Software Development

Introducing OOP & Java

Koen Pelsmaekers

Unit Informatie (GT 03.14.05)

email: koen.pelsmaekers@kuleuven.be

Overview

- Software engineering: Object-oriented programming (90's)
  - level of abstraction
- Agile development: iterative & incremental
- Java: the language
  - the JVM
- Short history of Java
- Java 5: Tiger
- Some "special" language features
  - Documentation in Java
  - Exception handling
  - Implicit pointers
  - Class Object
    - Cloning objects, ...

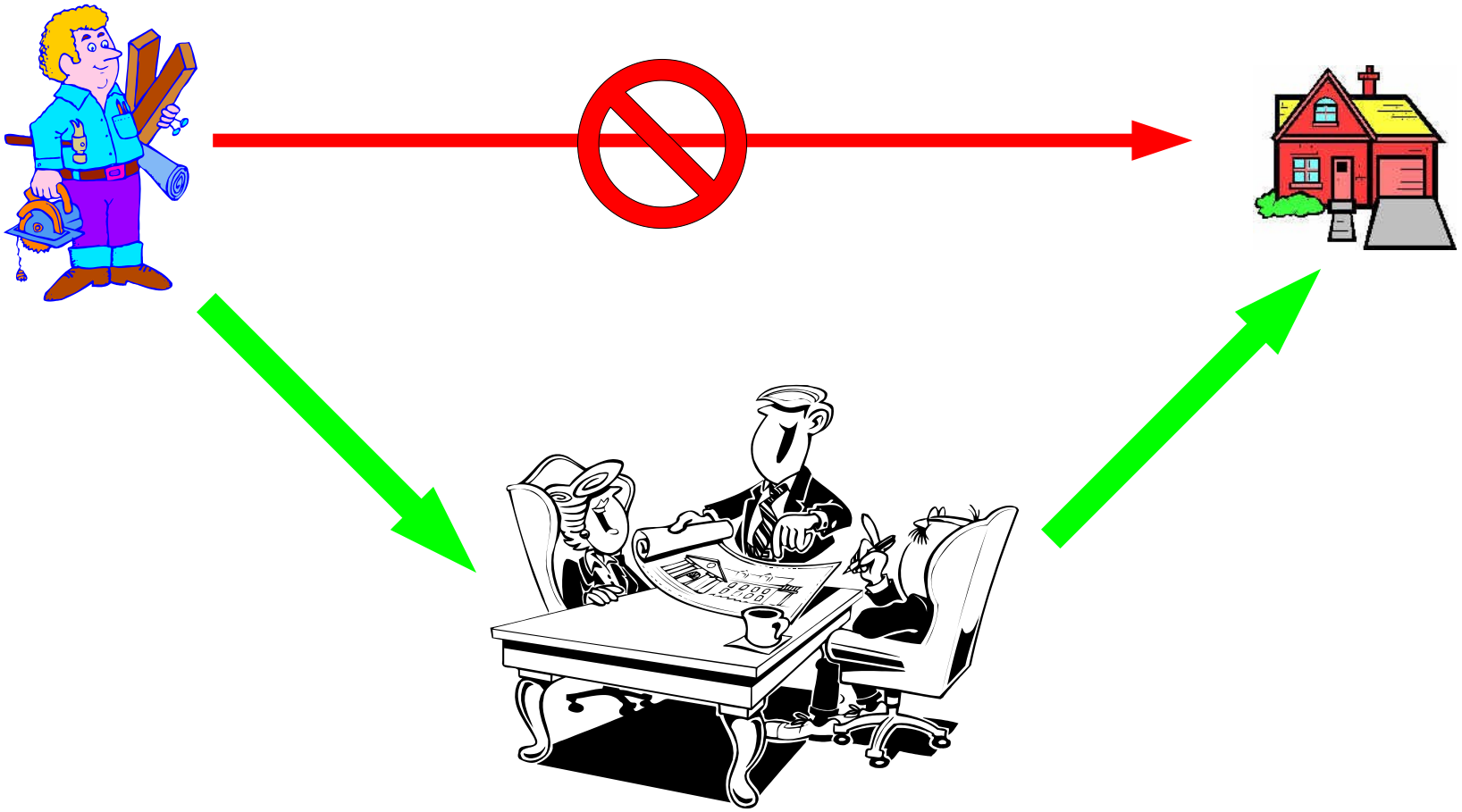
Object-oriented Software development

is

The key to modern Software engineering...

Software engineering

- Software engineering is a "natural" thing to do

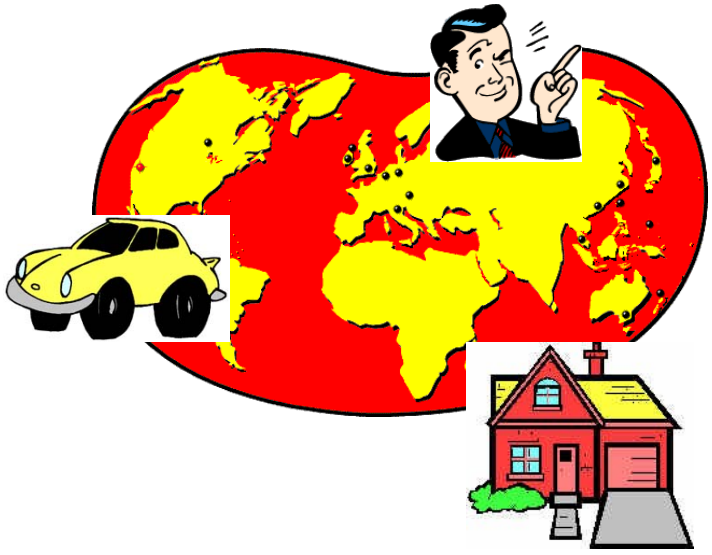


Structured programming

'70: “Good Programming”

== Structured Programming
+ language support (Pascal, C, ...)

Real world



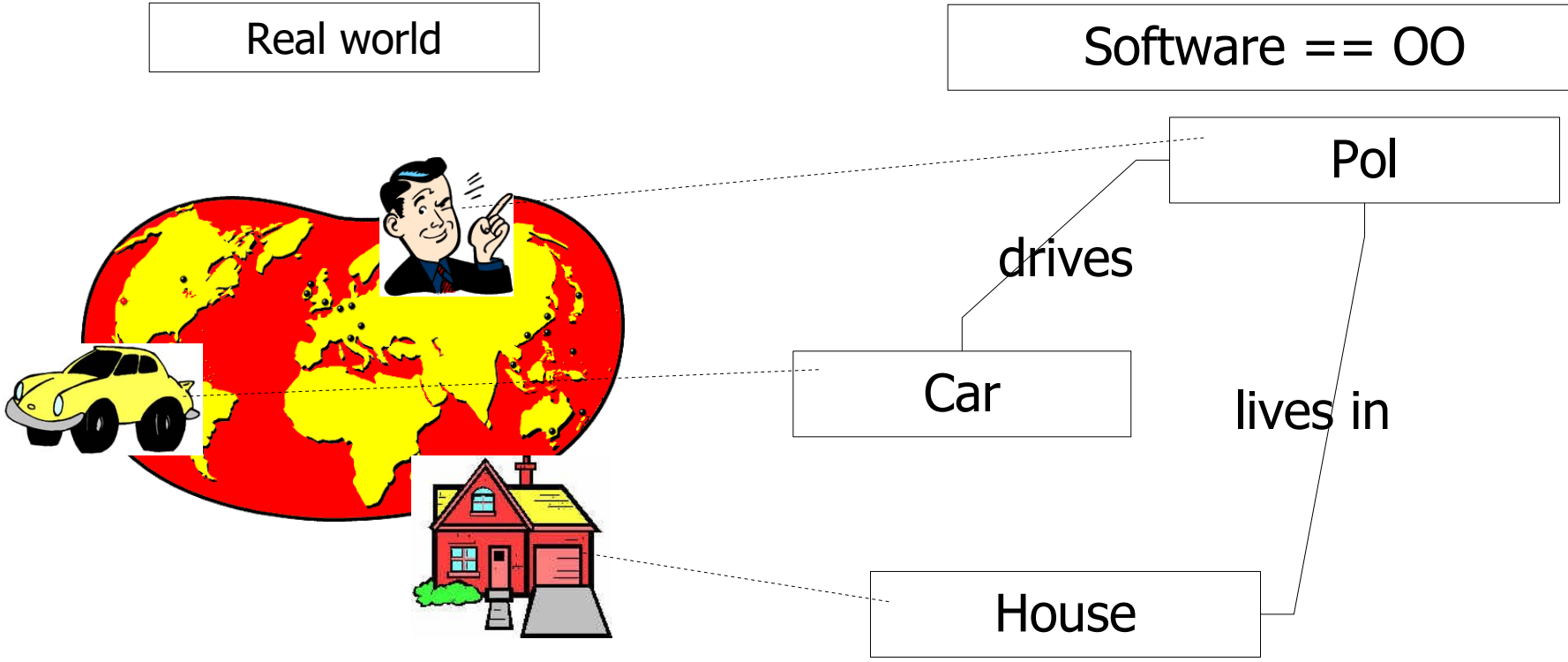
Semantic gap

Software == procedural

Object-oriented programming

'90: “Good Programming”

== Object-oriented Programming
+ language support (Smalltalk, C++, Java, ...)



Benefits of object-oriented programming

- Better modelling of the real (complex!) world

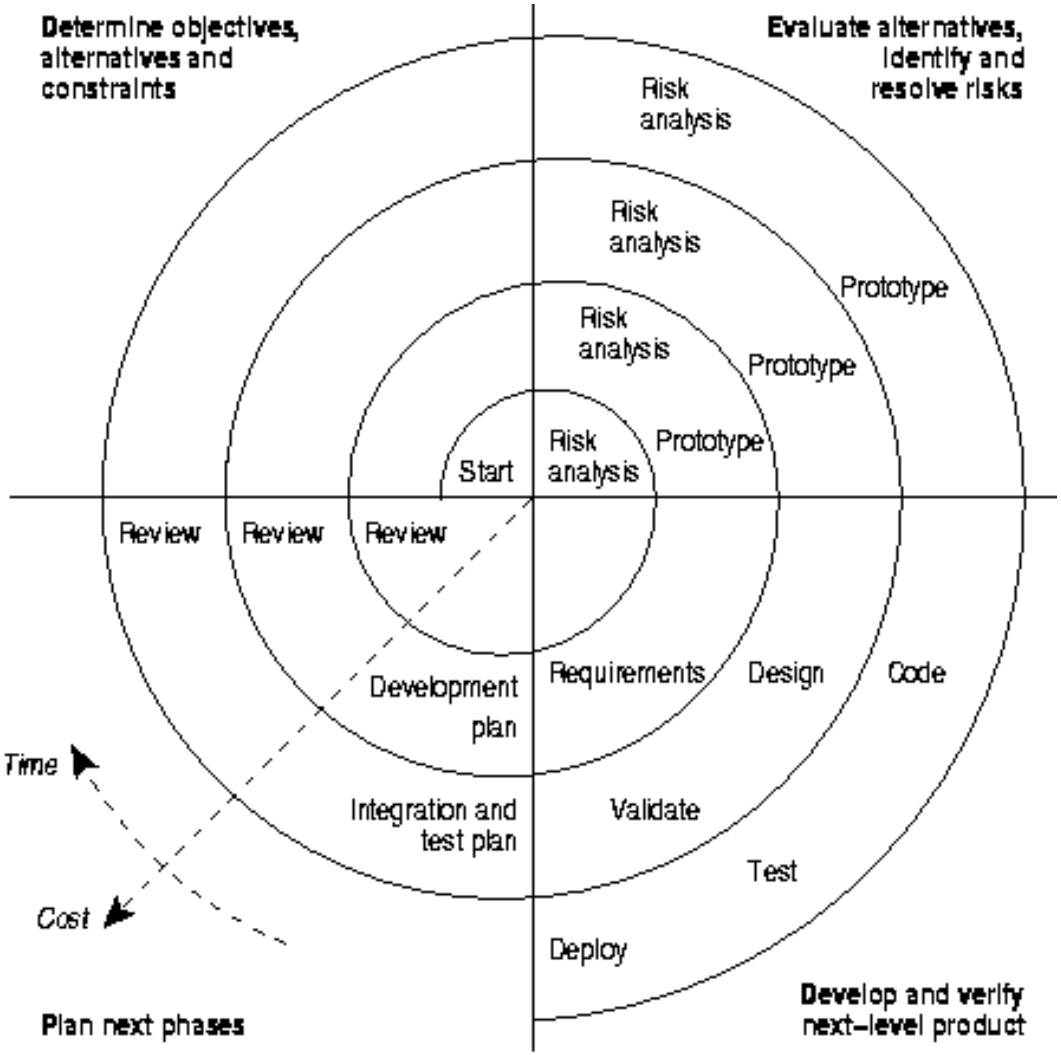
  Requirements == Analysis == Implementation
  (no semantic gap any more)
- Less and easier maintenance
- Higher level of abstraction
  - less sensitive to changes
    - Seperation of concerns, information hiding
  - “locality of change”
- Quality control for software
- Reusable code (long term)
  - Writing generic code is difficult
- Delay of implementation details
  - Objects can be changed easily
- Fast release cycle

Biggest challenge

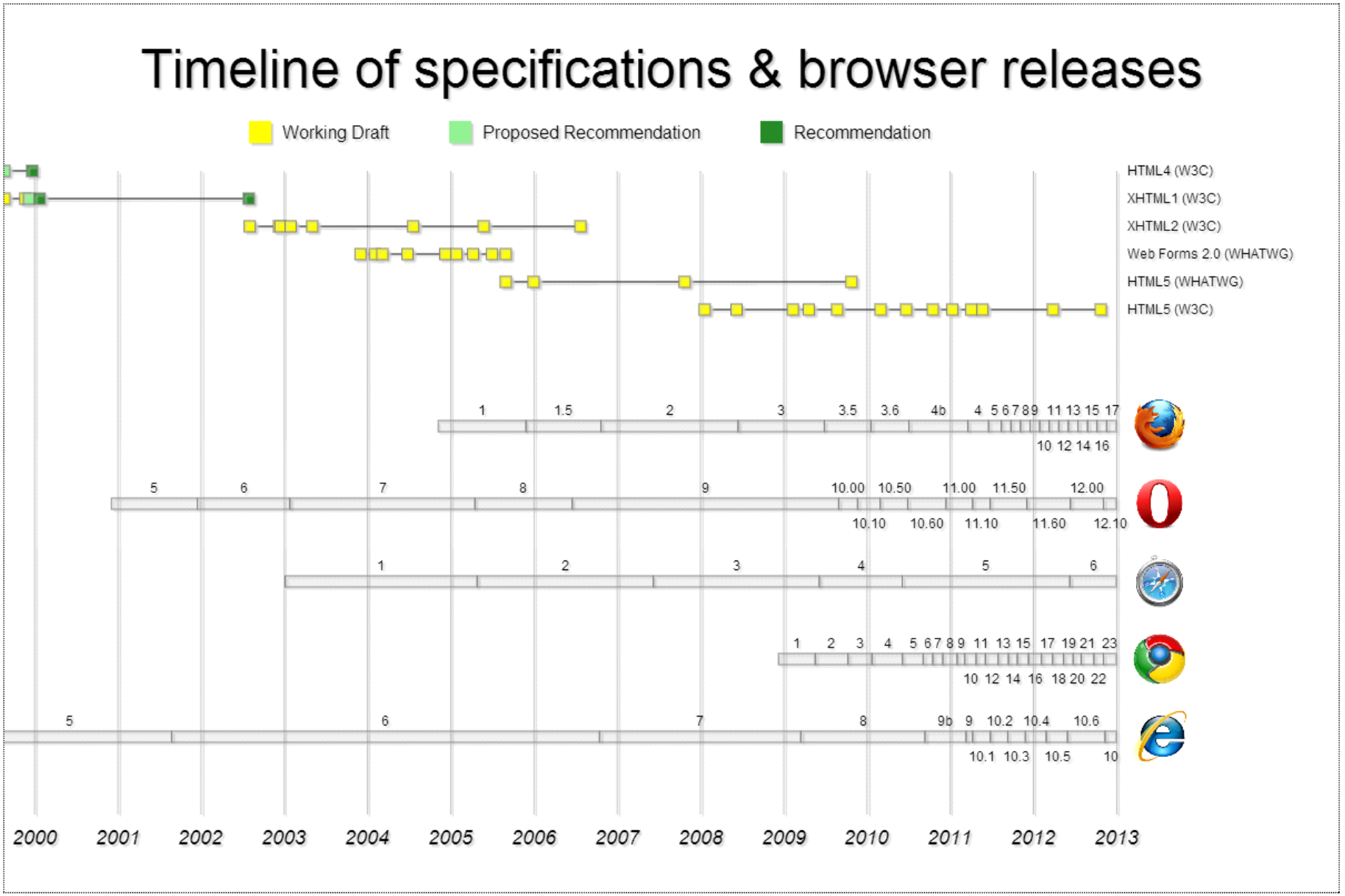
"Let's find classes"

Agile development process

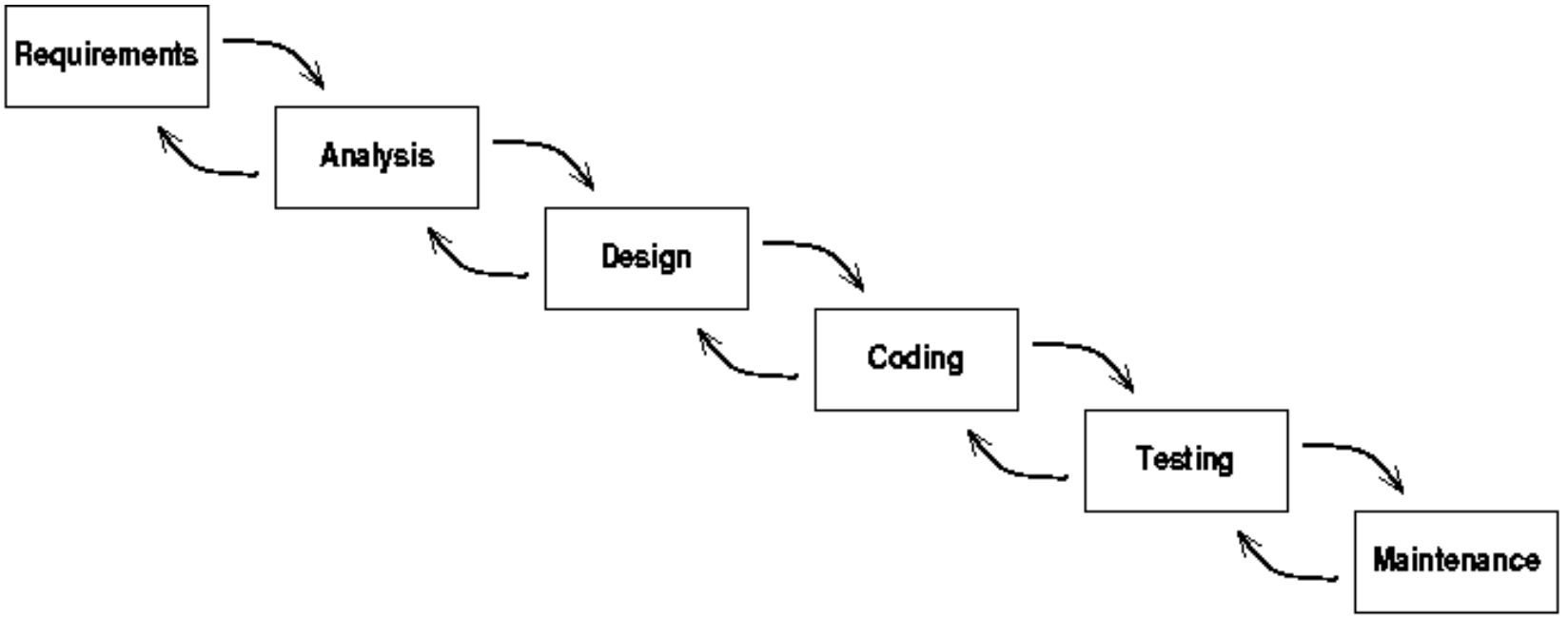
- Iterative
- Incremental
- Example:
  - Spiral Model
- System evolves toward final form
  - changes inspired by user feedback on prototypes



Shorter time to market...



Waterfall model

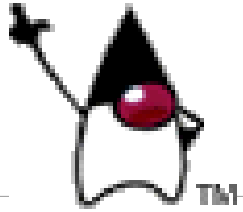


Java: the language

Bruce Eckel about Java 5

[...but J2SE5 (clearly driven by the features in C# 2.0) has leapt forward far enough that you suddenly wake up and say "hey, this isn't a 'simple' language anymore!" (Many will argue that it never was).]

What is Java?



20 years of Java!

programming language

Sun/ORACLE

JDK 1.0.2

J2SE Core Java Software

J2ME Java Mobility Software

oak

Write Once, Run Everywhere™

familiar (to C++)

Tiger (J2SE 5.0)

Train once, write anywhere!

Indonesia

classes & objects

packages

interpreted

virtual machine

byte code

interface

Duke

exception handling

applet

application

networks

rich built in class library

servlet

server programming

James Gosling
the “father” of Java

no operator overloading

distributed

dynamic binding

simple

distributed application

J2EE Enterprise Java Software

single inheritance

unicode

performance

JSP & tag libraries

multi threaded

javadoc

portable

New language features

Microsoft

JavaBeans

no pointers

secure

automatic garbage collector

implicit pointers to objects

robust

12 000 000 developers

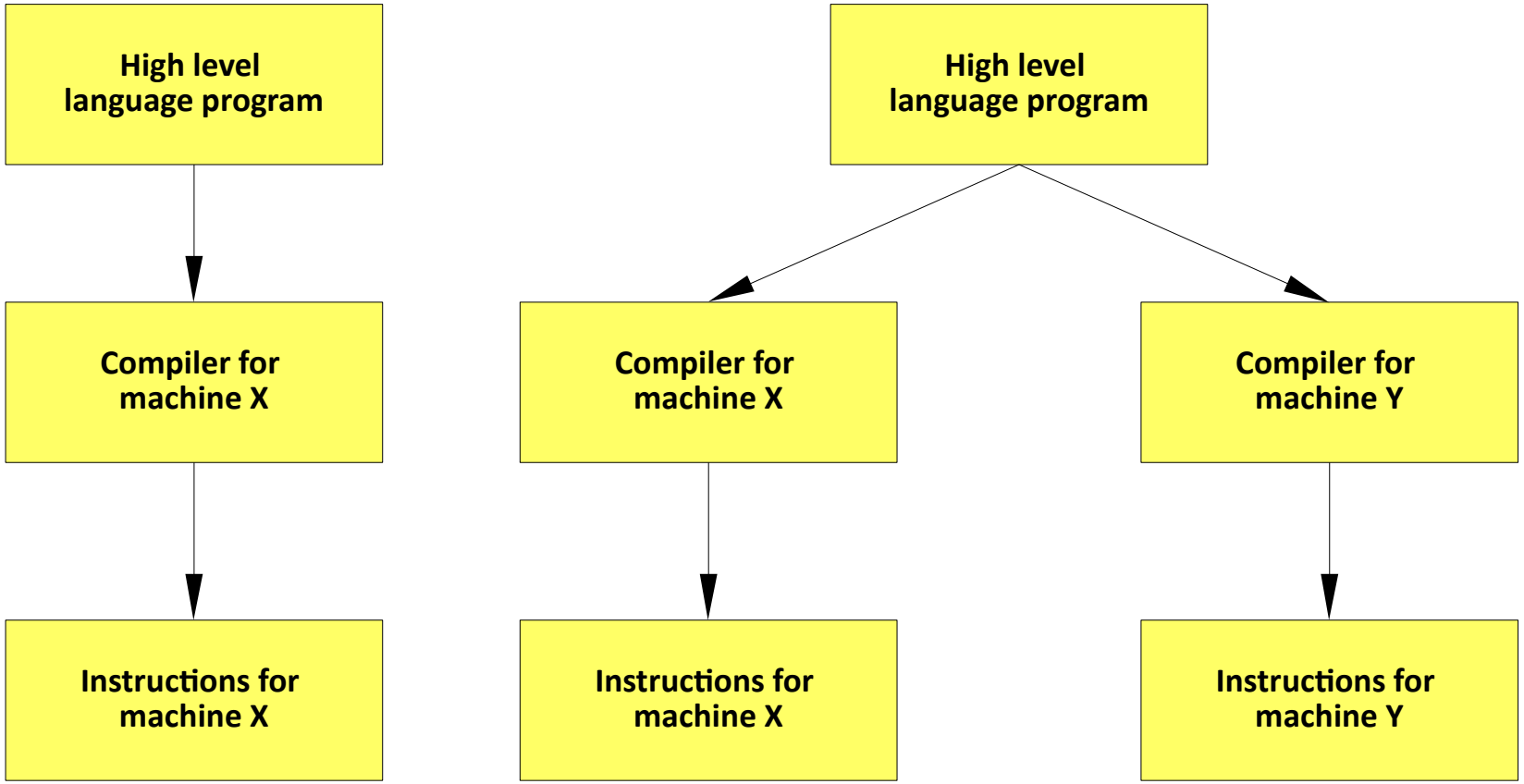
coffee

C++--++

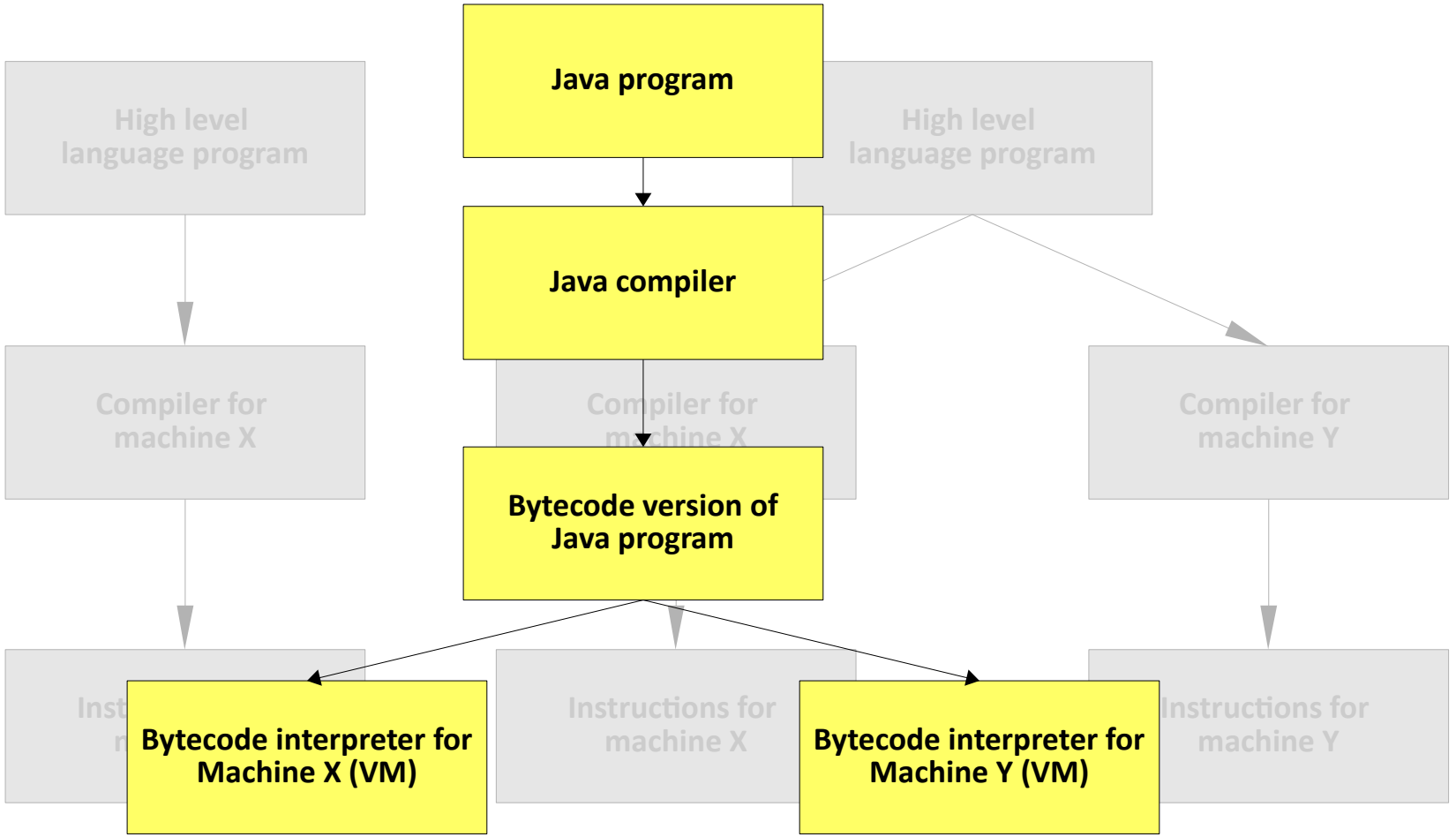
lambda expressions

Java SE 6 - Java SE 7 – Java 8 – Java 9 – Java 10 – Java 11 – 12 – 13 – 14 – ...

"classical" programming language

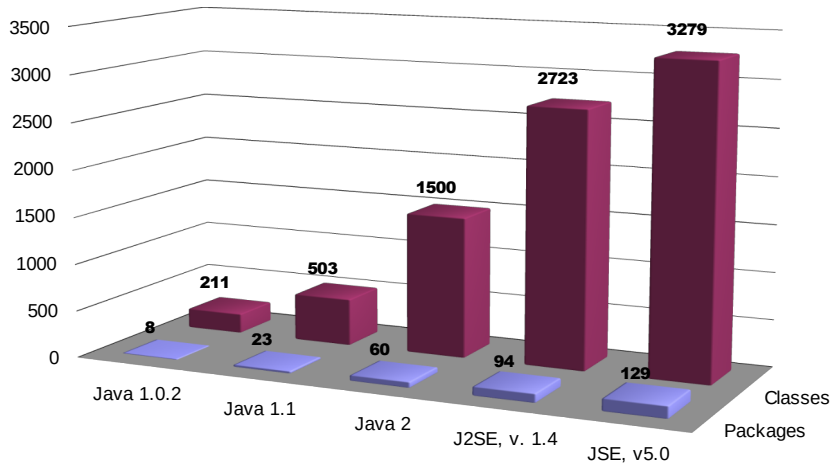


Java and the JVM



Short history (1)

- 23 May 1995: Java launched
- 9 December 1996: JDK 1.1
  - JavaSoft releases the Java Developer's Kit (JDK) 1.1, which includes new versions of the Java Vitual Machine, Java class libraries, and development tools. The JDK 1.1 also adds several other features in high demand among developers, including JDBC for database connectivity, RMI, ...
- December 1998: Java 2
  - Major release with a lot of enhancements: Swing, Java 2D, Drag & Drop, Collections, Audio Enhancements, JDBC Enhancements, Performance Enhancements, ...



Short history (2)

- 1 October 2004: Java 5, Tiger (next slide)
- 27 January 2010: ORACLE acquires SUN (+ Java)
- 28 July 2011: JDK 7
  - Dynamic languages support
  - Minor changes in try...catch, string in switch statement, generic type definition improvements (<>), ...
- 18 March 2014: JDK 8
  - Lambda expressions, new Date & Time API, …
- September 2017: JDK 9
  - Project Jigsaw (Java Platform Module System (jlink), JShell, new version-string: $MAJOR.$MINOR.$SECURITY.$PATCH
- March 2018: JDK 10 (20/03/2018)
  - var, ... (no "big" releases any more, only smaller changes every 6 months)

Short history (3)

- September 2018: JDK 11 (LTS) (25/09/2018)
  - Http-client, var in Lambda-expression, …
  - LTS: Long Term support version
- March 2019: JDK 12 (19/03/2019)
  - Internal improvements
  - Switch expression (preview)
- September 2019: JDK 13 (17/09/2019)
  - "hundreds of smaller enhancements and thousands of bug fixes"
  - Text Blocks (preview)

Short history (4)

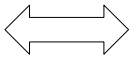
- March 2020: JDK 14 (17/03/2020?)
  - Switch expression (no preview any more)
  - Better NullPointerException messages (preview)
  - Enhanced instanceof (preview)
  - Records (preview) (example: see next slide)
- September 2020: JDK 15
  - Text Blocks (multi-line String literal)
  - Records (re-preview)(= "data-carrier class")
  - Sealed classes and interfaces (preview)
- March 2021: JDK 16
- September 2021: JDK 17 (LTS)

More information: http://jdk.java.net

Records (preview)

- Only keep data
- No custom behaviour, methods
- No "OOP"
  - No inheritance, dynamic binding, ...
  - Getters, setters, toString(), equals(), hashCode() for free

```
public record BankTransaction(LocalDate date,
        double amount,
        String description) {}
```



```
public class BankTransaction {
    private final LocalDate date;
    private final double amount;
    private final String description;

    public BankTransaction(final LocalDate date,
                           final double amount,
                           final String description) {
        this.date = date;
        this.amount = amount;
        this.description = description;
    }

    public LocalDate date() {
        return date;
    }

    public double amount() {
        return amount;
    }

    public String description() {
        return description;
    }

    @Override
    public String toString() {
        return "BankTransaction{" +
                "date=" + date +
                ", amount=" + amount +
                ", description='" + description + '\'' +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        BankTransaction that = (BankTransaction) o;
        return Double.compare(that.amount, amount) == 0 &&
                date.equals(that.date) &&
                description.equals(that.description);
    }

    @Override
    public int hashCode() {
        return Objects.hash(date, amount, description);
    }
}
```

Source: https://blogs.oracle.com/javamagazine

Java 5: Tiger

- Significant new language elements
  - Generics (typed collections)

    ```
    List<String>, Map<String,Person>, ...
    ```
  - Enhanced for loop (iteration over collections: foreach)

    ```
    for (Person p: persons) { ... }
    ```
  - Typesafe enums
    ("enum" keyword introduced)
  - Autoboxing/unboxing (primitives vs. reference types)

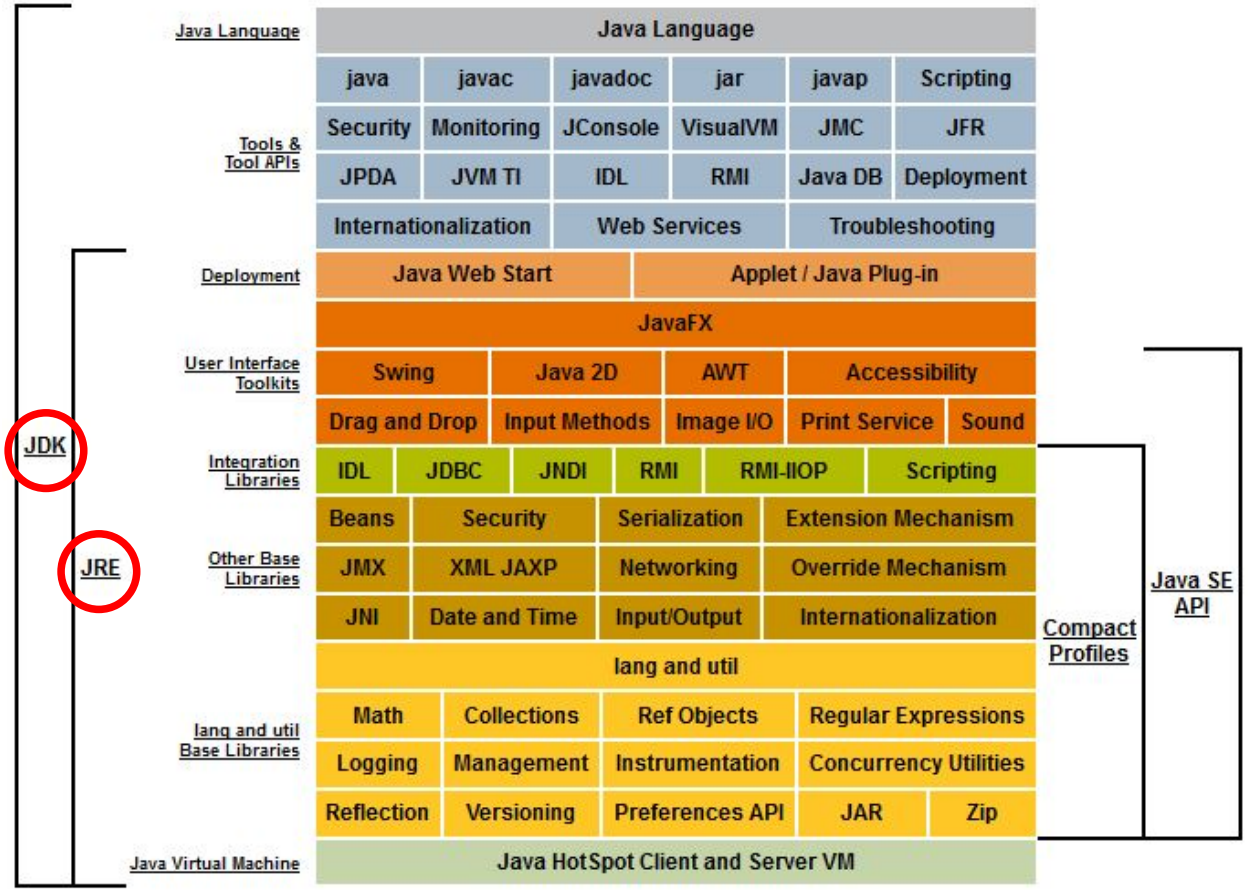
    ```
    new Integer(15) + 5,  List<Integer> list; list.add(17);
    // list.add(new Integer(17));
    ```
  - Varargs (methods with a variable number of arguments)
  - …
- Many enhancements

Java 5 packages

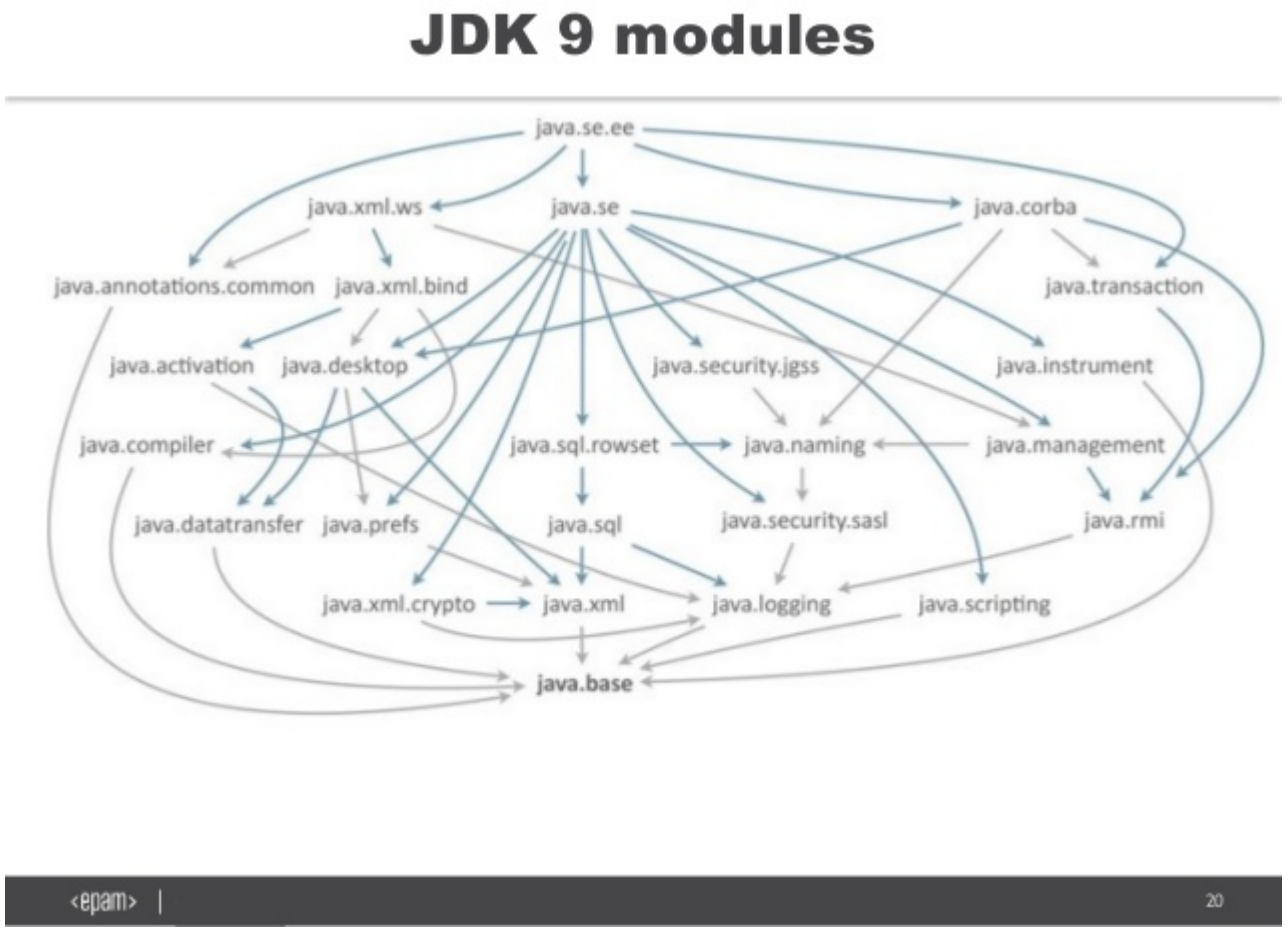
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
java.nio.channels
java.nio.channels.spi
java.nio.charset
java.nio.charset.spi
java.rmi
java.rmi.activation
java.rmi.dgc
java.rmi.registry
java.security.spec
java.sql
java.text
java.util
java.util.concurrent
java.util.concurrent.atomic
java.util.concurrent.locks
java.util.jar
java.util.logging
java.util.prefs
java.util.regex
java.util.zip
javax.accessibility
javax.crypto
javax.crypto.interfaces
javax.crypto.spec
javax.imageio
javax.imageio.event
javax.imageio.metadata
javax.imageio.plugins.bmp
javax.imageio.plugins.jpeg
javax.imageio.spi
javax.imageio.stream
javax.management
javax.management.loading
javax.management.modelmbea
javax.management.monitor
javax.management.openmbea
javax.naming.directory
javax.naming.event
javax.naming.ldap
javax.naming.spi
javax.net
javax.net.ssl
javax.print
javax.print.attribute
javax.print.attribute.standar
javax.print.event
javax.rmi
javax.rmi.CORBA
javax.rmi.ssl
javax.security.auth
javax.security.auth.callback
javax.security.auth.kerberos
javax.security.auth.login
javax.security.auth.spi
javax.security.auth.x500
javax.security.cert
javax.security.sasl
javax.sound.midi
javax.sound.midi.spi
javax.sound.sampled
javax.sound.sampled.spi
javax.sql
javax.sql.rowset
javax.sql.rowset.serial
javax.swing.filechooser
javax.swing.plaf
javax.swing.plaf.basic
javax.swing.plaf.metal
javax.swing.plaf.multi
javax.swing.plaf.synth
javax.swing.table
javax.swing.text
javax.swing.text.html
javax.swing.text.html.parse
javax.swing.text.rtf
javax.swing.tree
javax.swing.undo
javax.transaction
javax.transaction.xa
javax.xml
javax.xml.datatype
javax.xml.namespace
javax.xml.parsers
javax.xml.transform
javax.xml.transform.dom
javax.xml.transform.sax
javax.xml.transform.stream
javax.xml.validation
javax.xml.xpath

129 packages

Java 8 JDK/JRE



Java 9: modules



Some other new language features

- Java 8: streams & lambda's (see next part)
- Java 10: type inference for local variables with var (but Java stays a strongly typed language)

```
Map<String, List<City>> countryToCity = new HashMap<>();
// ...
for (Map.Entry<String, List<City>> citiesInCountry :
                                          countryToCity.entrySet()) {
    List<City> cities = citiesInCountry.getValue();
    // ...
}

var countryToCity = new HashMap<String, List<City>>();
// ...
for (var citiesInCountry : countryToCity.entrySet()) {
    var cities = citiesInCountry.getValue();
    // ...
}
```

Java vs. other programming languages

- Some links to interesting web-pages (and nice posters) about the history of programming languages

  http://www.levenez.com/lang/

- And the history of Java

  http://ei.cs.vt.edu/book/chap1/java_hist.html

  http://www.java.com/en/javahistory/

Other JVM languages

- Many languages can compile to Java classes
  - JVM is very performant
- Most important languages
  - Kotlin (JetBrains)
    - type inference
    - can cooperate with Java
    - adopted by Google (Android) and Pivotal (Spring 5 framework)
  - Scala
    - statically typed and functional
    - scalable and language
  - Groovy
    - scripting, dynamic types
  - JRuby
    - Java version of Ruby
    - Ruby on Rails (web framework)

JVM: some opcodes (= byte code)

Stack			Math			Conversions		
87	(0x57)	*pop*	96	(0x60)	*iadd*	133	(0x85)	*i2l*
88	(0x58)	*pop2*	97	(0x61)	*ladd*	134	(0x86)	*i2f*
89	(0x59)	*dup*	98	(0x62)	*fadd*	135	(0x87)	*i2d*
90	(0x5a)	*dup_x1*	99	(0x63)	*dadd*	136	(0x88)	*l2i*
91	(0x5b)	*dup_x2*	100	(0x64)	*isub*	137	(0x89)	*l2f*
92	(0x5c)	*dup2*	101	(0x65)	*lsub*	138	(0x8a)	*l2d*
93	(0x5d)	*dup2_x1*	102	(0x66)	*fsub*	139	(0x8b)	*f2i*
94	(0x5e)	*dup2_x2*	103	(0x67)	*dsub*	140	(0x8c)	*f2l*
95	(0x5f)	*swap*	104	(0x68)	*imul*	141	(0x8d)	*f2d*
			105	(0x69)	*lmul*	142	(0x8e)	*d2i*

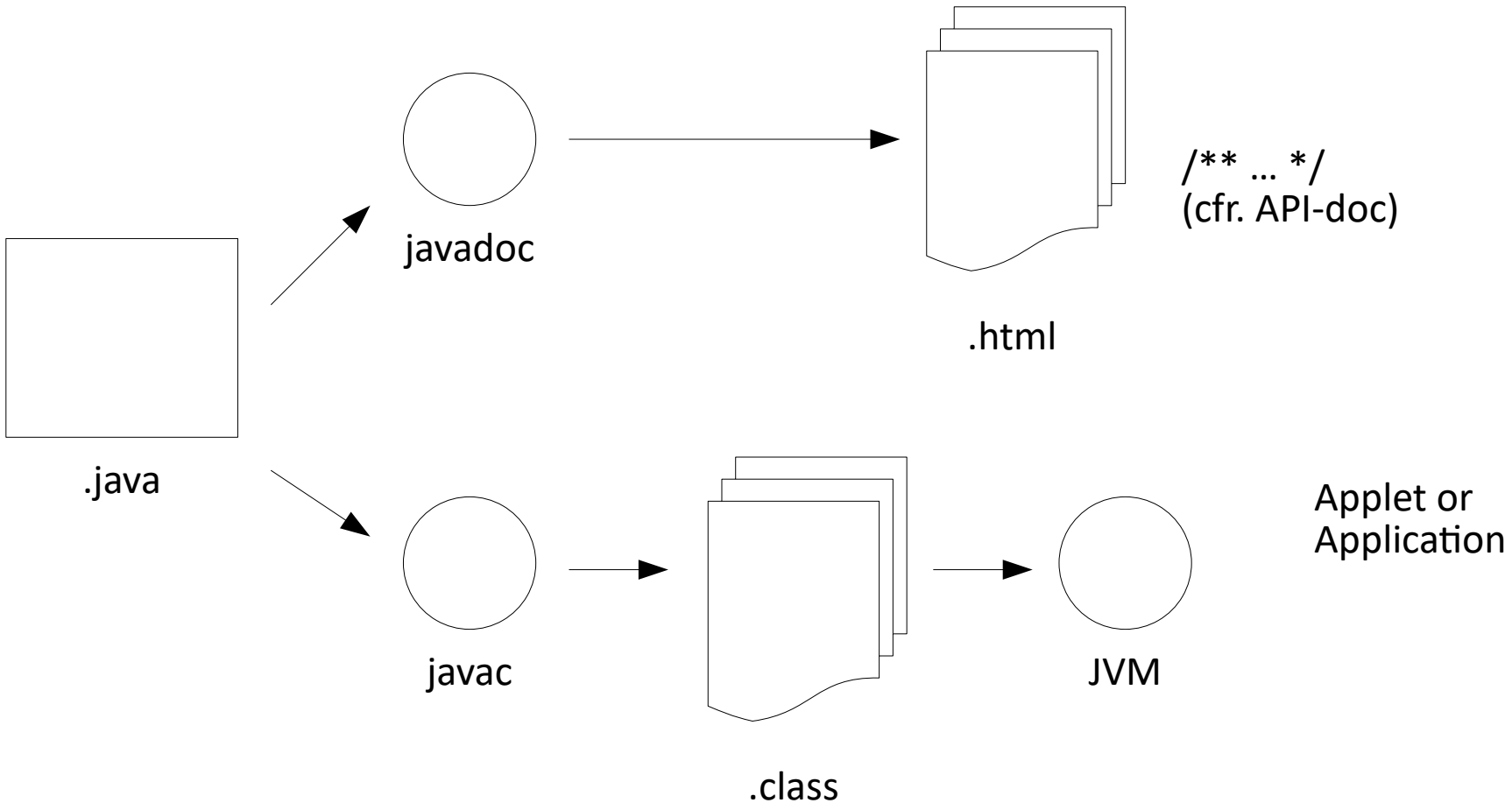
Java: some typical language features

Javadoc, Exceptions, Run-time stack vs. Garbage collected heap, implicit pointers to objects, class Object, cloning objects: deep and shallow copy

Some language features

- “C”-syntax?!
- Conditional and/or: && and ||
- Ternary operator
  - `int absValue = (a < 0) ? -a : a;`
- Widening vs. Narrowing (explicit cast needed!)
  - `int i = 13;`
  - `byte b = (byte) i; // byte b = i gives error`
- Javadoc (next slide)
- Exception handling: throw/ try … catch (next slide)
- Primitive type vs. Reference type (next slide)
  - Run-time stack vs. Garbage-collected heap

Built-in documentation with javadoc



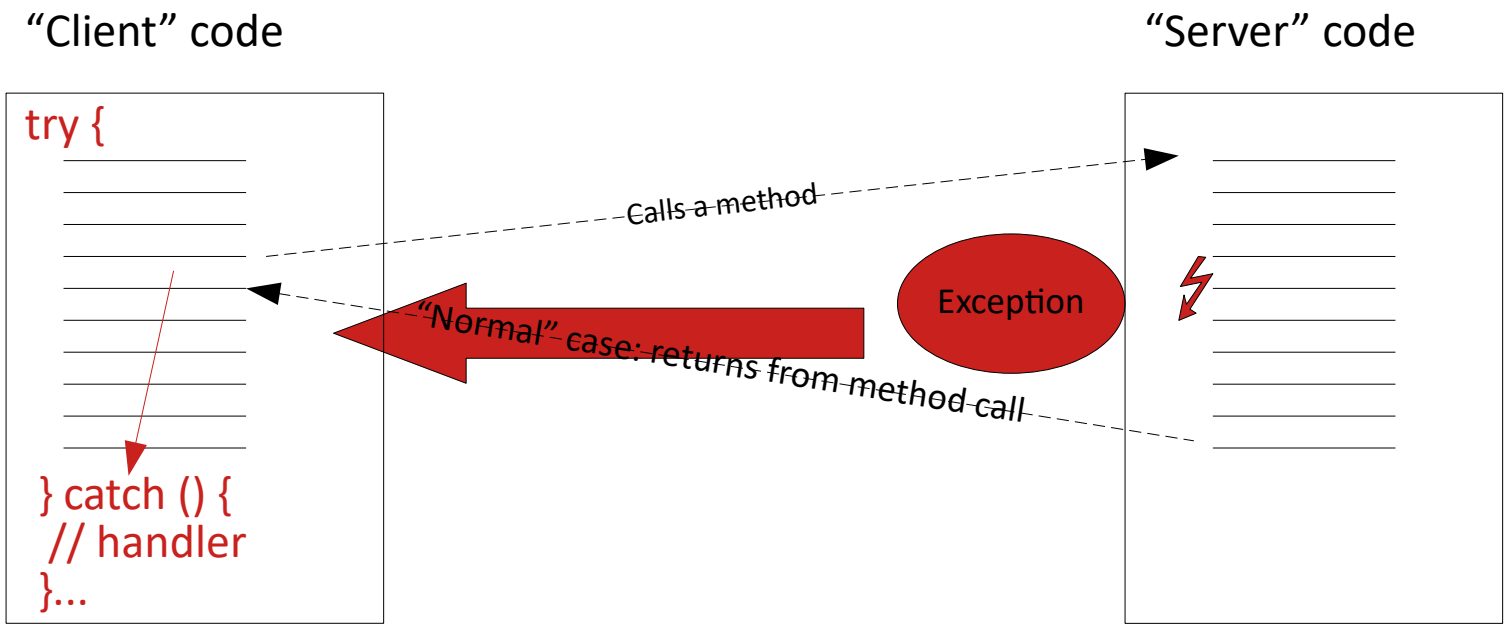
Some javadocs tags

- @author
- @param
- @return
- @throws
- @exception
- @since (library)
- @deprecated (library)
- …

Three levels of documentation

- End-user documentation
  - How can I use the program?
  - On-line help, f.i. html-based
- Integrator documentation
  - How can I use this class, this package?
  - Documentation of the *public* interface of a class
  - Javadoc comments & html-pages: `/** ..... */`
- Implementator documentation
  - How does this method do the job?
  - Documentation of the internals, *private* part, of a class
  - Well known comments: `//` and `/* .... */`

Exception handling

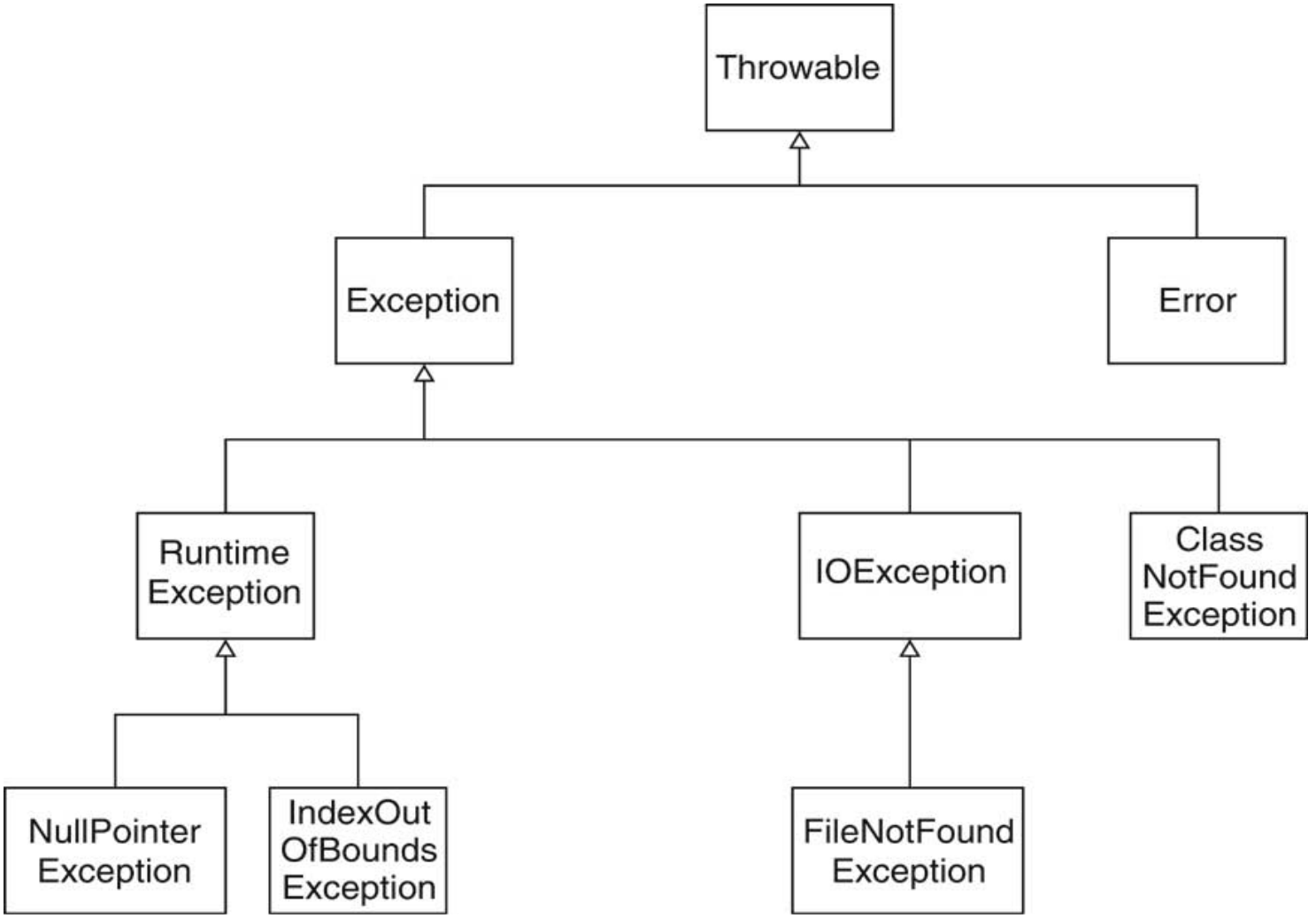


Exception handling

- Checked vs. Unchecked Exceptions
  - Unchecked: programming error
    - Inherits from RuntimeException
    - ArrayIndexOutOfBoundsException, ArithmeticException (f.i. “divide by zero”), ...
  - Checked exceptions: external problem
    - Does not inherit from RuntimeException
    - Should be declared in method throws declaration or handled in a try-catch
    - ClassNotFoundException, IOException, ConnectException, SQLException, …

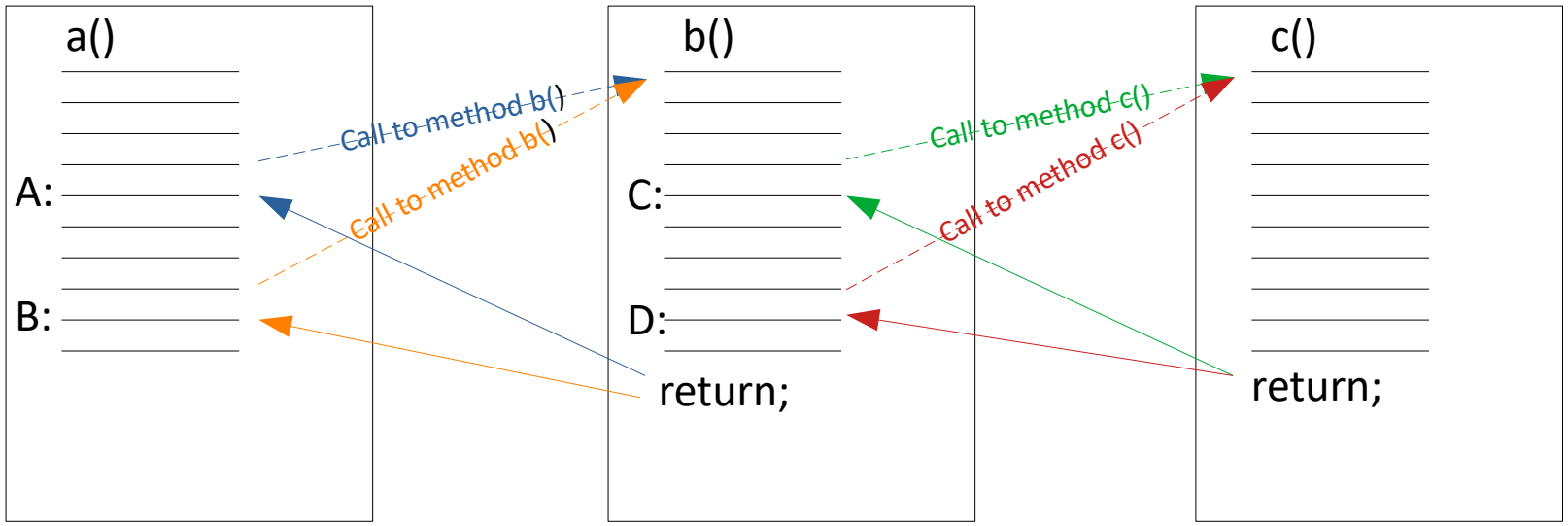
try….catch….finally statement

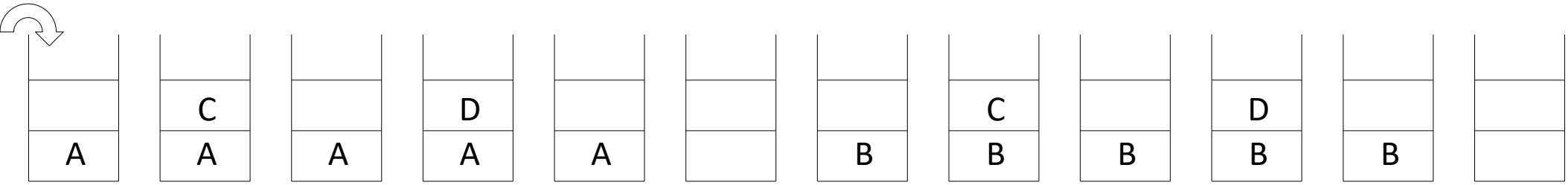
Exception hierarchy



(Nested) method calls

- What happens during (nested) method calls?

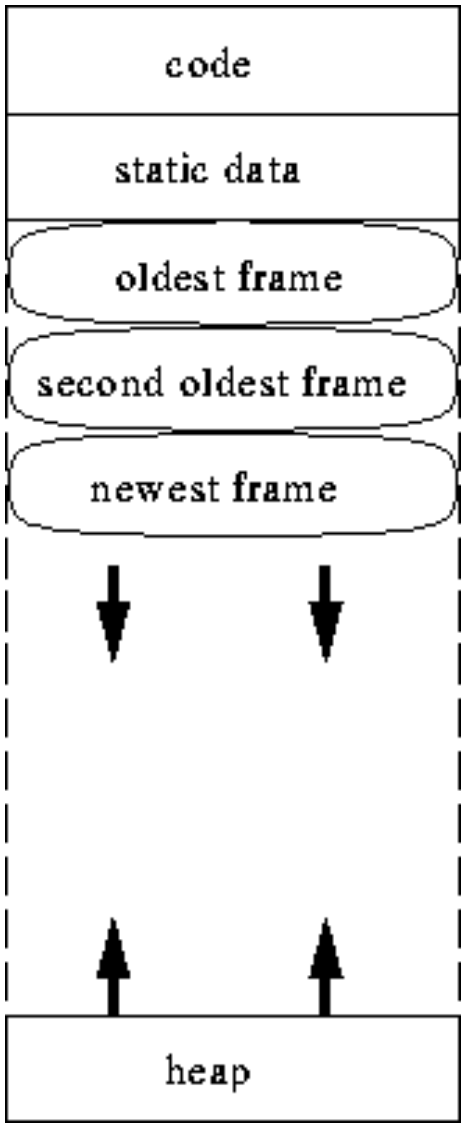




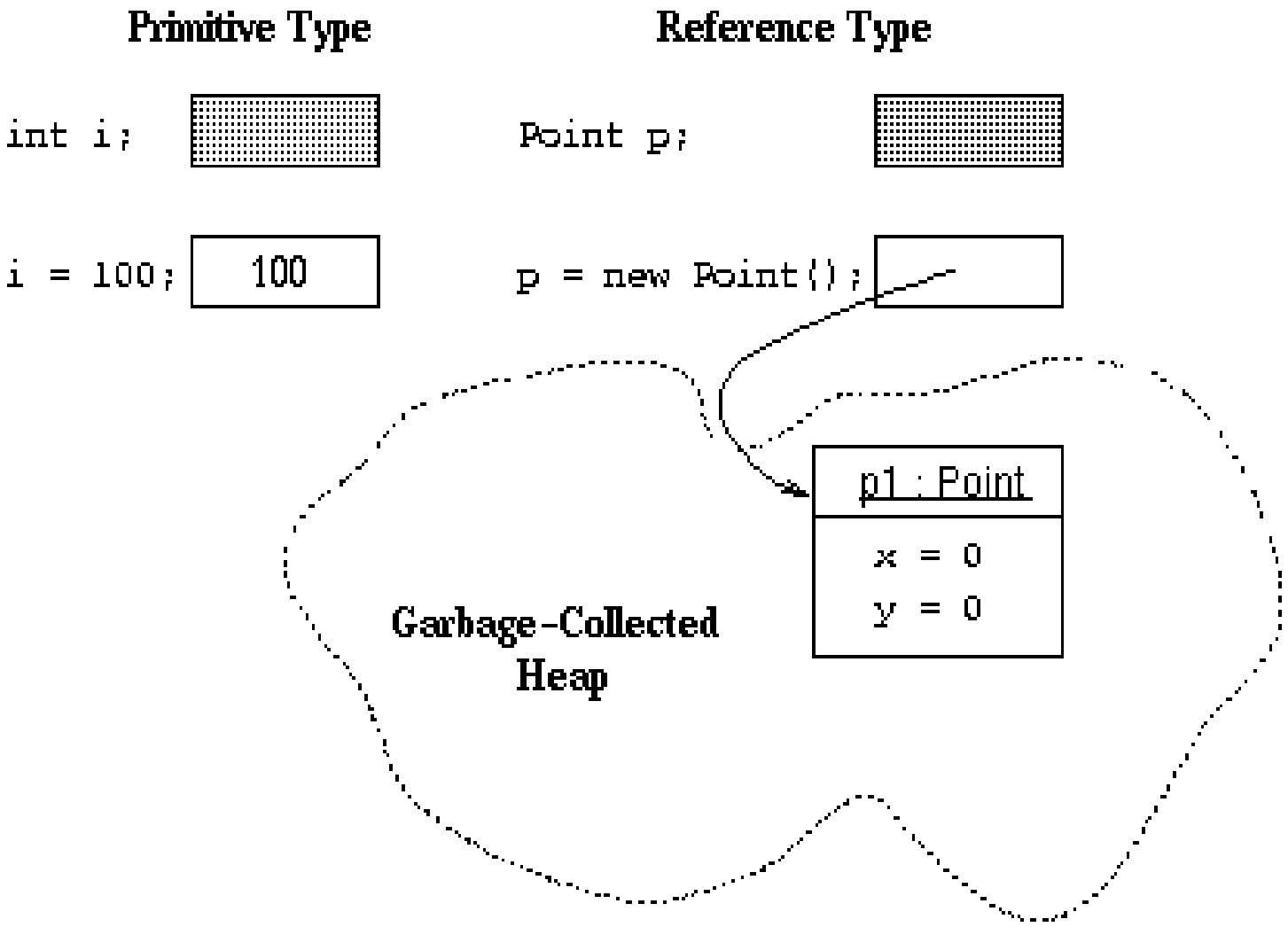
(Stack = LIFO)

run-time stack vs. heap

- call stack, control stack, execution stack, run-time stack
- keeps information for a method call (activation record)
  - “current” frame in a thread
  - local variables, operand stack, reference to run-time constant pool (dynamic linking)
- stack trace in Java
- “Stack overflow” or “Stack overruns heap”



Reference type and garbage collection



Be aware of the implicit pointers!

- Garbage collection: automatic
- Identity copy (next slide)
  - ➔ "dangerous" side effects?
- Cloning
  - ➔ shallow vs. deep copy (next slide)
- identity/equality check: == vs. equals()
  - ➔ String objects?
- Objects as parameter
  - ➔ call-by-value becomes call-by-reference
- Substitutability and dynamic binding
  - ➔ Downcast possible (no information is lost)

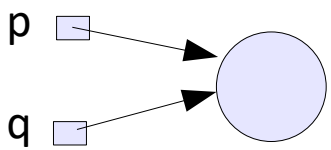
class Object

- Class Object is the root of the class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.
- Most important methods
  - `clone()`, `equals(Object obj)`, `getClass()`, `hashCode()`, `toString()`

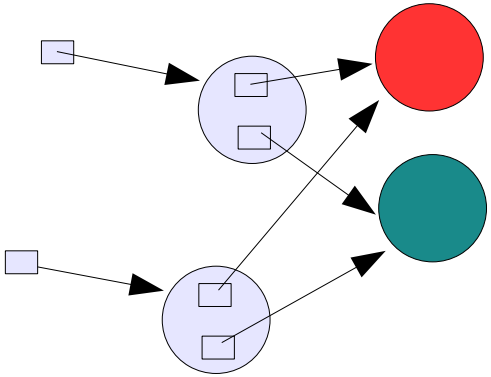
Cloning objects

Marker interface

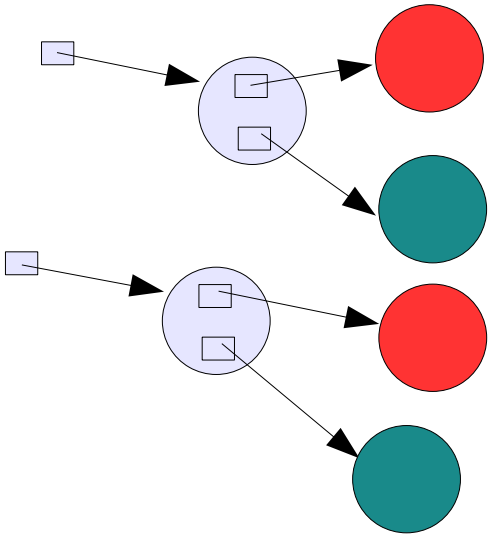
- Assignment is an **identity** copy
- Default: if Cloneable interface : **shallow** copy
  else CloneNotSupportedException
- Override clone() for a **deep** copy



q = p; (identity copy)



shallow copy
default clone()



deep copy
overridden clone()

Edit-Compile-Run-Debug Cycle