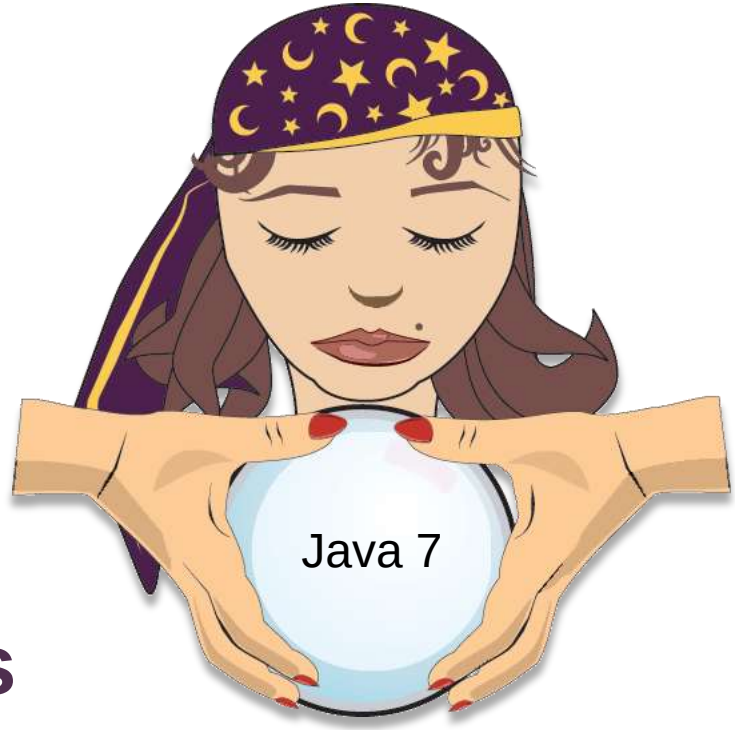
Xebia

Business Integration Architects



Java SE 7.0 Proposals

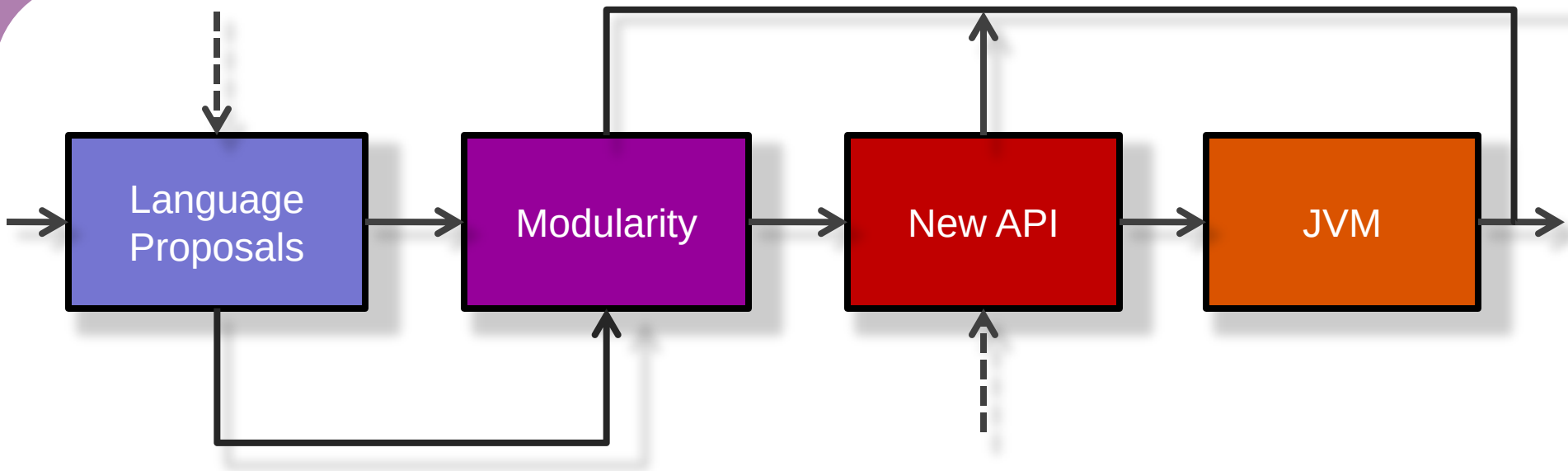
Dolphin's Overview (02 / 07 / 2008)

Erwan Alliaume

ealliaume(*AT*)xebia(*DOT*)fr

http://www.xebia.fr | http://blog.xebia.fr

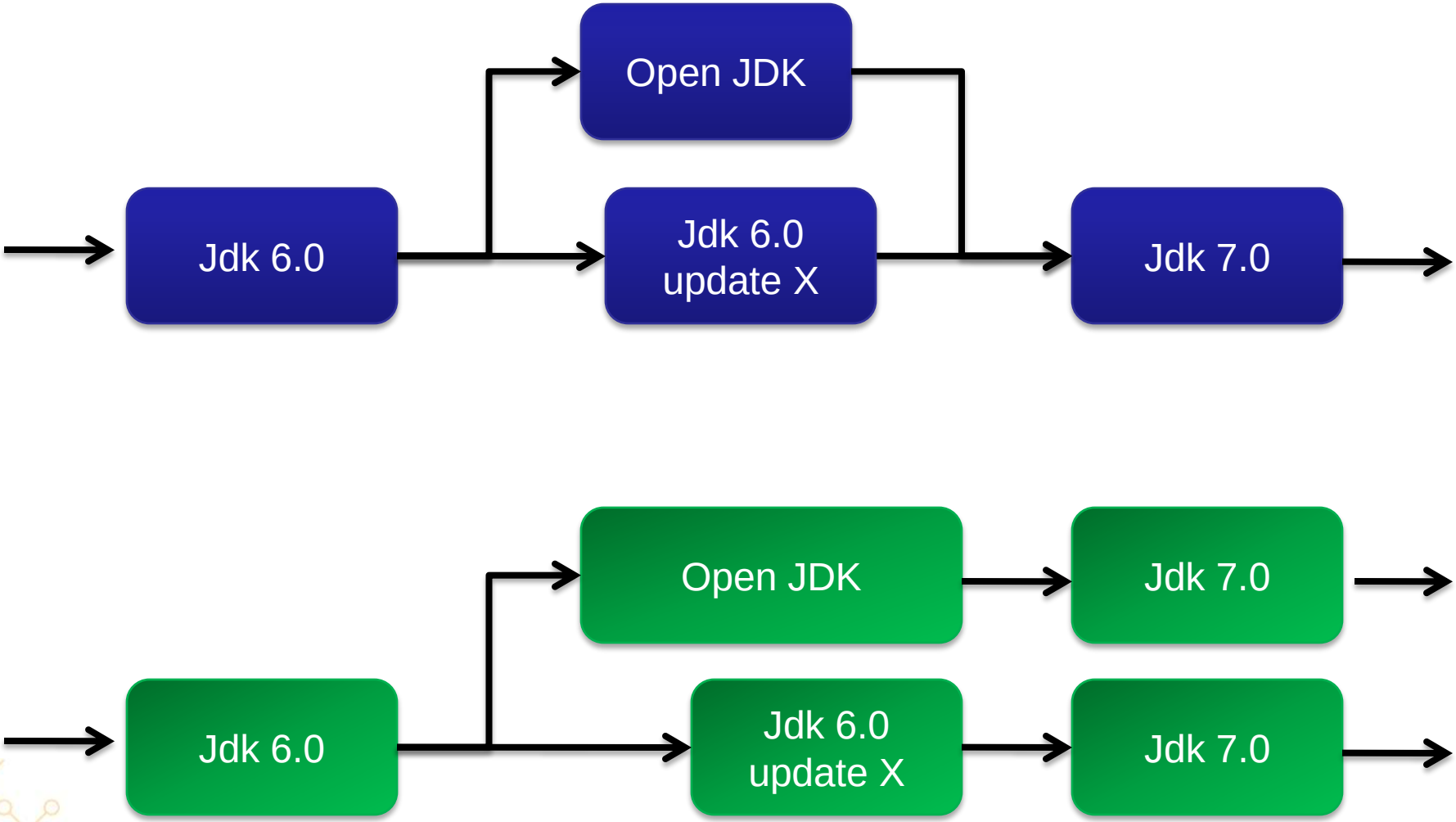
Please follow the guide … wind's direction =>



1. Langage proposals
2. Modularity
3. New API
4. JVM

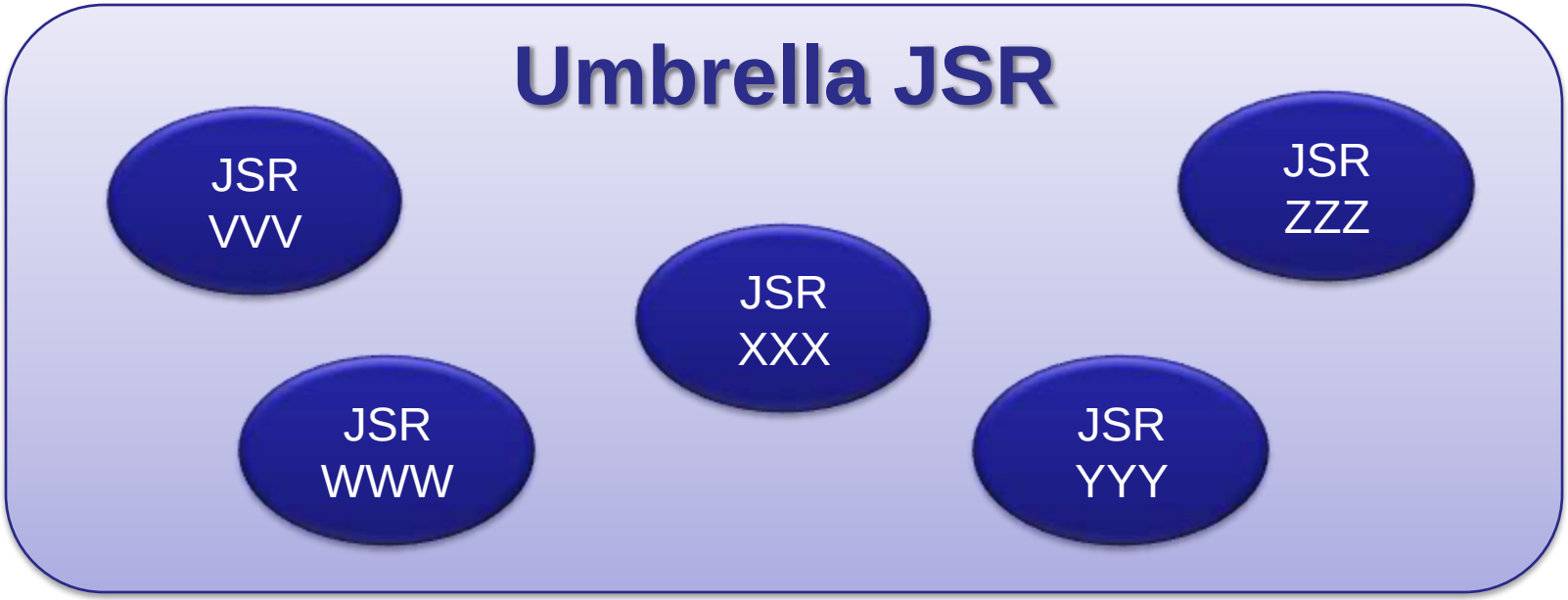
Xebia
Business Integration Architects

Openjdk Spaghettis Plate



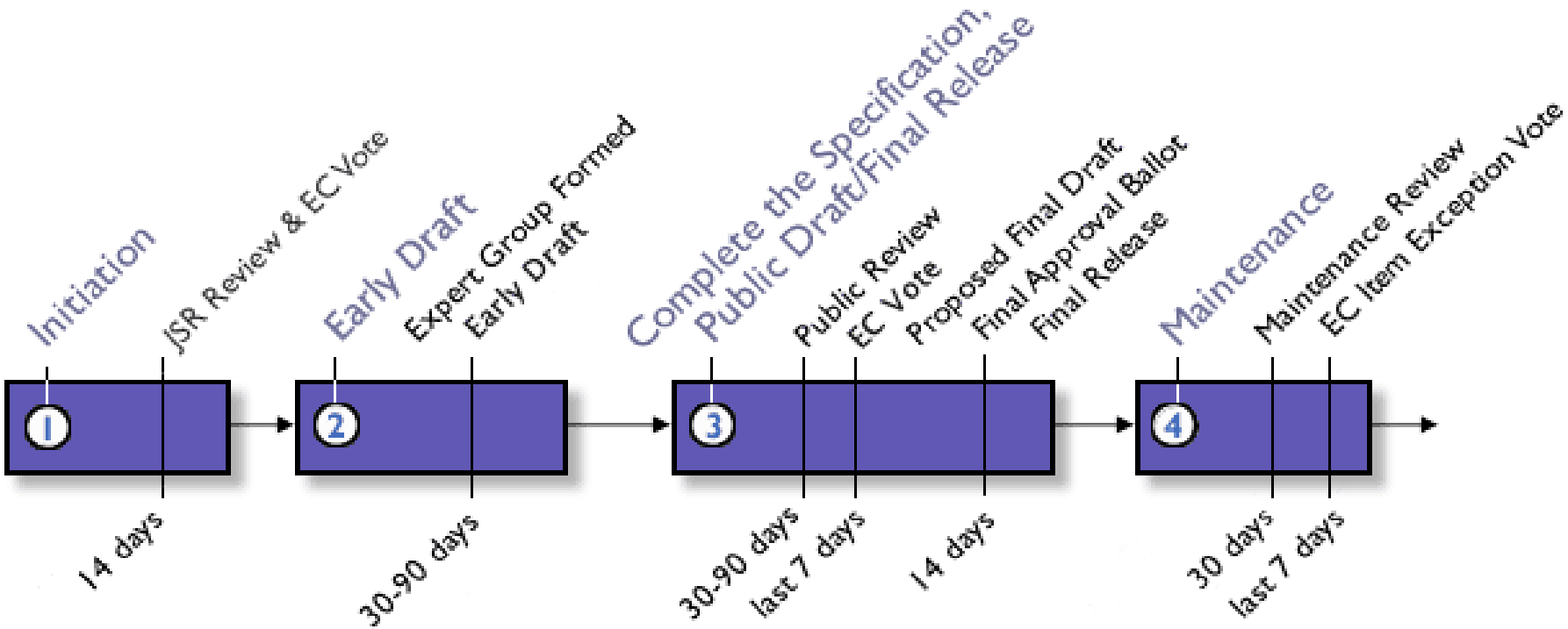
Xebia
Business Integration Architects

Looking for an Umbrella JSR



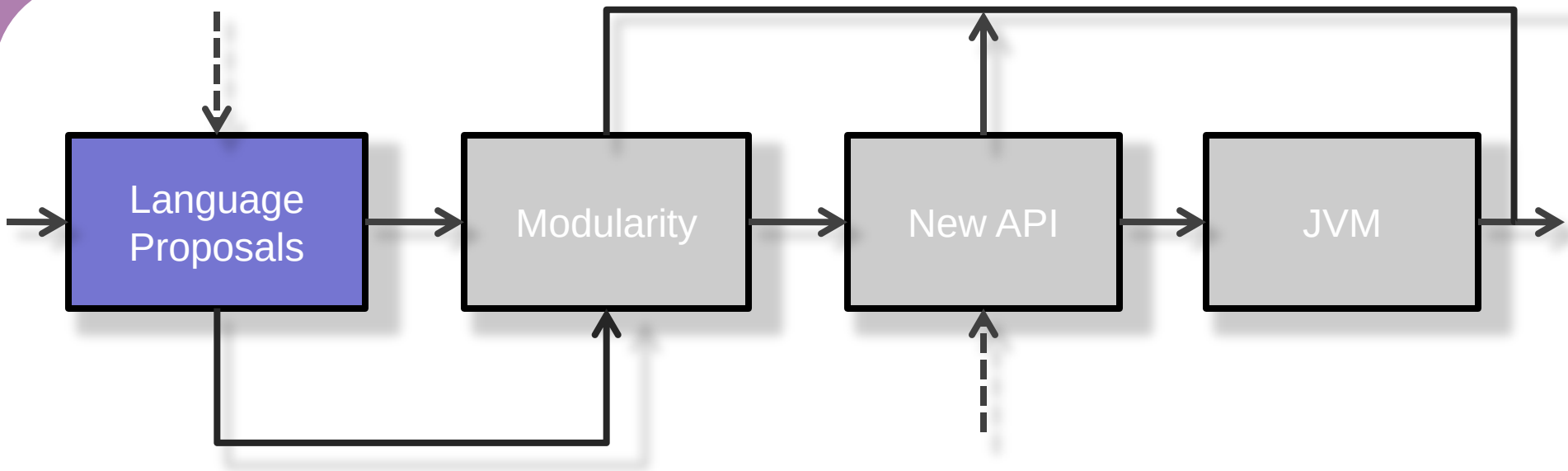
- **Sample with Java SE 6.0**
  - **Umbrella JSR 270 : Java SE 6.0 release content**
    - **JSR 199 : Java Compiler API**
    - **JSR 221 : Jdbc 4.0**
    - **JSR 223 : Scripting for the Java platform**
    - **JSR 269 : Pluggable Annotation Processing API**

JCP Timeline



Xebia
Business Integration Architects

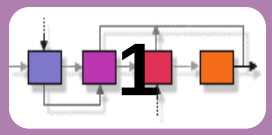
Please follow the guide …



1. **Langage proposals**
2. Modularity
3. New API
4. JVM

Xebia
Business Integration Architects

Language Level XML support



- Compiler understand XML
- No String / StringBuilder / StringBuffer is needed
- Language level include multi line support

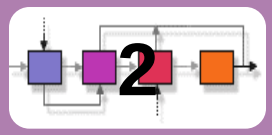
Use case sample

```
1. org.w3c.dom.Document d =
   <this><that><other>2</other></that></this>;
```

```
1. element.appendChild(
2.     <user>
3.         <name>Xebia</name>
4.     </user>);
```

Xebia
Business Integration Architects

Javabean properties declaration



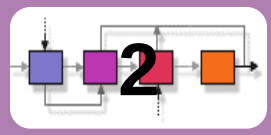
- IOC friendly
- No unchanged generated code
- Already exists in C# (syntax is different)
- How to make protected the setter ?

Use case sample

```
1. public class MyObject {
2.     private property String name;

3.     public String getName() {
4.         return name == null ? "" : name;
5.     }
6. }
```

Xebia
Business Integration Architects

Javabean properties access



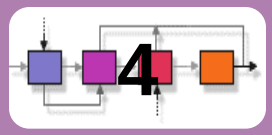
- Automatically call getters and setters
- What is the less confusing syntax ?

Use case sample

```
1.    p.name = "Xebia";              // calls setter
2.    String str = p=>name;          // calls getter
3.    P->name = "Xebia";             // calls setter
4.    String str = p#name;           // calls getter
```

Xebia
Business Integration Architects

Simplify BigDecimal usage



- Redefine Operator with annotation (C++ functionality)

NOW

```
1. BigDecimal dec = new BigDecimal("12.34");
2. dec = dec.add(new BigDecimal("34.45"))
3.           .multiply(new BigDecimal("1.12"))
4.           .subtract(new BigDecimal("3.21"));
```

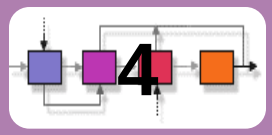
Could be with operator overloading

```
1. BigDecimal dec = new BigDecimal("12.34");
2. dec = dec + new BigDecimal("34.45")
3.           * new BigDecimal("1.12")
4.           - new BigDecimal("3.21");
```

Could be with literals

```
1. BigDecimal dec = 12.34n;
2. dec = dec + 34.45n * 1.12n - 3.21n;
```

String in switch statement

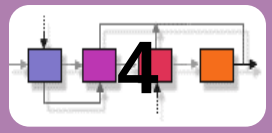


Use case sample

```
1.      final String str = “”;

2.      switch(str) {
3.            case "true":
4.                  return true;
5.            case "false":
6.                  return false;
7.            default:
8.                  throw new IllegalArgumentException(str);
9.      }
```

Xebia
Business Integration Architects

Enum comparison

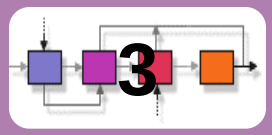


Use case sample

```
1. enum MyColor {
2.       EMPTY,
3.       WHITE, BLACK,
4.       RED, YELLOW, BLUE,
5.       PINK, ORANGE, GREY, BROWN;
6. }

7. boolean isPrimaryColor(MyColor c) {
8.     return c >= MyColor.RED && c <= MyColor.BLUE;
9. }
```

Xebia
Business Integration Architects

Chained invocation



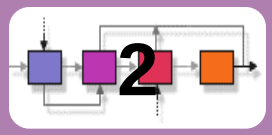
- Easy chaining of methods calls
- Allow void method to implicitly return ‘this’

Use case sample

```
1. class Factory {
2.       void setSomething(Something something) { ... }
3.       void setOther(Other other) { ... }
4.       Thing result() { ... }
5.  }
6.
7.  Thing thing = new Factory()
8.       .setSomething(something)
9.       .setOther(other)
10.      .result();
```

Xebia
Business Integration Architects

Extensions methods



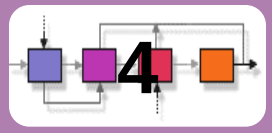
- Simulate a non existing interface method

Use case sample

```
1.import java.util.Collections;
2.List<String> list = …;
3.Collections.sort(list);

4.import static java.util.Collections.sort;
5.List<String> list = …;
6.list.sort();
```

Xebia
Business Integration Architects

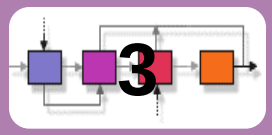
Catch improved




```
try {
     return klass.newInstance();
 } catch (InstantiationException | IllegalAccessException e) {
     throw new AssertionError(e);
 }
```

```
try {
     doable.doit();          // Throws several types
} catch (final Throwable ex) {
     logger.log(ex);
     throw ex;               // OK: Throws the same several types
}
```

Automatic ressource management blocks

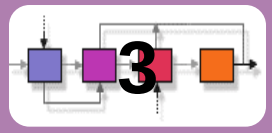


- **Automatically clean up resources at the end of the block**
  - No finally block is needed
  - Multi resources compliant

Use case sample

```
1. // Start a block using two resources, which will automatically
2. // be cleaned up when the block exits scope
3. do (BufferedInputStream bis = ...; BufferedOutputStream bos = ...) {

4.       ... // Perform action with bis and bos
5. }
```

Simplified CTOR



Simplified CTOR

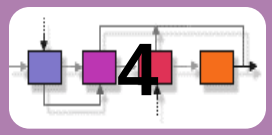
- `Map<String, List<String>> anagrams = new HashMap<>();`

Tabular notation for List and Map

- `List<String> list = new ArrayList<String>();`
- `list.add[1] = ''test'';`

Xebia
Business Integration Architects

Closures



- Captures the bindings of free variables in its lexical context.
- Closures support passing functions around and make many anonymous inner class use cases easier to use.
- Support additional high-level programming abstractions.

Without closures

```
lock.lock();
try {
    ++counter;
} finally {
    lock.unlock();
}
```

With closures

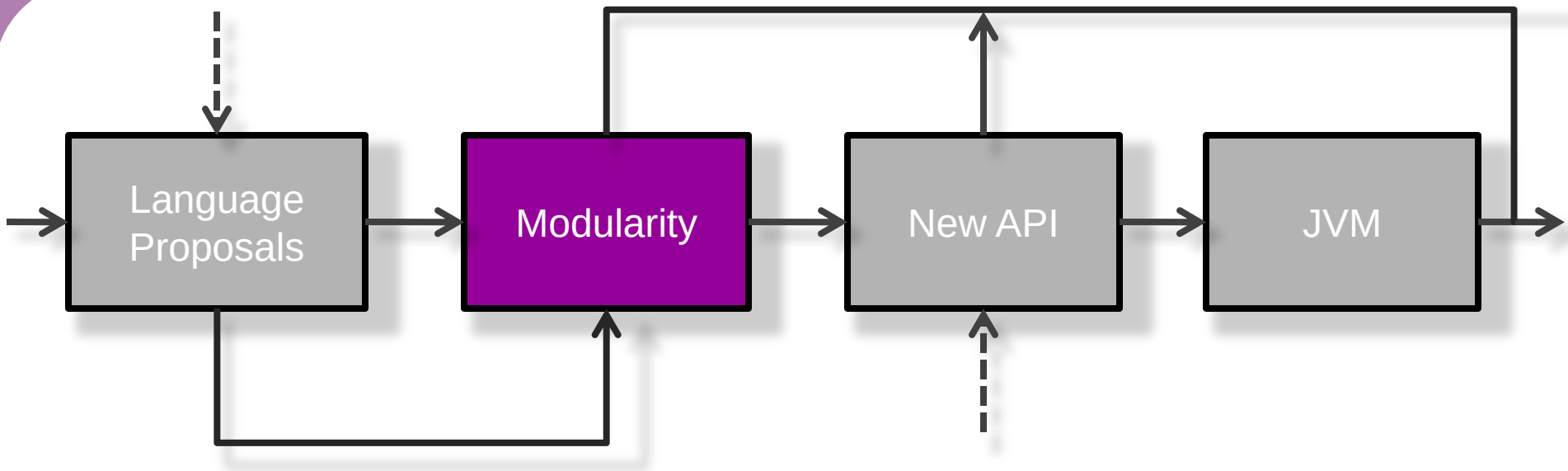
```
withLock(lock) {
  ++counter;
}
```

Code needed

```
public static <T,throws E extends Exception>
T withLock(Lock lock, {=>T throws E} block)
throws E {
  lock.lock();
  try {
    return block.invoke();
  } finally {
    lock.unlock();
  }
}
```

Xebia
Business Integration Architects

Please follow the guide …

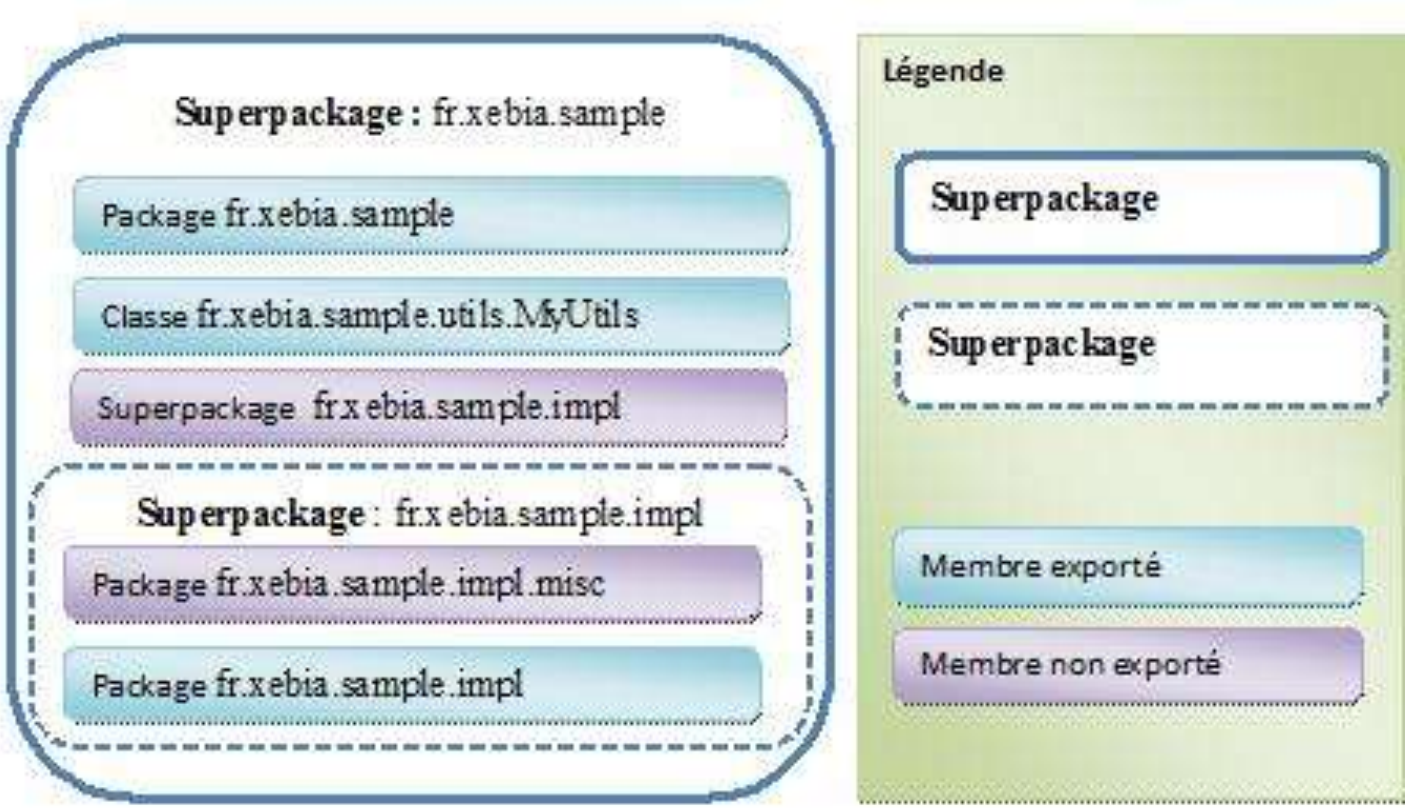


1. Langage proposals
2. **Modularity**
3. New API
4. JVM

Xebia
Business Integration Architects

Superpackages

- **Modularity at development time**



JSR 294 : Superpackages

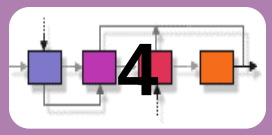
File: fr/xebia/sample/super-package.java

```
1. superpackage fr.xebia.sample {
2.         member package fr.xebia.sample;
3.         member package fr.xebia.sample.utils;
4.         member superpackage fr.xebia.sample.impl;
5.         export fr.xebia.sample.utils.MyUtil;
6.         export package fr.xebia.sample;
7. }
```

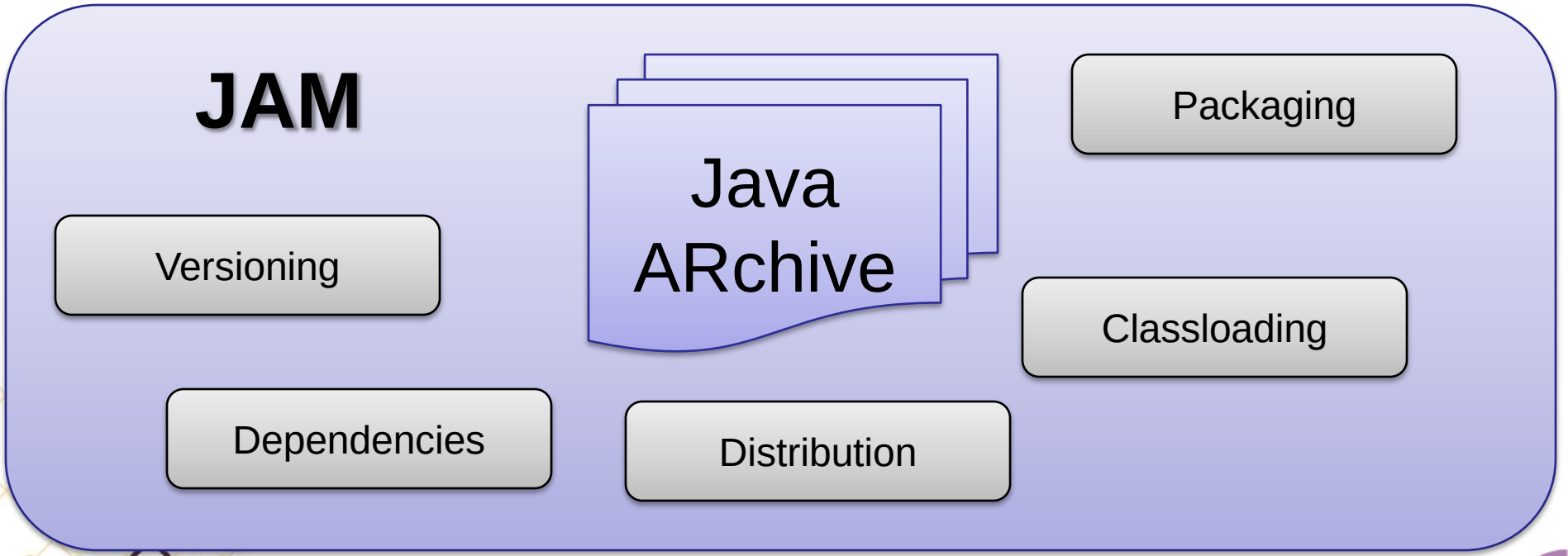
File: fr/xebia/sample/impl/super-package.java

```
1. superpackage fr.xebia.sample.impl member fr.xebia.sample {
2.          member package fr.xebia.sample.impl;
3.          member package fr.xebia.sample.impl.misc;
4.          export package fr.xebia.sample.impl;
5. }
```

JSR 277 : JAva Module System



- **Deployment and packaging**
  - New distribution format : bye bye JAR welcome JAM
  - Version and Dependency management
  - Storage module repository
  - Defines discovery, loading, and integrity mechanisms at runtime
  - Can specify which public / hidden portions

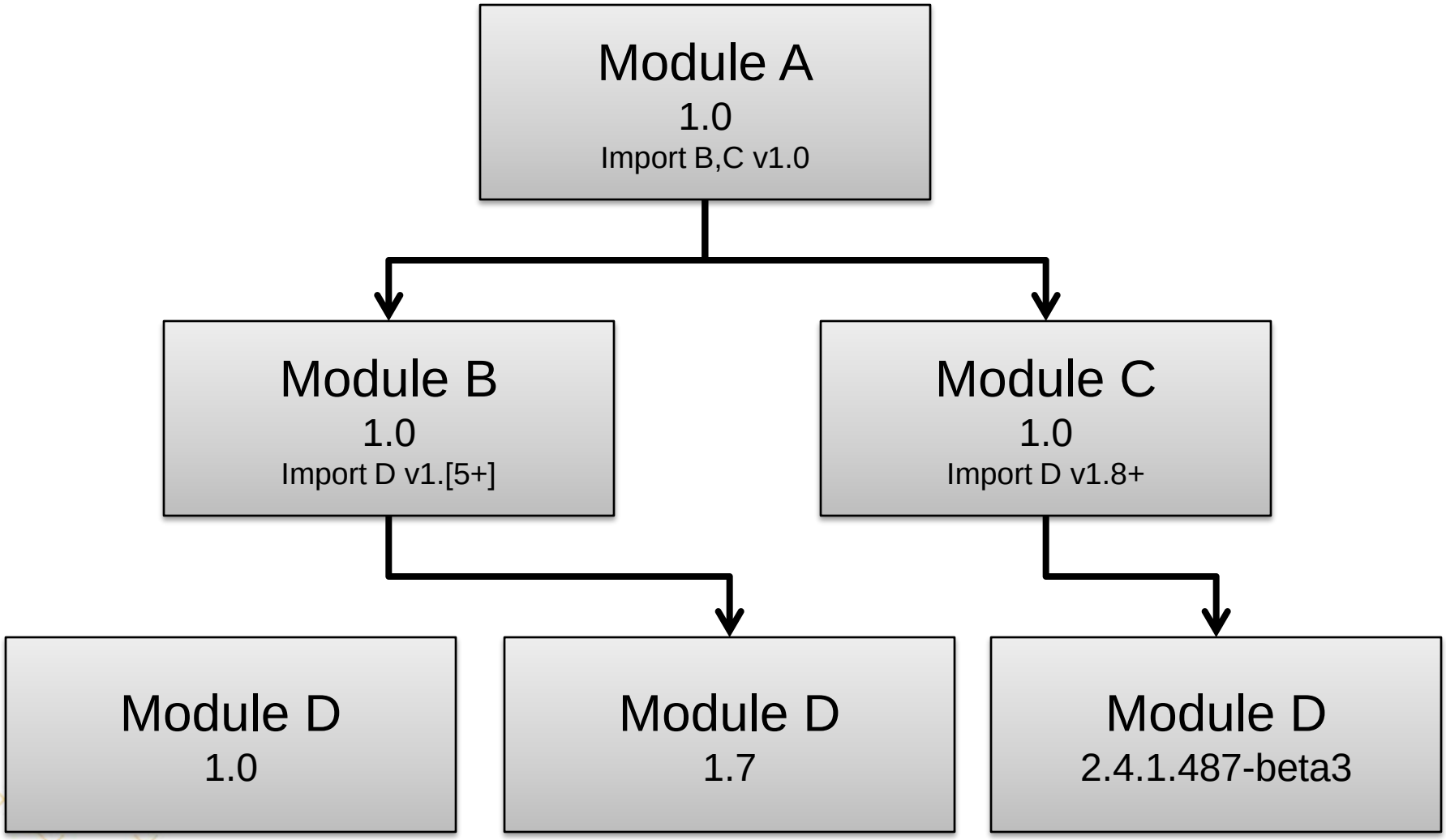


Java Module System

- **Versioning**
  - major[.minor[.micro[.update]]][-qualifier]
  - "1", "1.7", "1.7.0.1", "1.7.0.2", "1.7.2-rc " ...
- **Samples**

1+	Version 1 or after	(1 <= x)
1.5+	Version 1.5 or after	(1.5 <= x)
1.5.2+	Version 1.5.2 or after	(1.5.2 <= x)
1*	Every version of the family 1	(1 <= x < 2)
1.5*	Every version of the family 1.5	(1.5 <= x < 1.6)
1.5.2*	Every version of the family 1.5.2	(1.5.2 <= x < 1.5.3)
1.[5+]	Version 1.5 or after in the family 1.0	(1.5 <= x < 2.0)
1.5.[2+]	Version 1.5.2 or after in the family 1.5	(1.5.2 <= x < 1.6)
1.[5.2+]	Version 1.5.2 or after in the family 1.0	(1.5.2 <= x < 2.0)

Java Module System



Xebia
Business Integration Architects

Java Module System

- **JAM : JAR with**
  - Other JAR
  - Resources
  - Metadata
  - Can contain native code in /MODULE-INF/bin/<platform>/<arch>/
  - Can be specific to a particuliar platform
  - Can be compress with Pack200-gzip
- **Metadata**
  - File “MODULE-INF/METADATA.module”
  - Name / Imports / Class exports / Members
  - Version / Main Class / Resource Exports / Attributes
  - Generated at compilation time
  - Use superpackage

Java Module System

```
1. // Déclaration du superpackage :
2. @Version("1.0")       // Module Version
3. superpackage com.site.pack {   // Module Name

4.       // Module Import with version 1.0 minimum
5.       @VersionConstraint("1.0+")
6.       import com.site.name;

7.       // Module Import with version 1.5 family
8.       @VersionConstraint("1.5*")
9.       import com.site.xml;

10.      // packages  list
11.      member package com.site.pack, com.site.pack.impl;

12.      // Exported types
13.      export com.site.pack.*;
14.}
```

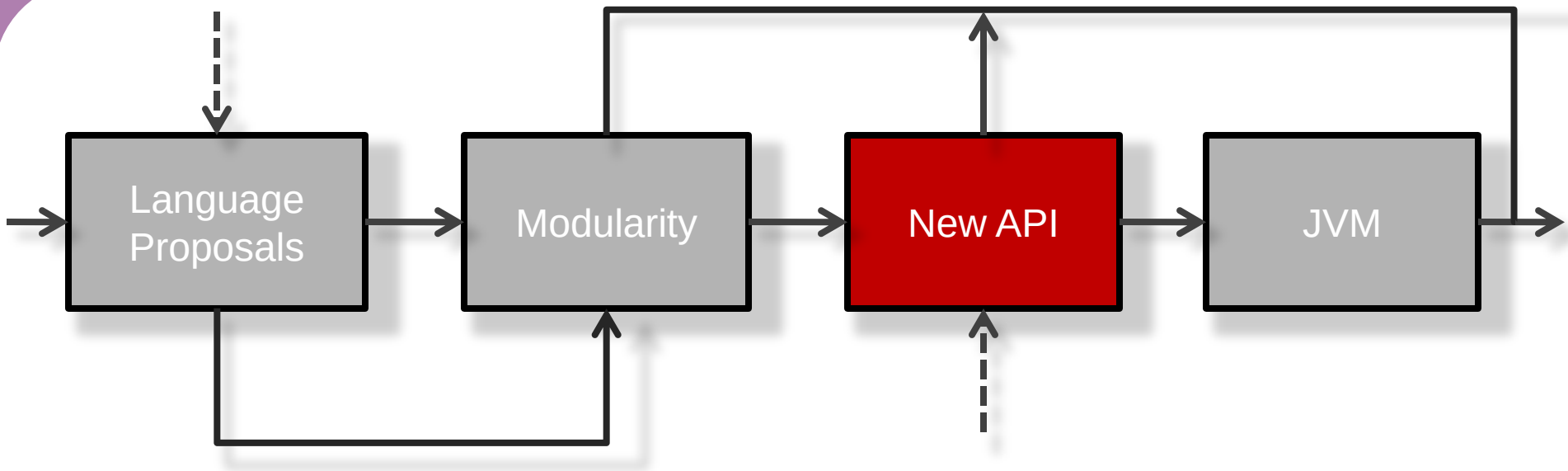
Xebia
Business Integration Architects

Java Module System

- **Each application**
  - Will have a standard Java API Repository (initialized by the JVM)
  - Could have others custom repositories
    - » Ex : global repository

- **Repository [local | remote]**
  - Module Storage
  - Module Search
  - Module Loading
    - » Application startup
    - » Runtime

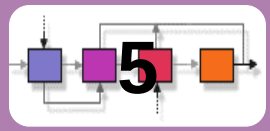
Please follow the guide …



1. Langage proposals
2. Modularity
3. **New API**
4. JVM

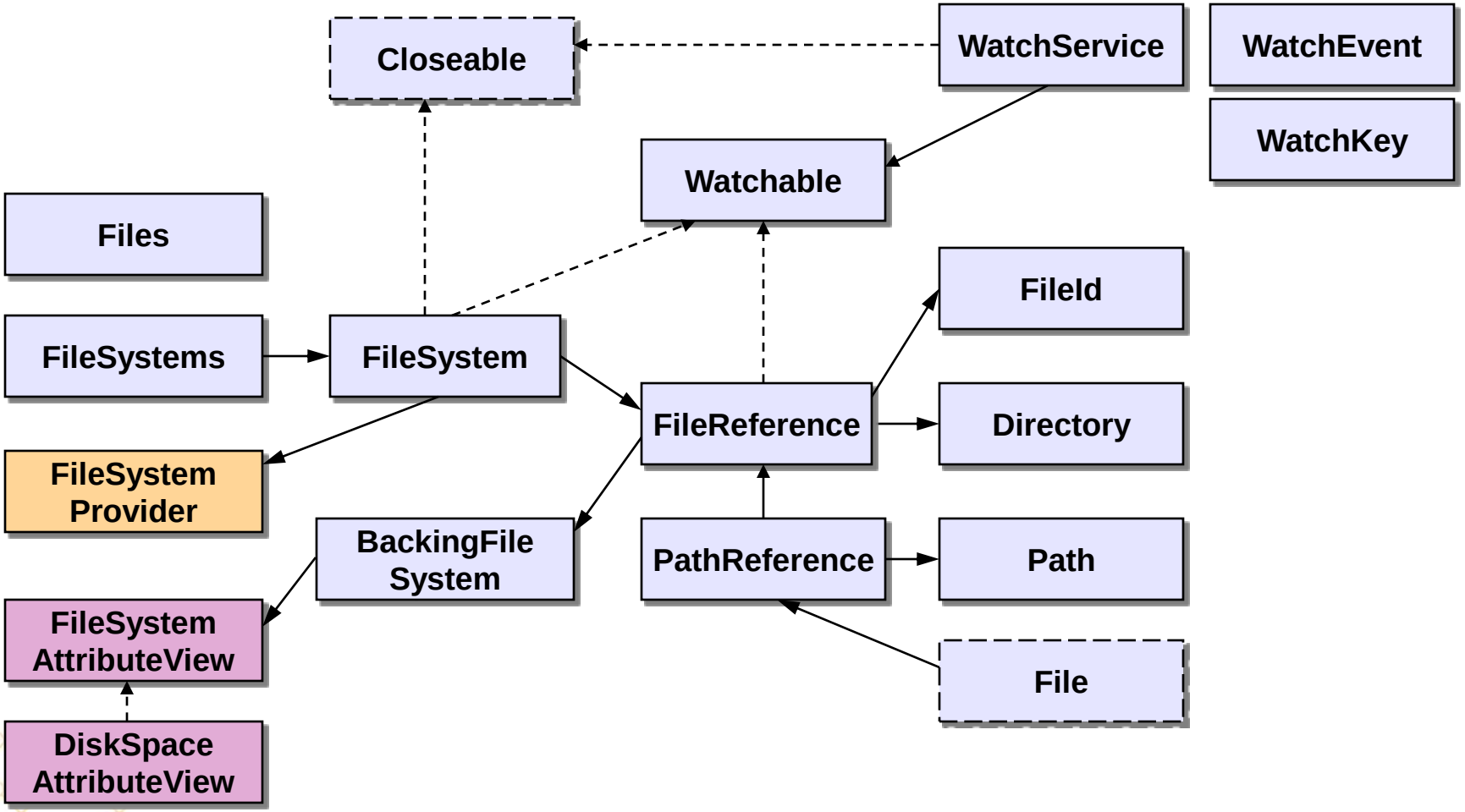
Xebia
Business Integration Architects

JSR 203 : Nio 2



- **Major features:**
  - New filesystem interface
    - Permissions
    - File attributes
    - Pluggable implementations
    - Escape to filesystem-specific APIs
  - API for asynchronous I/O on sockets and files
  - Completion of socket channels (multicast, options, etc)

JSR 203 : Nio 2



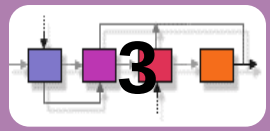
Xebia
Business Integration Architects

JSR 203 : Nio 2

Use case Sample

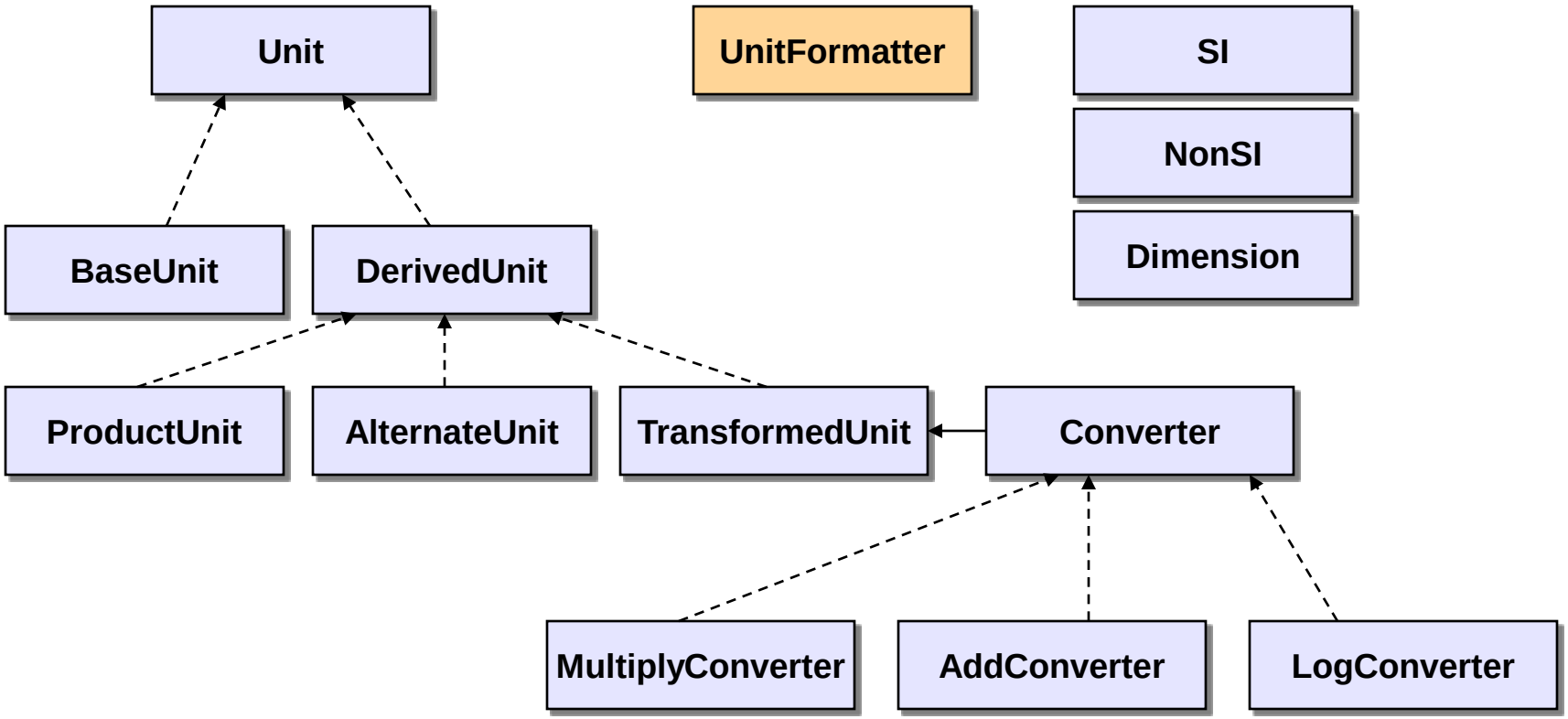
```
1. PathReference source =  PathReference.from("spongebob.jpg");
2. PathReference target =  PathReference.from("squarepants.jpg");

3. int flags = CopyFlag.COPY_ATTRIBUTES | CopyFlag.REPLACE_EXISTING;
4. source.copyTo(target, flags);
```

JSR 275 : Units and quantities



- **Units and quantities**
  - Checking of unit compatibility
  - Expression of a quantity in various units
  - Arithmetic operations on units
  - Concrete classes implementing the standard types of units (such as base, supplementary, and derived) and unit conversions.
  - Classes for parsing unit specifications in string form and for formatting string representations of quantities.
  - A database of predefined units.

JSR 275 : Units and quantities



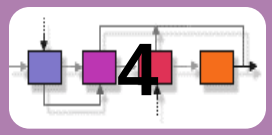
JSR 275 : Units and quantities

Use Case Sample

```
1.  // Conversion between units.
2.  KILO(METER).getConverterTo(MILE).convert(10)
3.  => 6.213711922373395
4.
5.  // Retrieval of system unit (identifies the measurement type).
6.  REVOLUTION.divide(MINUTE).getSystemUnit()
7.  => rad/s
8.
9.  // Dimension checking (allows/disallows conversions)
10. ELECTRON_VOLT.isCompatible(WATT.times(HOUR))
11. => true

12. // Retrieval of unit dimension (depends upon the current model).
13. ELECTRON_VOLT.getDimension()
14. => [L]² ·[M]/[T]²
```

Xebia
Business Integration Architects

JSR 310 : Date and time API



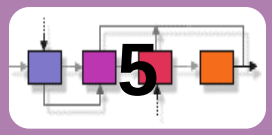
- **Based on Joda Time**
- **Objectives**
  - Immutability
  - Fluent API
  - Clear, Explicit and Expected
  - Extensible
- **Concepts**
  - Instant – timeline based
  - Human-scale datetimes – field based (fully or partially)
  - Interval – between 2 datetimes
  - Duration – scientific definition of time
  - Period – such as 3 days, 5 hours and 2 minutes

JSR 310 : Date and time API

```
1. public boolean isAfterPayDay(DateTime datetime) {
2.     if (datetime.getMonthOfYear() == 2) { // February is month 2!!
3.         return datetime.getDayOfMonth() > 26;
4.     }
5.     return datetime.getDayOfMonth() > 28;
6. }

7. public Days daysToNewYear(LocalDate fromDate) {
8.     LocalDate newYear = fromDate.plusYears(1).withDayOfYear(1);
9.     return Days.daysBetween(fromDate, newYear);
10.}

11.public boolean isRentalOverdue(DateTime datetimeRented) {
12.    Period rentalPeriod = new Period().withDays(2).withHours(12);
13.    return datetimeRented.plus(rentalPeriod).isBeforeNow();
14.}

15.public String getBirthMonthText(LocalDate dateOfBirth) {
16.    return dateOfBirth.monthOfYear().getAsText(Locale.ENGLISH);
17.}
```

Xebia
Business Integration Architects

JSR 166y : Concurrency Utilities

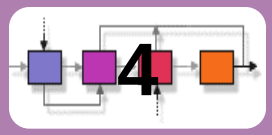


- **JSR 166 : Java 5.0 Lea Doug Concurrency**
- **JSR 166x : Java 6.0**
  - Deque / BlockingDeque / NavigableSer / LinkedBlokingDeque …
- **JSR 166y:**
  - Fork / Join library
  - Parallel programming

```
Result solve(Problem problem) {
      if (problem is small)  {
            directly solve problem
      } else {
            split problem into independent parts
            fork new subtasks to solve each part
            join all subtasks
            compose result from subresults
      }
}
```

Xebia
Business Integration Architects

JSR 225 : Xquery API for Java



- **Executing XQuery calls and working with results.**
- **XQJ is to XQuery what JDBC is to SQL.**
- **XQJ**
  - JDBC concepts : DataSource, Connection, PreparedExpression …
  - Specific concepts : static / dynamic phrases, XML Oriented Retrival …

Sample 1

```
XQDataSource xqjd = new DDXQDataSource();
XQConnection xqjc = xqjd.getConnection();
XQExpression xqje = xqjc.createExpression();
XQSequence xqjs = xqje.executeQuery("'Hello World!'");
```

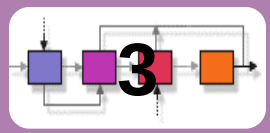
```
xqjs.writeSequence(System.out, null);
xqjc.close();
```

Sample 2

```
xqje.executeQuery("doc('orders.xml')//order[id='174']");
```

Xebia
Business Integration Architects

JSR 284 : Resource Consumption Managmt

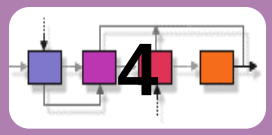


- **Partitioning resources**
- **Querying resource availability**
- **Exposing various kinds of resources**

- **Resources :**
  - heap memory size,
  - CPU time,
  - open JDBC connections, etc.

- **Goal :**
  - allow resources to be managed so that multiple applications might run simultaneously in a JVM and be given finite amounts of memory, cpu, etc.

JSR 308 : Annotations on Java Types



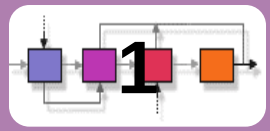
Class

- **for generic type arguments to parameterized classes:**
  Map<@NonNull String, @NonEmpty List<@Readonly Document>> files;
- **for type parameter bounds and wildcards:**
  class Folder<F extends @Existing File> { ... }
  Collection<? super @Existing File>
- **for class inheritance:**
  class UnmodifiableList<T> implements @Readonly List<@Readonly T> { ... }
- **for throws clauses:**
  void monitorTemperature() throws @Critical TemperatureException { ... }
- **for typecasts:**
  myString = (@NonNull String) myObject;
- **for type tests:**
  boolean isNonNull = myString instanceof @NonNull String;
- **for object creation:**
  new @NonEmpty @Readonly List<String>(myNonEmptyStringSet)

Method

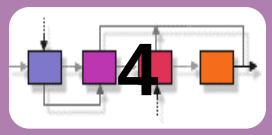
- **for generic type arguments in a generic method or constructor invocation:**
  o.<@NonNull String>m("...");

JSR 260 : Javadoc Technology update



- **Updates**
  - categorization of methods and fields by their target use
  - semantical index of classes and packages
  - distinction of static, factory, deprecated methods from ordinary methods
  - distinction of property accessors from ordinary methods
  - combining and splitting information into views
  - embedding of examples, common use-cases along with Javadoc

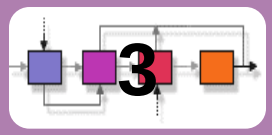
JSR 296 : Swing App Framework



- **Common infrastructure for desktop applications**
  - » Swing applications will be easier to create

- **Functionalities**
  - Remembering **state between sessions**.
  - Easier **managing of actions**, including running as background tasks and specifying blocking behavior.
  - Enhanced **resource management**, including resource injection for bean properties.
  - Providing the Java Desktop Application **project template** (Basic / Database)
  - Integration of framework features in the **IDE's GUI Builder**.
  - Generating your application's UI text and other resources in .properties files.
  - Providing a special **property editor** for actions
  - Automatically **packaging** the Swing Application Framework library into your project's dist/lib folder when you build your application in the IDE.

JSR 303 : Bean Validation



- **How to unsure constaints**
  - Database Schema constraints
    1. title varchar(30) not null
  - Business/Service level constraints
    1. if (document.getTitle() == null || document.getTitle().length() > 30) {
    2. throw new BusinessException("Document title invalid");
    3. }
  - Presentation level constraints
    1. if (documentForm.getTitle() == null || documentForm.getTitle().length() > 30) {
    2. throw new BusinessException("Document title invalid");
    3. }
  - Client-side contraints
    - // Your favorite DHTML + javascript OR
    - // Your favorite JSF component library
    - // Your favorite other web framework

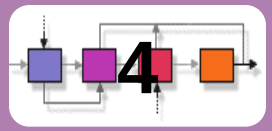
JSR 303 : Bean Validation

- **Standardize the constraints Metadata Model**
  - Primary through annotations
  - XML for complex cases
  - API to query it
- **Standardize the validation API**
  - Layer agnostic
- **Ability to write custom constraints**
  - Annotation
  - Validation logic
- **i18n**

JSR 303 : Bean Validation

- **With These API and Metadata…**
  - DDL update when generated by ORMs
    - » e.g., Hibernate Validator and Hibernate Core integration
  - Entity validation on insertion/update by Java Persistence API
  - WebBeans (Seam) validation
    - » Business level
    - » Presentation level
  - JSF technology validation at the client side from Metadata
  - Your code triggering validation
- **Annotations**
  - Logic : @NotNull / @NotEmpty / @AssertFalse/True …
  - Range : @Min / @Max / @Digits / @Length / @Size …
  - Business : @Email / @CreditCard …
  - Custom …

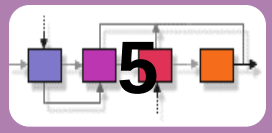
JSR 295 : Bean Binding



- **Bean Binding**
  - making of connections between beans
  - Keeps two properties of two objects in sync
  - Source properties are specified using Java Expression Language (EL) syntax:
    - » ex: “${customer}” or “${employee.salary}”
  - Does not require special object types, endpoints

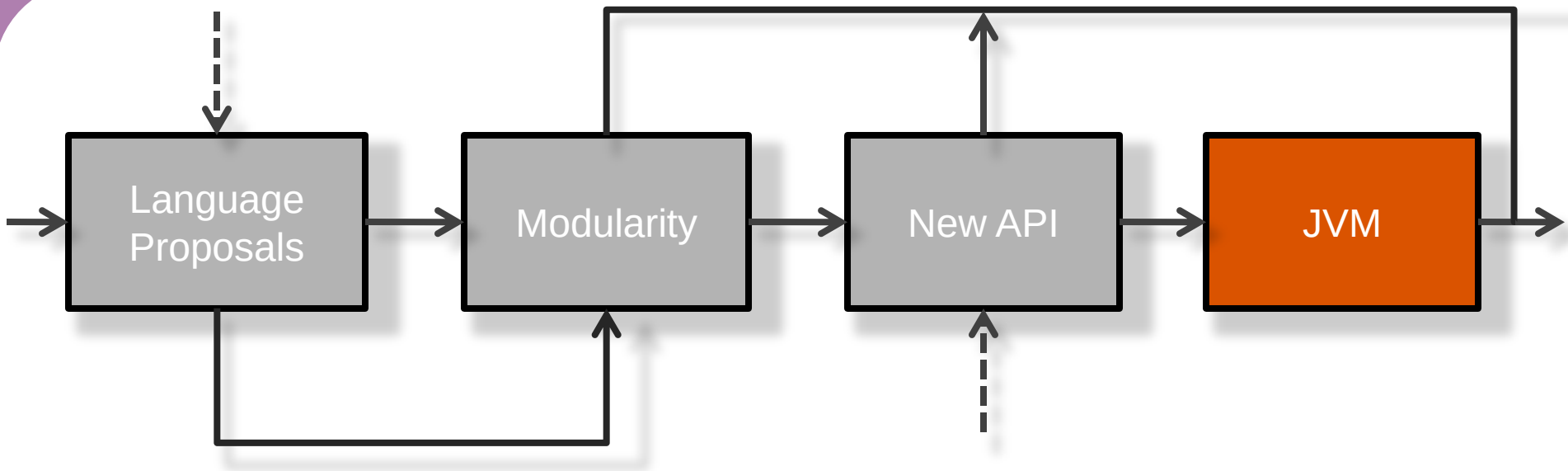
```
1. bind(Object source,
2.         String sourcePath,
3.         Object target,
4.         String targetPropertyName);
```

JMX



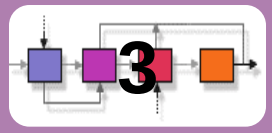
- **JSR 255 - JMX 2.0**
  - Retrofit with generics
  - Use annotations
  - Make Open MBeans easier to use
  - Generalize monitors to support non-simple types
  - Cascaded/federated MBean servers
- **JSR 262 - Web services connector**

Please follow the guide …

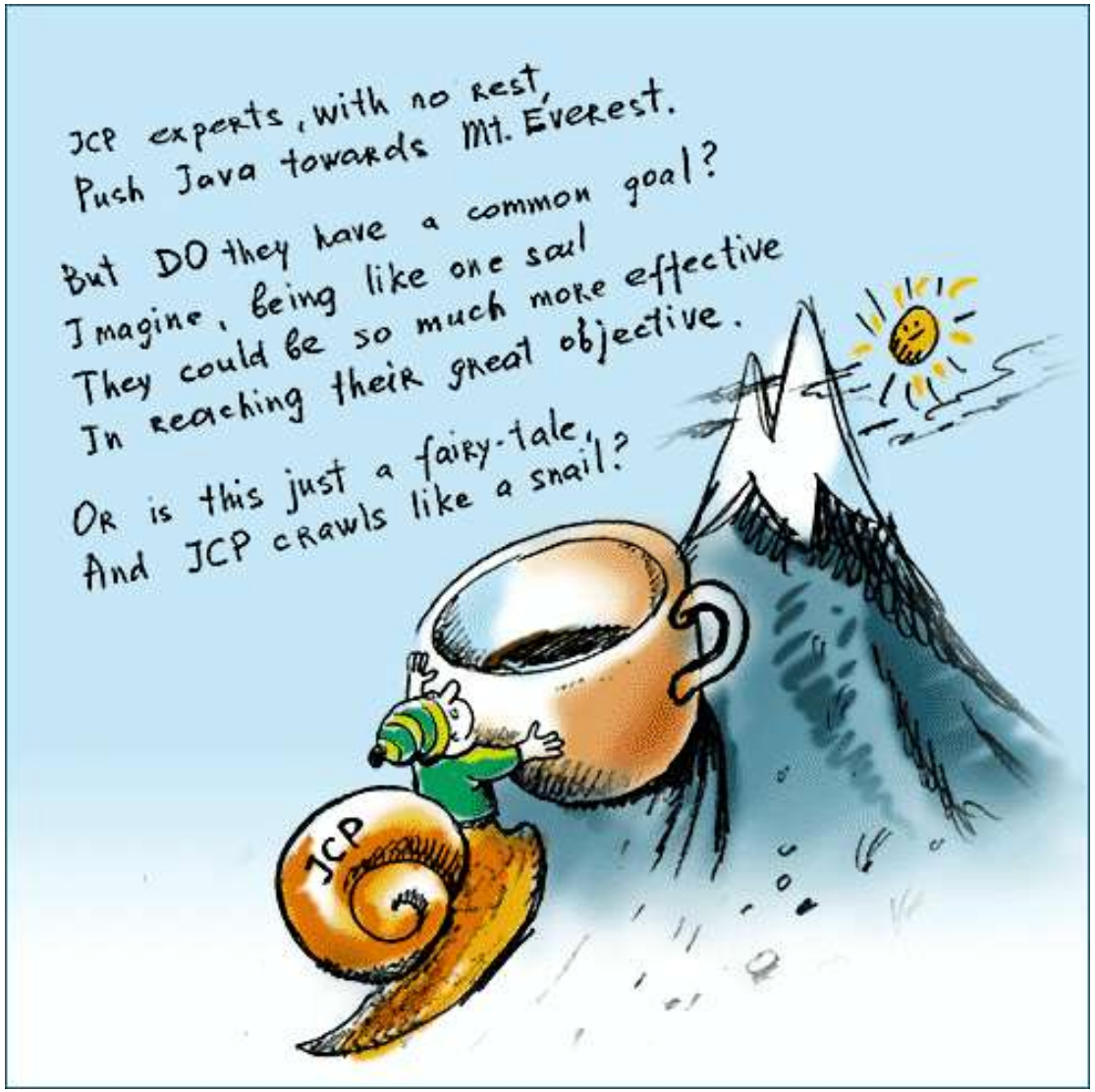


1. Langage proposals
2. Modularity
3. New API
4. **JVM**

Xebia
Business Integration Architects



- **JVM**
  - Bytecode invokedynamic in order to support dynamic langage
  - Tiered compilation
    - » uses a mix of the client (JIT) and server hot spot compilers
      - – the client engine improves startup time by using JIT over interpreted
      - – the server hot spot compiler gives better optimization for hot spots over time.
  - G1 Garbage Collector
    - » divides the entire space into regions
    - » allows a set of regions to be collected rather than split the space into an arbitrary young and old generation
    - » Sun intends to make this new collector the default in Java 7
  - More Script Engine
    - » Java 6 : JavaScript Rhino
    - » Java 7 : JRuby, Jython, Beanshell ?



Xebia
Business Integration Architects