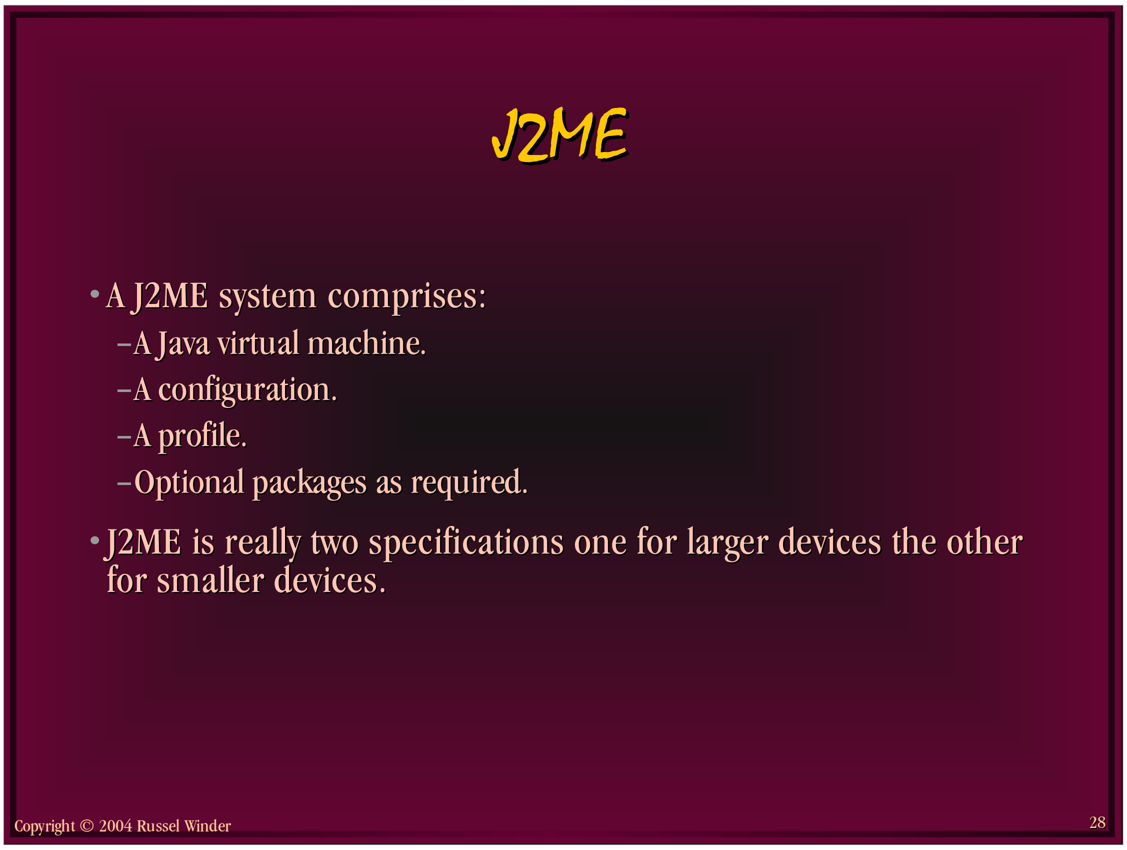
Real-Time Specification for Java

- The Real-Time for Java group have created a specification (RTSJ) which presents extensions to the Java virtual machine specification and a package javax.realtime to provide the API.
- Not only is Java an embedded language it also has real-time systems capability.

Copyright © 2004 Russel Winder 27

RTSJ basically models in a Java context all the traditional features of a real-time runtime system: thread management, memory management, scheduling, etc.

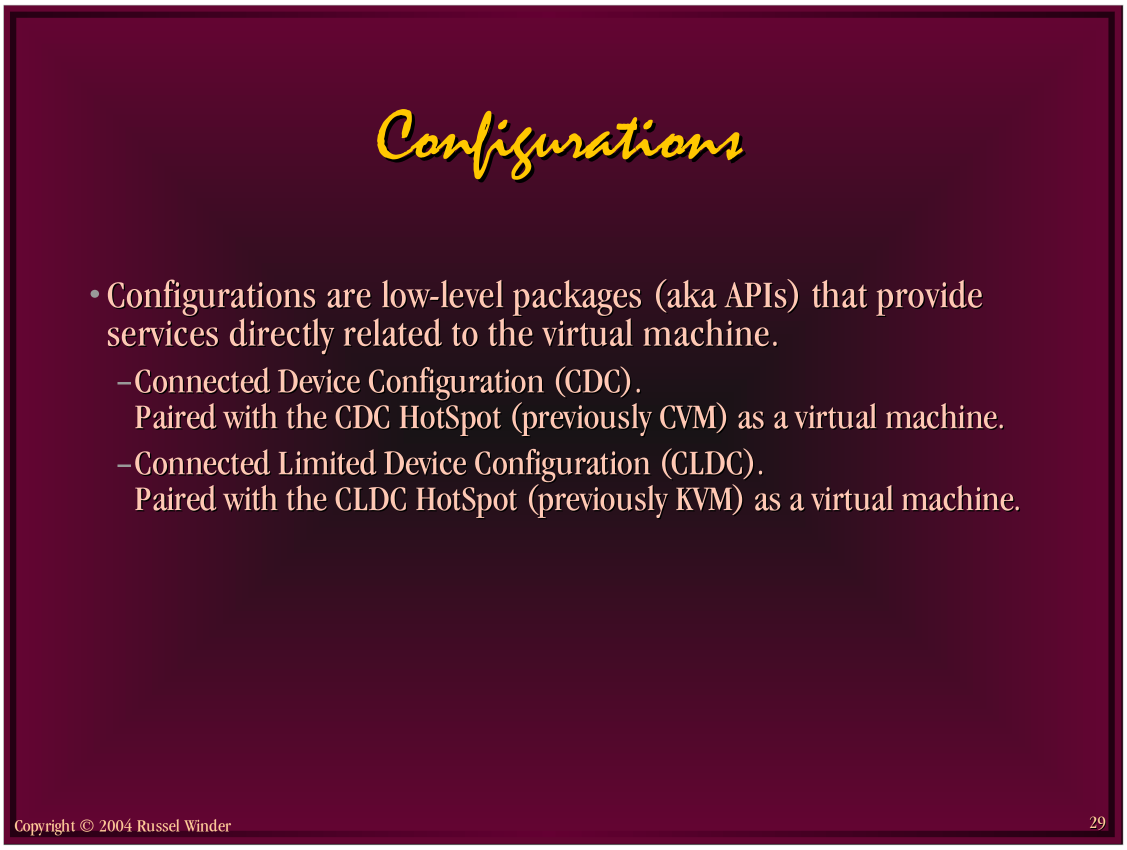
In principal this can be applied to any of of the Java 2 Platform specifications but in the context of embedded system J2ME appears to be the best partner.



J2ME

- A J2ME system comprises:
  - A Java virtual machine.
  - A configuration.
  - A profile.
  - Optional packages as required.
- J2ME is really two specifications one for larger devices the other for smaller devices.

Copyright © 2004 Russel Winder 28



Configurations

- Configurations are low-level packages (aka APIs) that provide services directly related to the virtual machine.
  - Connected Device Configuration (CDC).
    Paired with the CDC HotSpot (previously CVM) as a virtual machine.
  - Connected Limited Device Configuration (CLDC).
    Paired with the CLDC HotSpot (previously KVM) as a virtual machine.

Copyright © 2004 Russel Winder 29

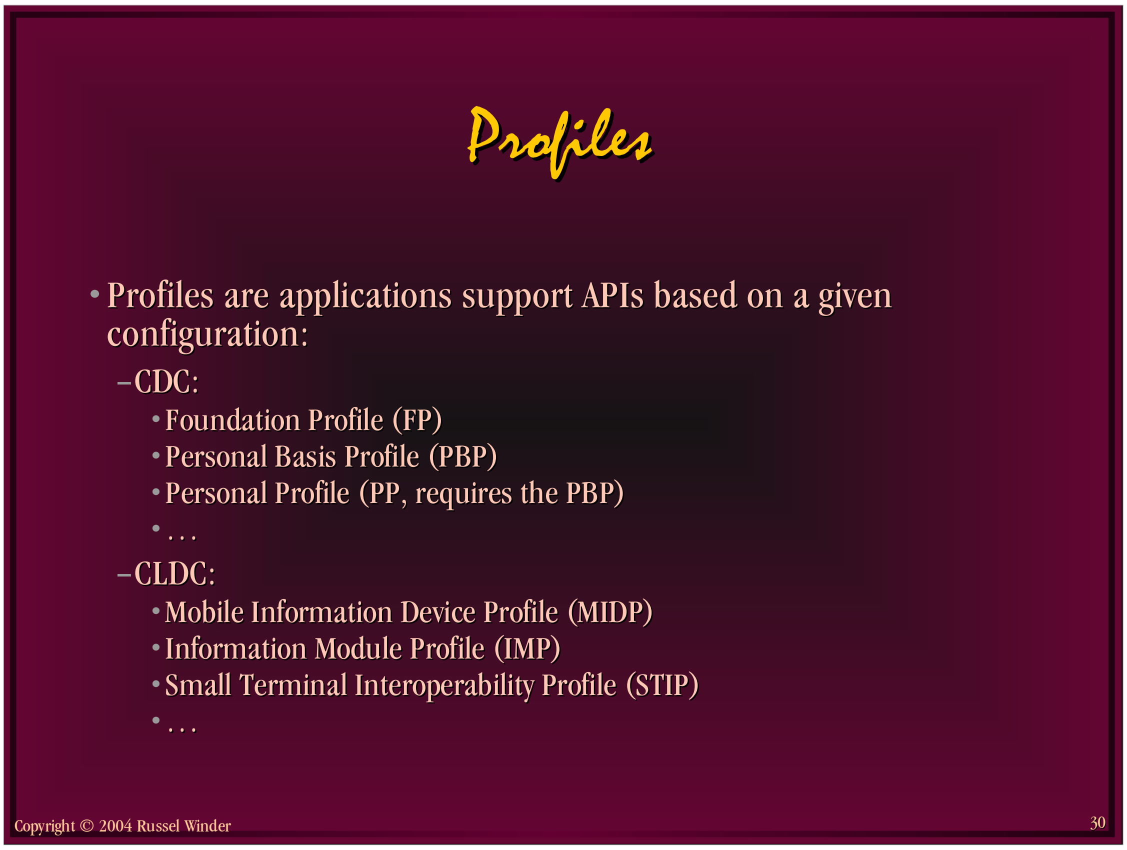
CDC 1.0.1 based on J2SE 1.3.1.

CDC 1.1 based on J2SE 1.4.1.

CDC HotSpot is the new name for CVM an implementation of the JVM optimized for devices smaller that workstations.

CLDC 1.0 based on J2SE 1.1.7.

CLDC HotSpot is new replacing the KVM an implementation of a restriction JVM.



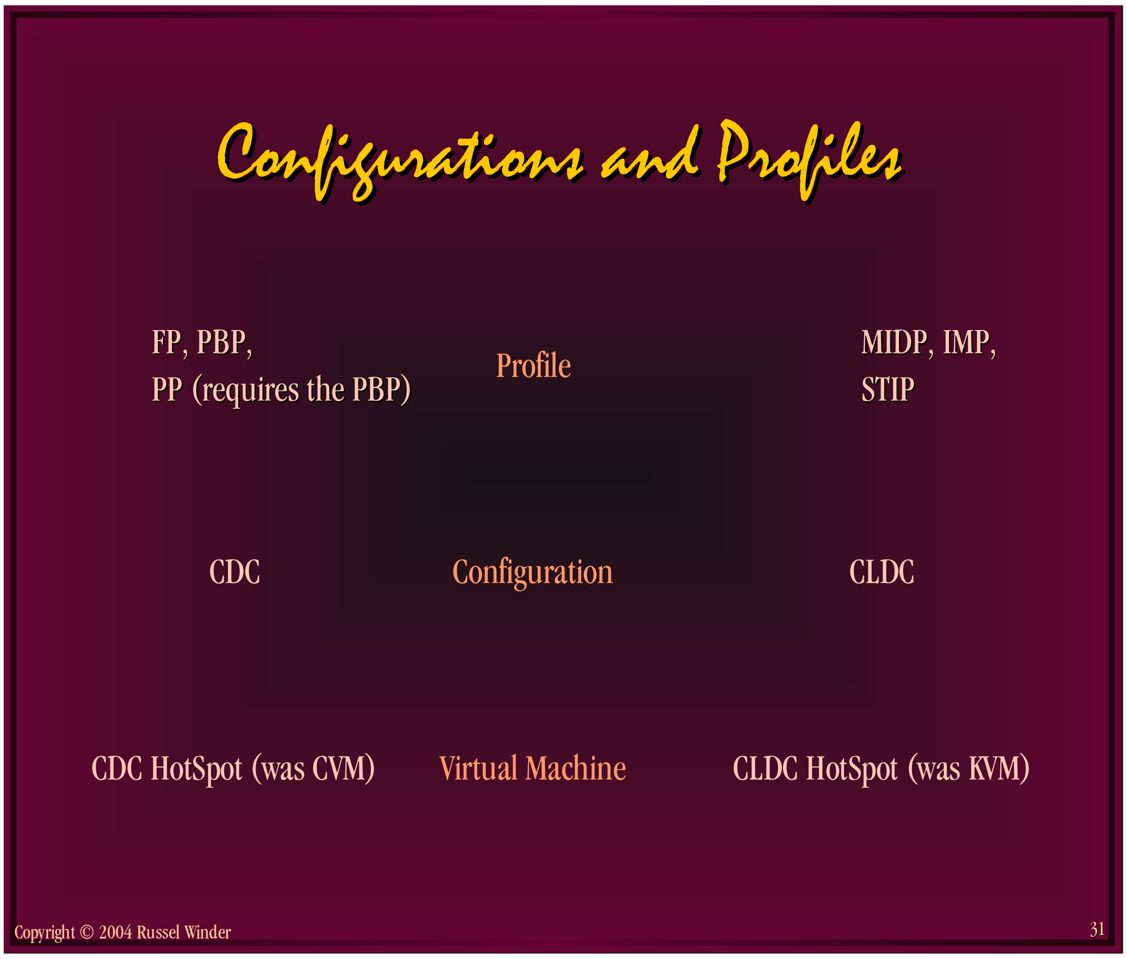
Profiles

- Profiles are applications support APIs based on a given configuration:
  - CDC:
    - Foundation Profile (FP)
    - Personal Basis Profile (PBP)
    - Personal Profile (PP, requires the PBP)
    - . . .
  - CLDC:
    - Mobile Information Device Profile (MIDP)
    - Information Module Profile (IMP)
    - Small Terminal Interoperability Profile (STIP)
    - . . .

Copyright © 2004 Russel Winder 30

Notice the bifurcation is not reified. A J2ME system is either a CDC or a CLDC system and the profiles usable are different.

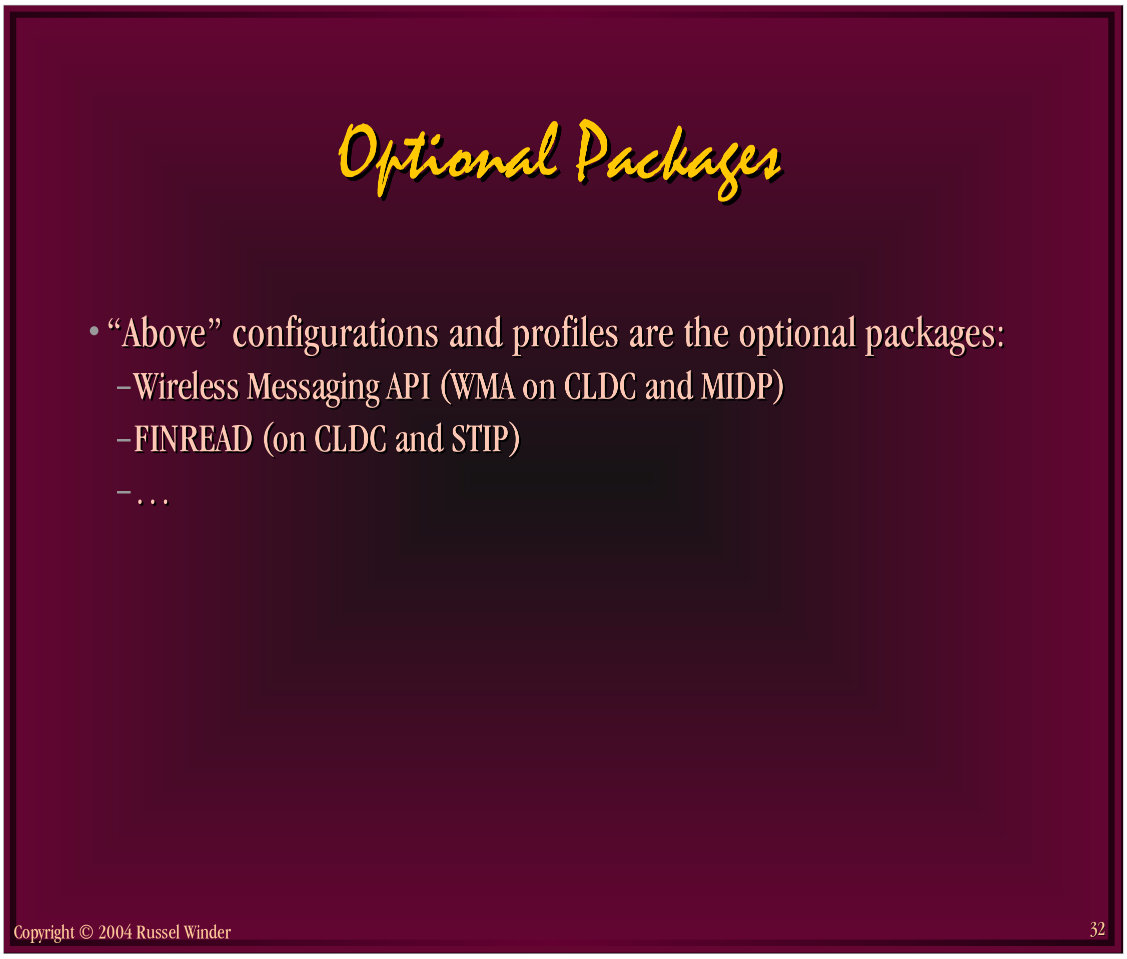
I have no experience of CDC so stick to thinking about CLDC. This is not unreasonable for me as CDC is really for small computers and/or big embedded systems and I am more interested in really small embedded devices.



Configurations and Profiles

FP, PBP, PP (requires the PBP)	Profile	MIDP, IMP, STIP
CDC	Configuration	CLDC
CDC HotSpot (was CVM)	Virtual Machine	CLDC HotSpot (was KVM)

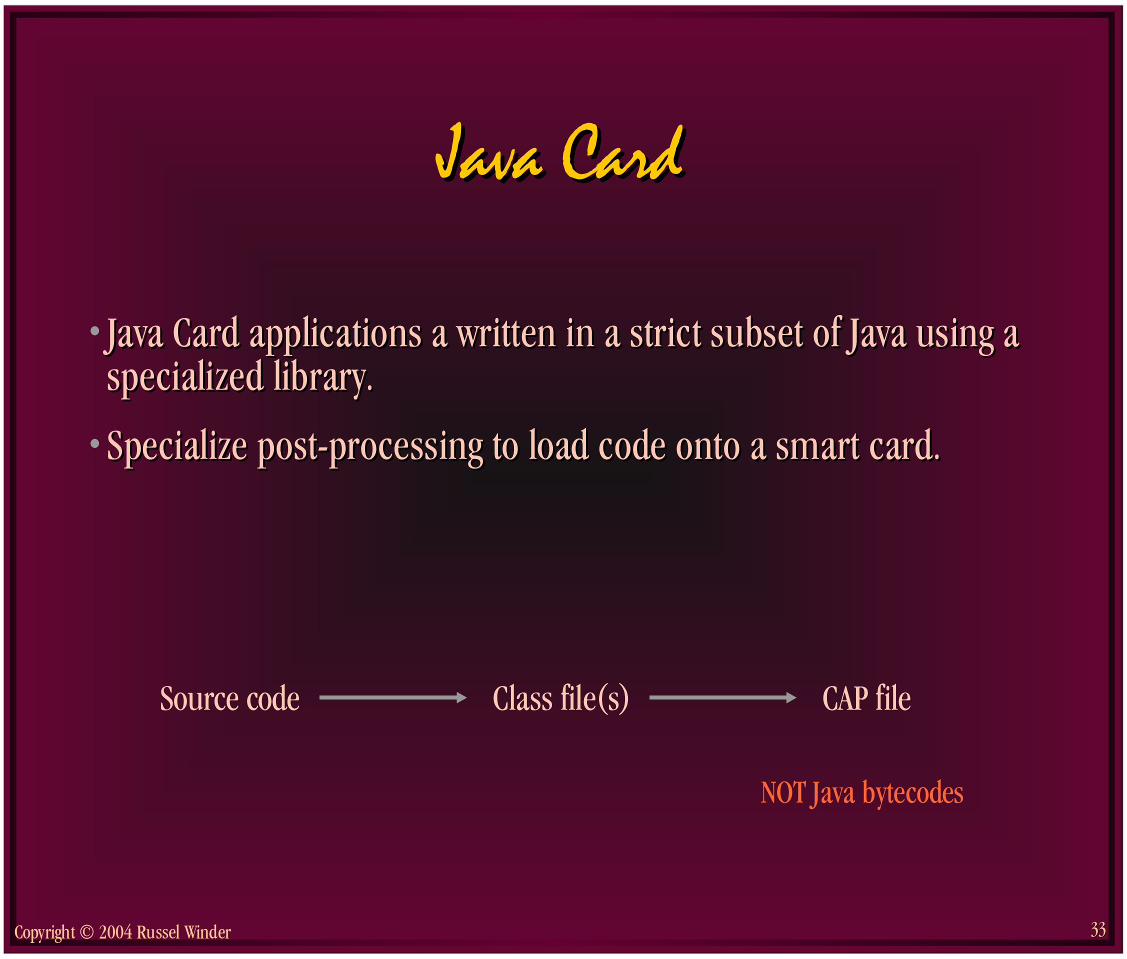
Copyright © 2004 Russel Winder



Optional Packages

- "Above" configurations and profiles are the optional packages:
  - Wireless Messaging API (WMA on CLDC and MIDP)
  - FINREAD (on CLDC and STIP)
  - . . .

Copyright © 2004 Russel Winder 32



Java Card

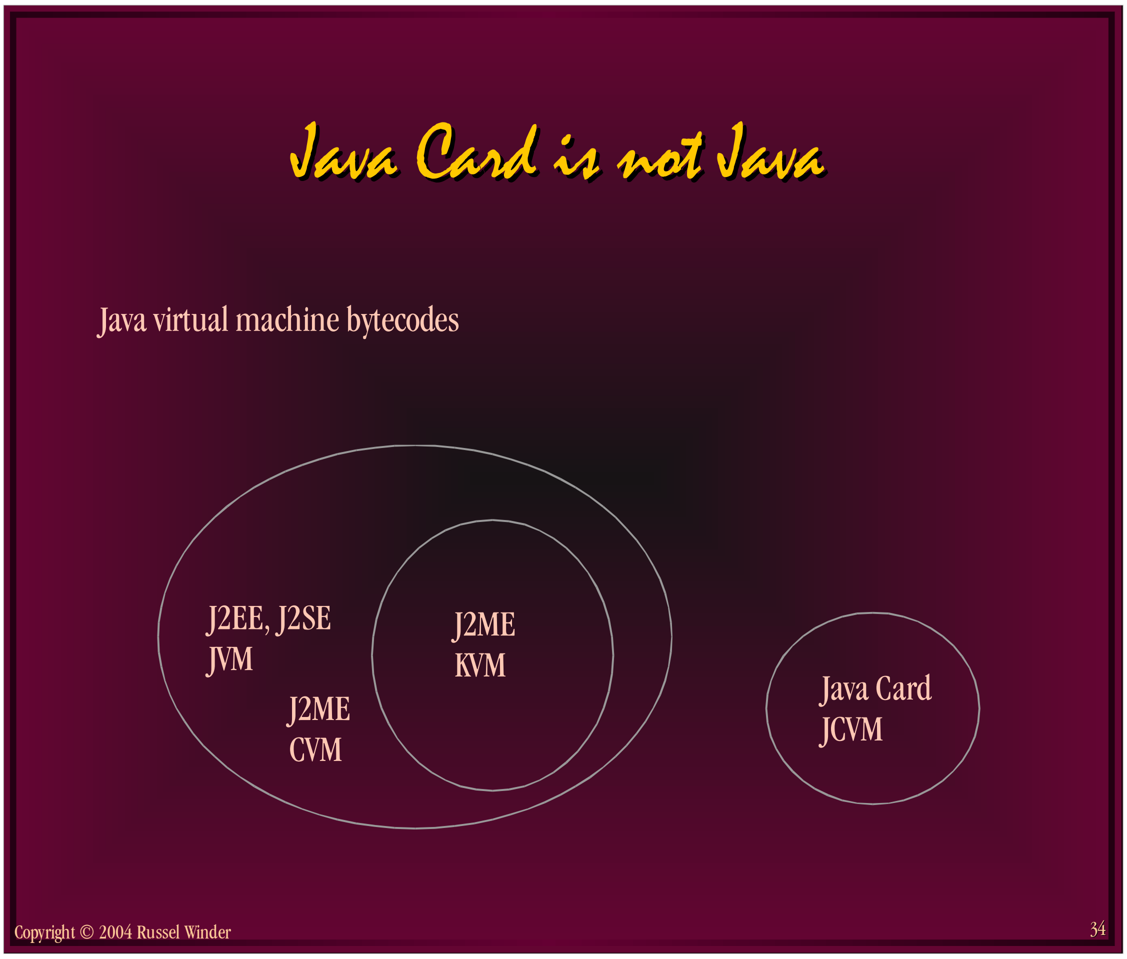
- Java Card applications a written in a strict subset of Java using a specialized library.
- Specialize post-processing to load code onto a smart card.

$$\begin{tikzcd}\text{Source code} \arrow[r] & \text{Class file(s)} \arrow[r] & \text{CAP file}\end{tikzcd}$$

NOT Java bytecodes

Copyright © 2004 Russel Winder 33

Although the soruce code of a Java Card system is Java the final executable is not.

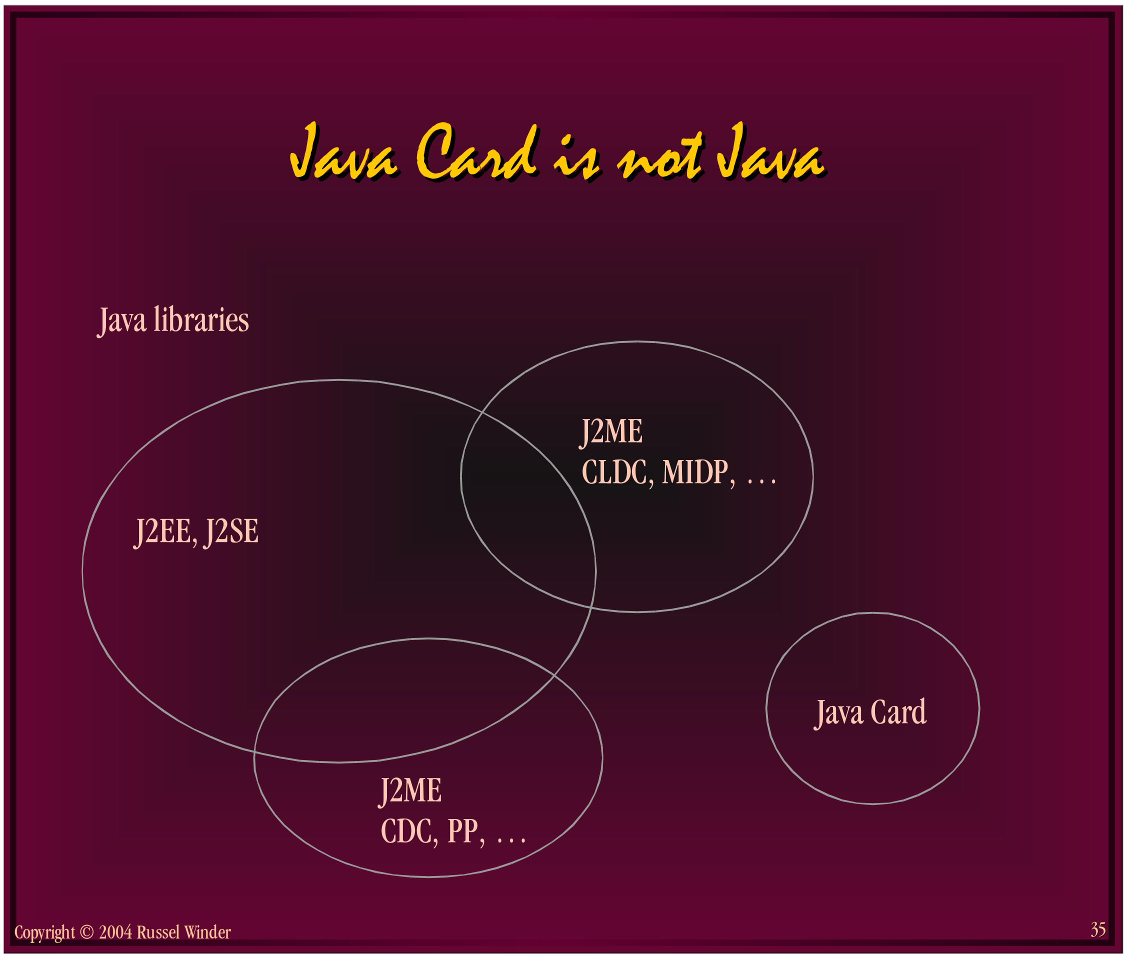


Java Card is not Java

Java virtual machine bytecodes

Copyright © 2004 Russel Winder 34

Java Card does not implement the bytecodes of J2EE, J2SE and J2ME (both CDC and CLDC versions).

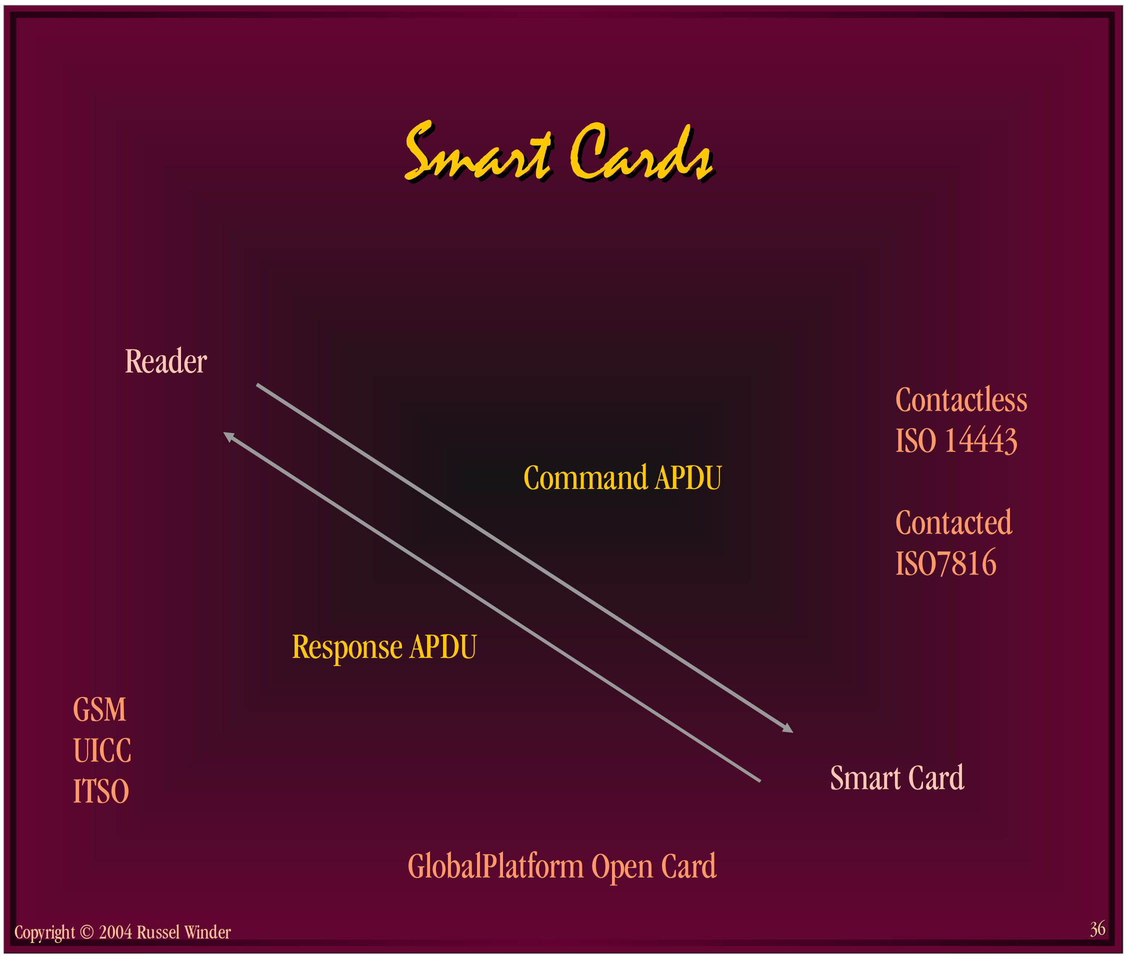


Java Card is not Java

Copyright © 2004 Russel Winder

The Java Card libraries have no overlap with any of the J2EE, J2SE or J2ME (both CDC and CLDC) libraries.

Java Card does not even share any of the concepts of the RTSJ.



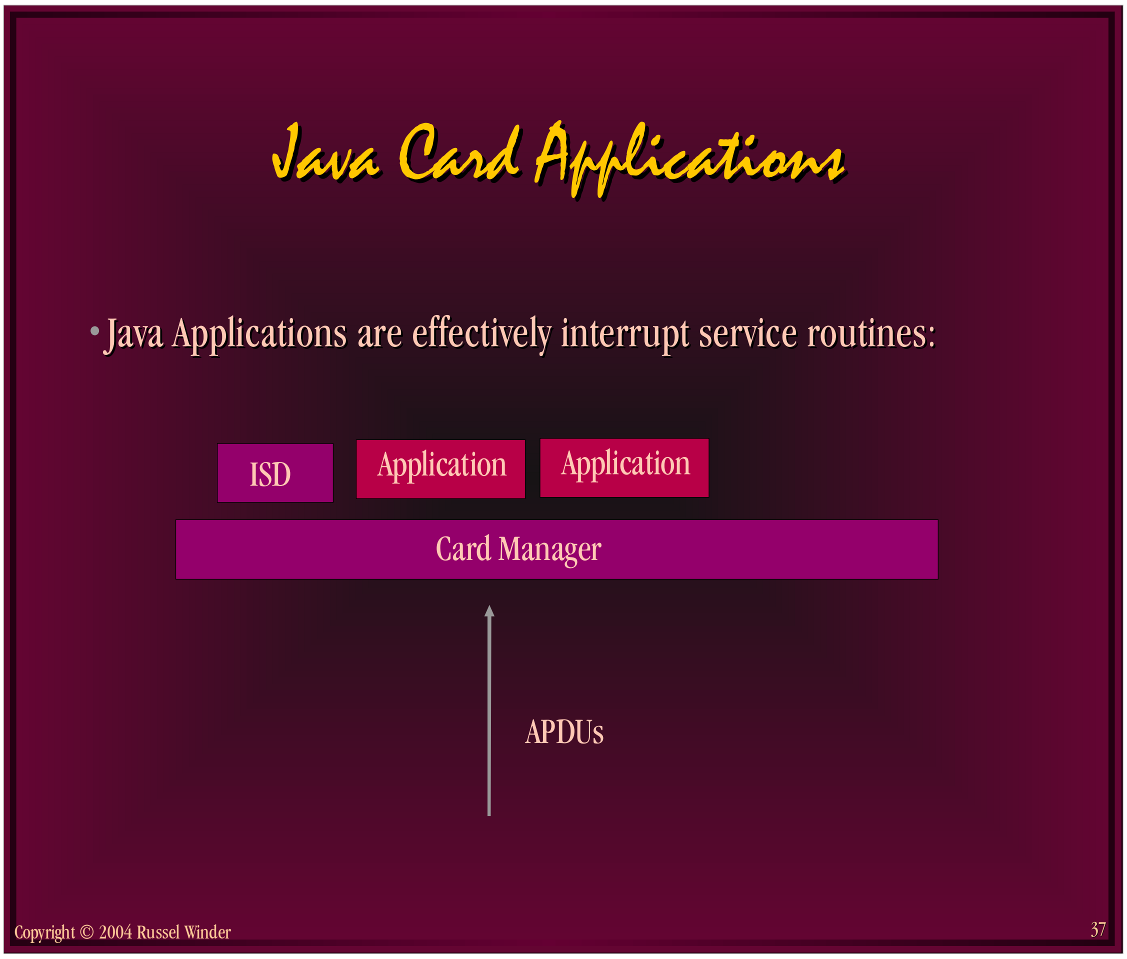
Smart Cards

Reader and smart card communicate over a serial line. 9.6Kb–115Kb is contacted and 400Kb–800Kb if contactless.

There are a number of protocols, T=0 and T=1 for contacted, T=CL for contactless. Protocols are half-duplex transmissions of packets (TPDUs and APDUs).

Soon there will be USB.

A variety of variations on the standards.



Java Card Applications

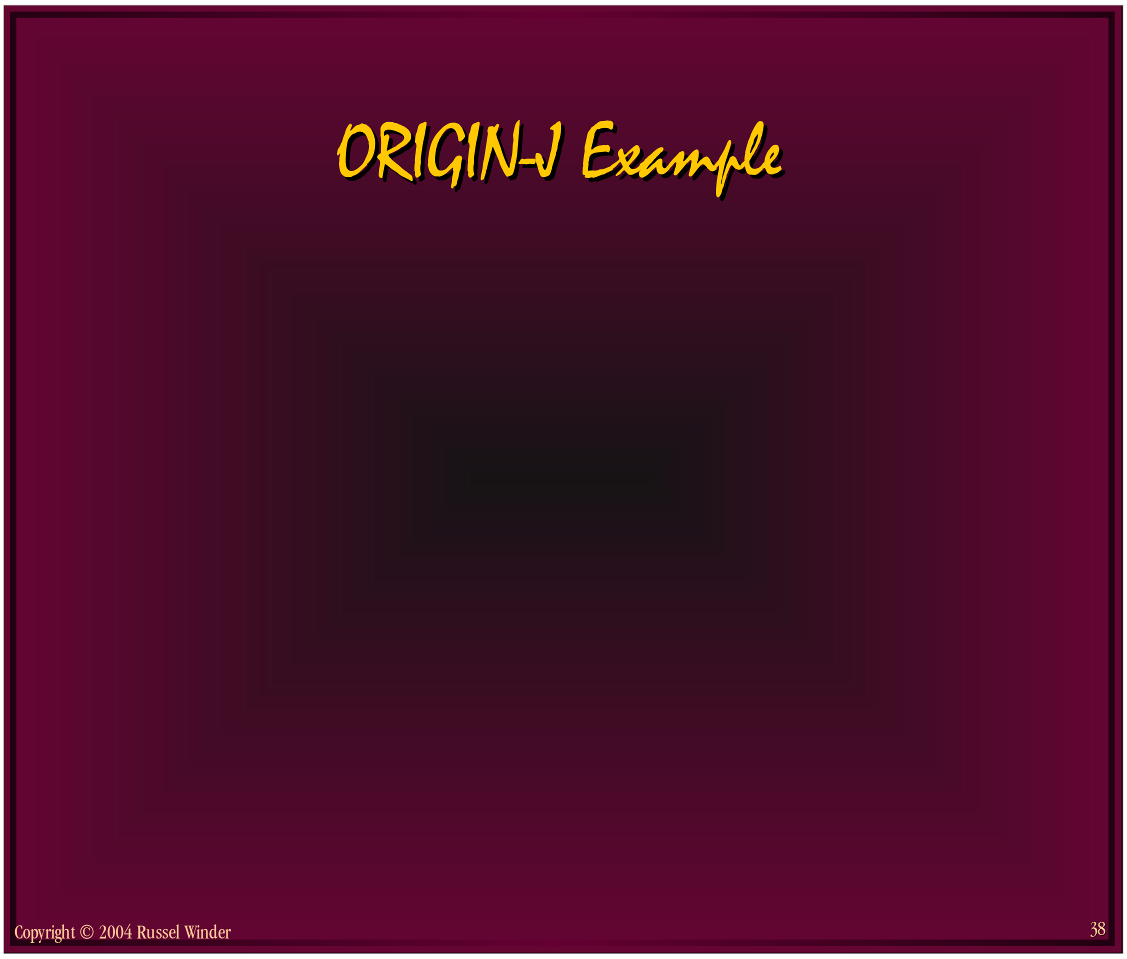
- Java Applications are effectively interrupt service routines:

ISD
Application
Application
Card Manager
APDUs

Copyright © 2004 Russel Winder 37

A smart card has a runtime system which is operating system plus GlobalPlatform compliant security.

Each application is a service. APDU arrives causes an event (interrupt) which is then serviced by the card maanger choosing which application.



ORIGIN-J Example

Copyright © 2004 Russel Winder 38

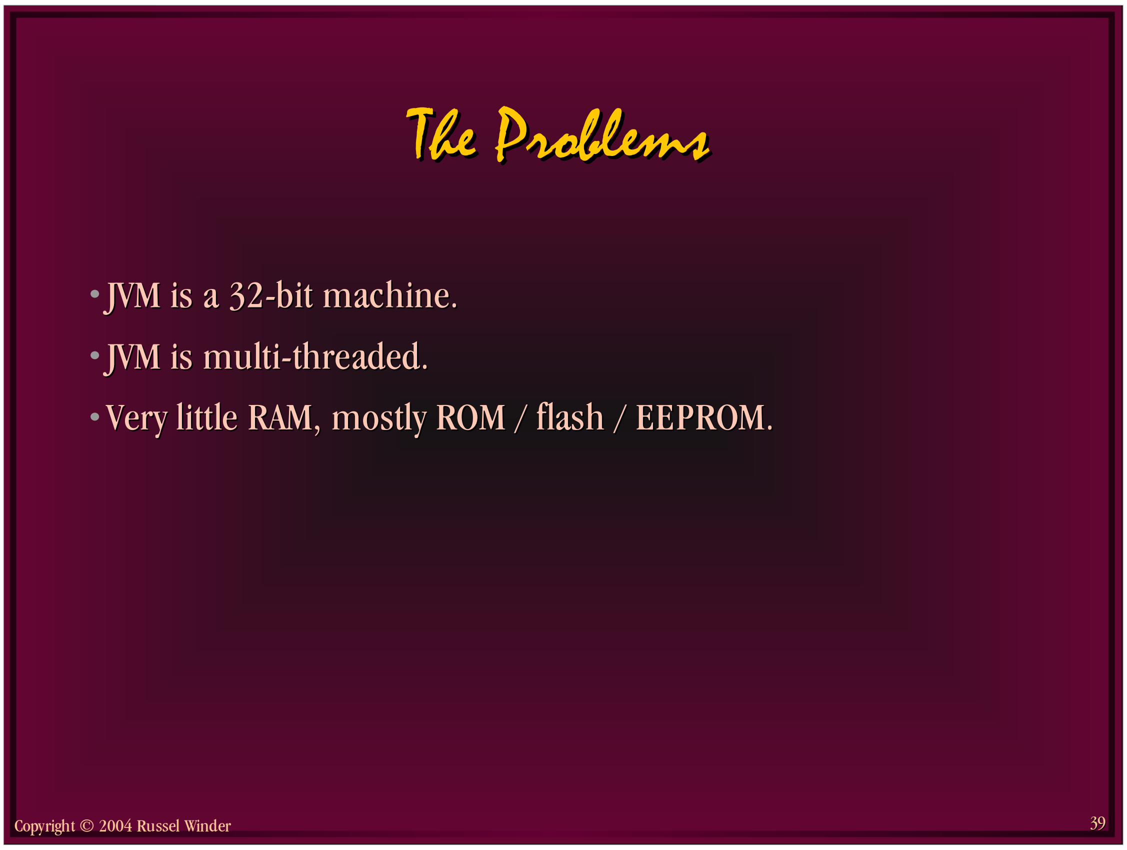
Show the ORIGIN-J Purse application to show some code:

ISR like nature of the code.

APDU processor.

Multi-threading available. See one of the other ORIGIN-J applications.

ORIGIN-J has the ability to support threads Java Card doesn't.



The Problems

- JVM is a 32-bit machine.
- JVM is multi-threaded.
- Very little RAM, mostly ROM / flash / EEPROM.

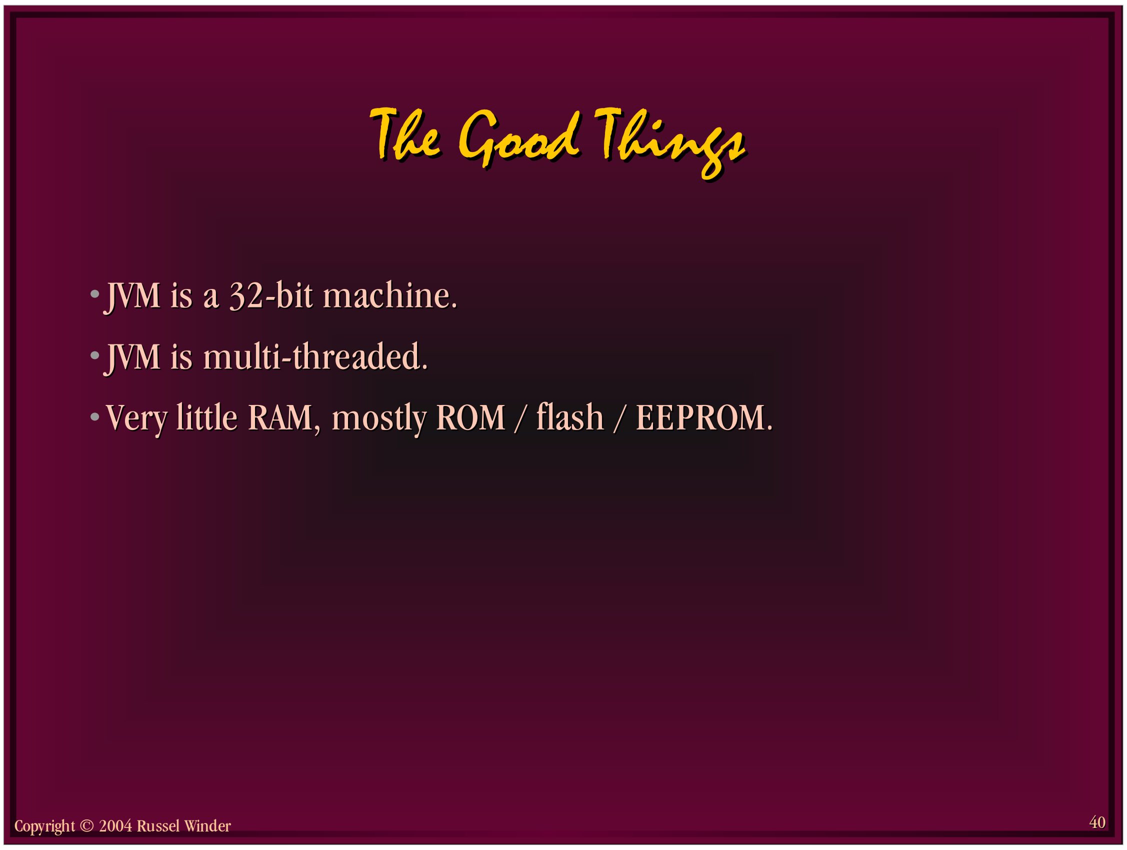
Copyright © 2004 Russel Winder 39

Working with 8- and 16-bit processors is hard because of the 32-bit nature of the JVM.

32-bit procesors are coming but only in high-end. 8-bit is still the most important platform.

Multi-threading on a 3.75MHz 8-bit processor is not totally fun. Most processors are now 60+MHz.

How do you run a virtual machine with so little RAM?



The Good Things

- JVM is a 32-bit machine.
- JVM is multi-threaded.
- Very little RAM, mostly ROM / flash / EEPROM.

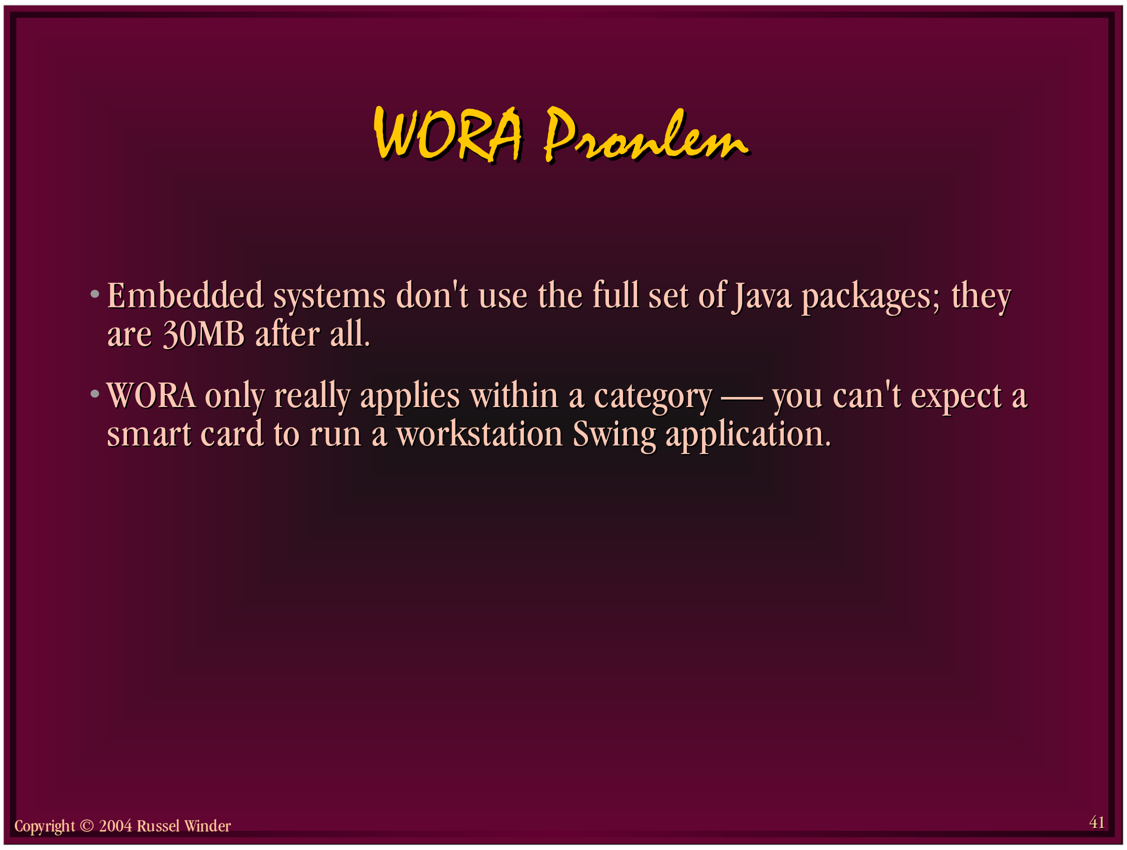
Copyright © 2004 Russel Winder 40

The issue is again that the virtual machine means the platform is standard and the issue is the porting of the virtual machine

Don't need to think about persistent objets if the state is held in EEPROM or flash as the memory itself is persistent — changes the whole perspective on memory.

BUT

EEPROM write take 1-5ms.



WORA Pronlem

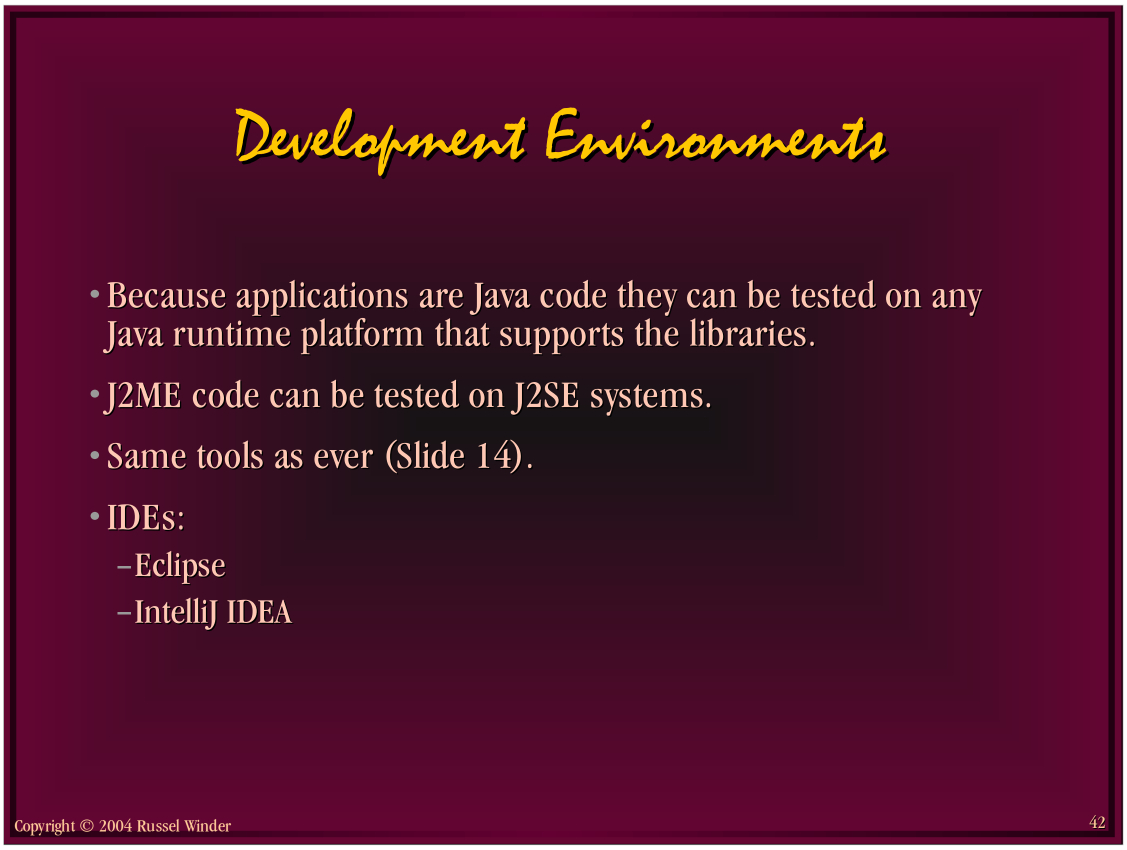
- Embedded systems don't use the full set of Java packages; they are 30MB after all.
- WORA only really applies within a category — you can't expect a smart card to run a workstation Swing application.

Copyright © 2004 Russel Winder 41

Write Once, Run Anywhere.

The moral here is to apply realism to a catch phrase and not lambast Sun too much for being overly extravagant in their choice of catch phrase.

Joke probably only comprehensible to scousers – as in “worapins next”.



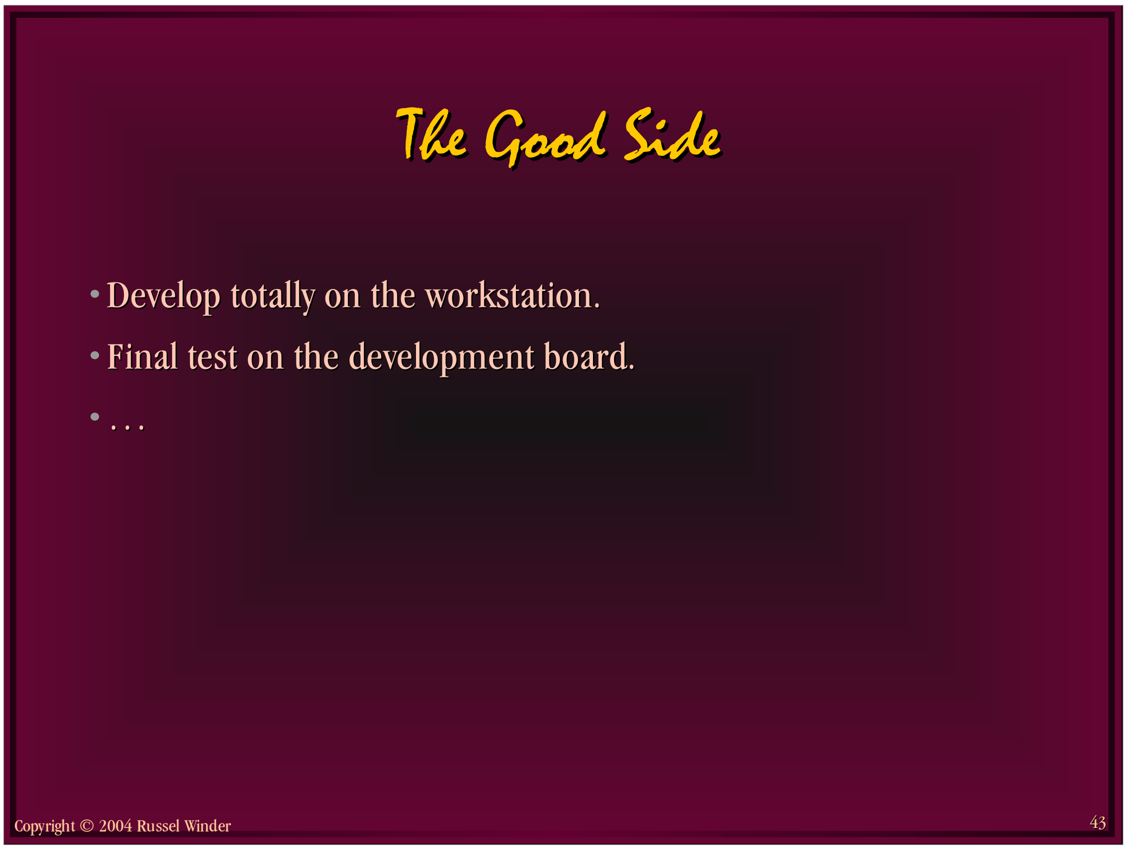
Development Environments

- Because applications are Java code they can be tested on any Java runtime platform that supports the libraries.
- J2ME code can be tested on J2SE systems.
- Same tools as ever (Slide 14).
- IDEs:
  - Eclipse
  - IntelliJ IDEA

Copyright © 2004 Russel Winder
42

The issue that all development can be done on the workstation is a real boon.

Applications can be developed and tested on workstations.



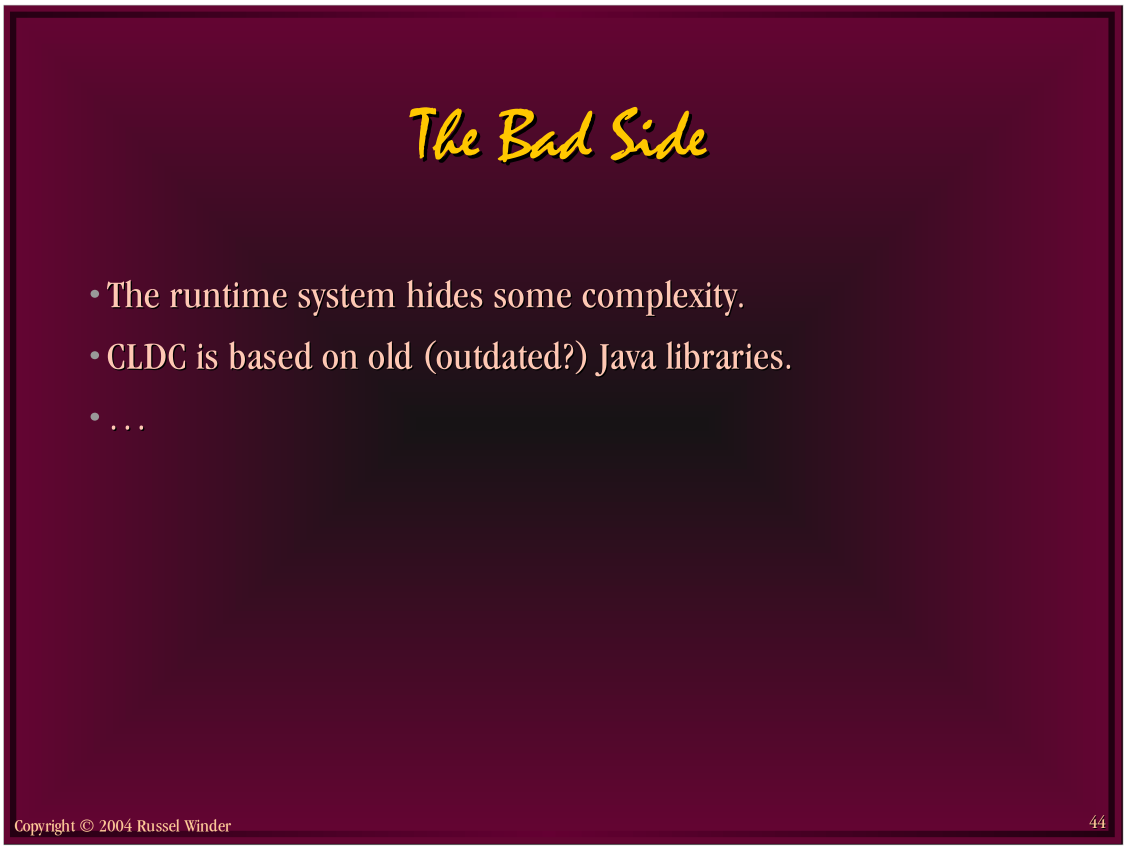
The Good Side

- Develop totally on the workstation.
- Final test on the development board.
- . . .

Copyright © 2004 Russel Winder 43

I cannot over-emphasize how good the virtual machine concept is in giving this application development approach.

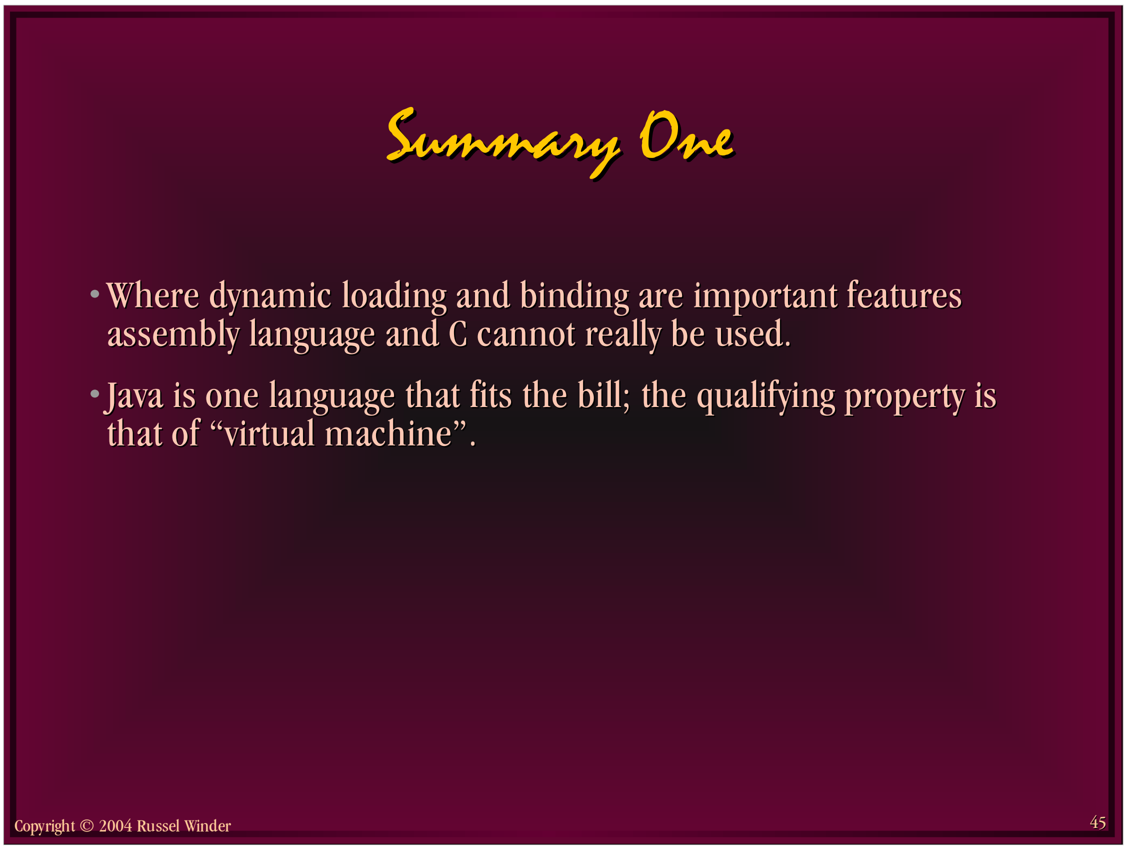
So I shall do it again!



The Bad Side

- The runtime system hides some complexity.
- CLDC is based on old (outdated?) Java libraries.
- . . .

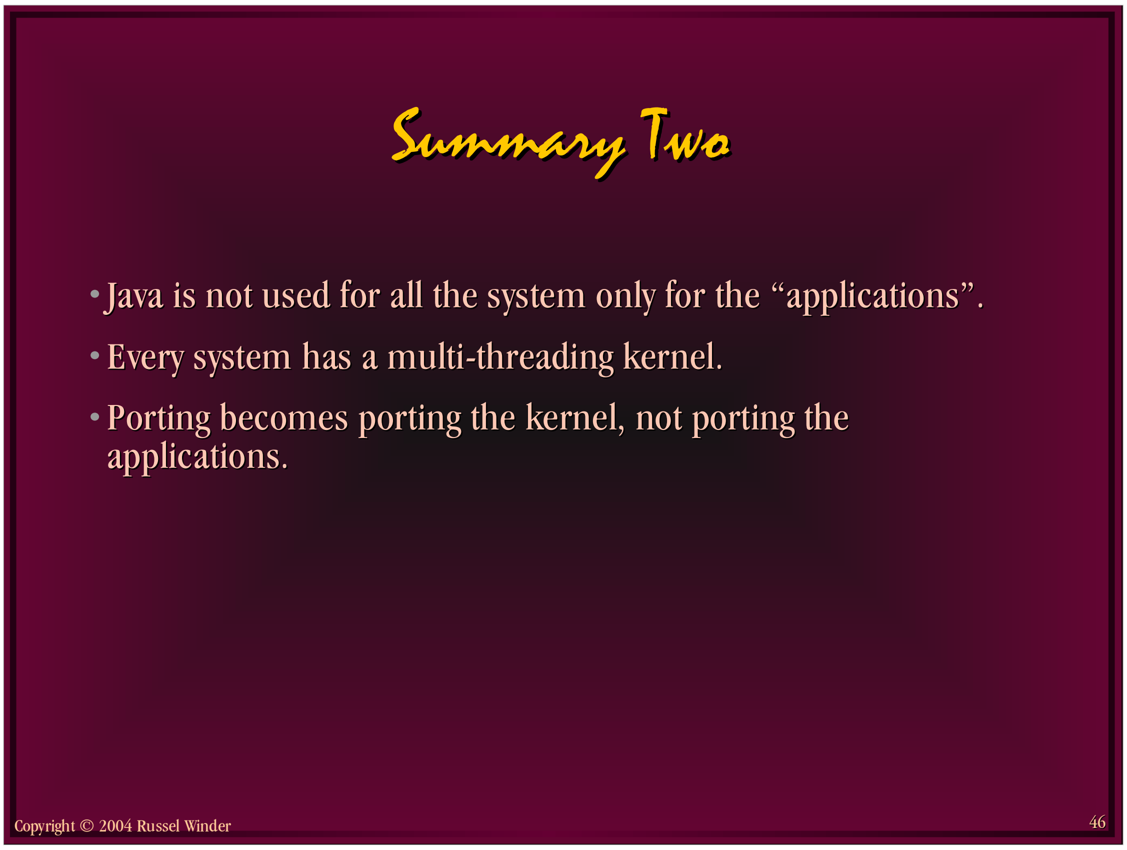
Copyright © 2004 Russel Winder 44



Summary One

- Where dynamic loading and binding are important features assembly language and C cannot really be used.
- Java is one language that fits the bill; the qualifying property is that of “virtual machine”.

Copyright © 2004 Russel Winder



Summary Two

- Java is not used for all the system only for the “applications”.
- Every system has a multi-threading kernel.
- Porting becomes porting the kernel, not porting the applications.

Copyright © 2004 Russel Winder