Java: The Good Parts

+ stuff, - rare
indep pieces stay indep
object code = portability

The Type System

Every object is a class
internal data layout
+ methods
Extend other classes
Override inheritance
Root: class Object
Single inheritance
interfaces: directed graph
class interface
abstract class
type safe
interface make abstraction concrete
✓ all signatures are OK
Interface = UI for programmers
Clarify semantics
Δ implementation, ✗ interface
test class w/interface
Sets of interrelated operations
minimum of what is done
Classes are related data
used to Δ data
Methods declared in interface
Forms a semantic unit
Extend interface to refine meaning
If system built to last,
will have methods in interfaces

Exceptions

p23

extends Throwable
Alternative return values
Reenter at d.f spot
Bad things happen
Can deal with it
Separate main code from error code
Some exceptions can be corrected

Packages

Import protection mechanism
Creates a namespace
Same name = same thing
Prob of unique identifier
Access control: public/private/protected
No label? Package
IDE good for refactoring
File system strange
it/goes/Comm/here
Cheap DB

Garbage Collection

p49

No explicit memory management
Large # of bugs
C/C# must explicitly free up
No pointers to avoid garbage issues
1 way to access an object
No pointer arithmetic
Need d.f between objects and other data
Most objects short lived
others older
2 heaps, young & old
Can move around by JVM
Objects ref other objects create probs
File handles + sockets scares
close when done
Finalizer might get run, might not
finalizer = bad design

JVM

p63

Abstraction
Higher than OS, hardware
Object code between compiler & JVM
Not novel Pascal in 78
Other languages → JVM (JyThon) Scala
Layer of security
No jumps to random parts of code
JVM separates you from chips/OSes
No #defines & ifdef
Write once, tune everywhere
Graphics most sensitive
File.separator
Java -DClass.var=32

Javadoc

embedded w/code
Δ when code Δ
/* whatever */
// just this line
/** Javadoc */
for interfaces classes & methods
package-info.java
{@link pkgname#class name}
Do for interfaces first
Can find bugs via good Javadocs
{@inheritDoc} to get interface javadoc

Collections

in java.util
Motley group
Set - all unique
HashSet
no need to ✓ for dupes
Iterator interface
.hasNext(), .next(), .remove(foo)
Parameterized Types - Safer
Set<Player> must all be players
NO casting in code ✓

Enum

EnumSet<class>
SortedSet, TreeSet
+ Comparitor
HashMap<String, Set<Player>>

Developer Ecology

IDE = editor/compiler/debugger
docs/✓ syntax
+ imports
refactoring
Helps navigate code
Continuous workflow

JUnit
Make code reliable
import org.junit. After
.Before
.Test
.Assert.*

Thorough code review Good
FindBugs.sf.net
checkstyle.sf.net