



JAVA™ SE 7 LANGUAGE FEATURES

Peter Ahé

Alex Buckley

Sun Microsystems

Who we are

- Peter Ahé
 - > Java Compiler Tech Lead
 - > JSR 199 (Compiler API) Spec Lead
- Alex Buckley
 - > Java Language & VM Specification Editor
 - > JSR 294 (Improved Modularity) Co-Spec Lead
- Thanks for inviting us!

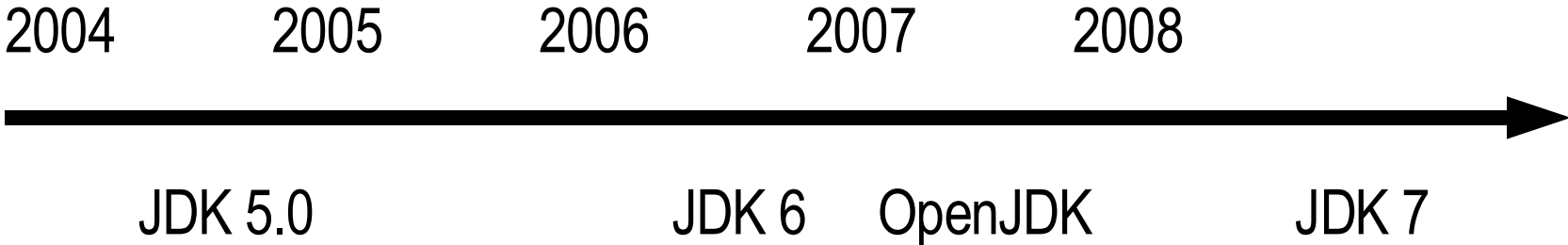
Agenda

- Where is Java?
- Potential language features
- Language design constraints
- The Kitchen Sink Language

Agenda

- **Where is Java?**
- Potential language features
- Language design constraints
- The Kitchen Sink Language

Where Java is



What Java is

- The Java programming language is:
- general-purpose,
concurrent,
class-based,
object-oriented language.
- strongly-typed.
- a relatively high-level language.
 - > Java Language Specification, Chapter 1

What Java is

- The Java language principles:
 - > Reading is more important than writing
 - > Simplicity matters
 - > One language, spoken the same everywhere
- The Java community:
 - > Cares about compatibility
 - > Solves problems in a massive design space

What Java isn't

- Aspect-oriented
- Dynamically typed
- Scripting language

- Keeping something out of the language does not imply keeping it out of the platform
 - > JRuby
 - > Groovy
 - > F3
 - > Scala

Agenda

- Where is Java?
- **Potential language features**
- Language design constraints
- The Kitchen Sink Language

Potential language features

- The information in this presentation concerns forward-looking statements based on current expectations and beliefs about future events that are inherently susceptible to uncertainty and changes in circumstances etc etc etc etc etc etc etc

JDK 5.0 Language Themes

- Readability
 - > Enhanced for loop, varargs, static import, boxing
- Expressibility
 - > Annotations, enumerated types
- Safety
 - > Generic types

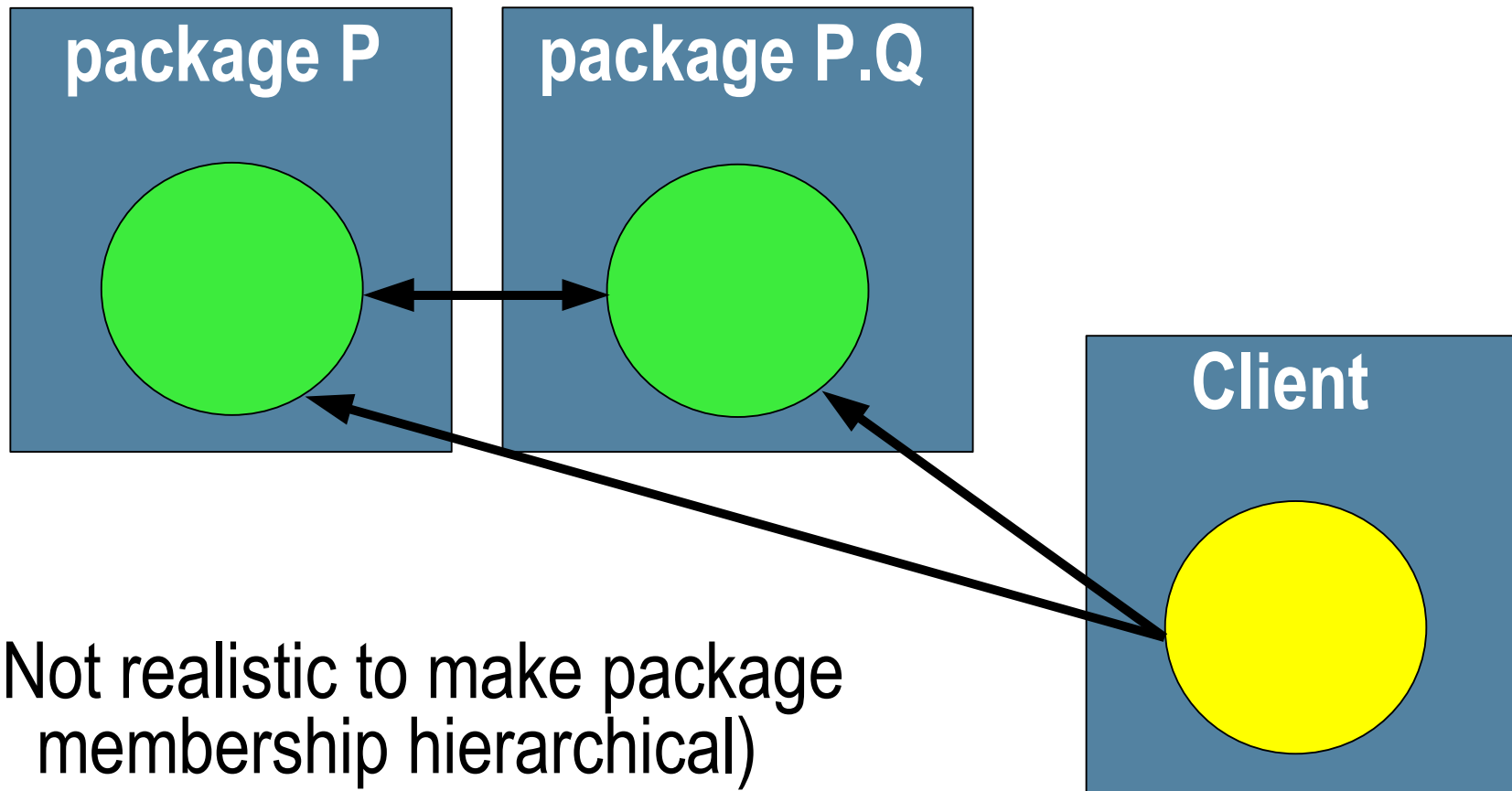
Tentative JDK 7 Language Themes

- Modularity
 - > Packages++
- Abstraction
 - > Anonymous classes++
- Consistency
 - > Erase erasure

Modularity: Motivation

- Package names are hierarchical
- Package membership is not hierarchical
 - > A type defined in package P.Q is not in package P
- Package visibility is not hierarchical
 - > Code in P.Q cannot access package-visible members in P
- Result: Too much is declared public

Modularity: Package shortcomings

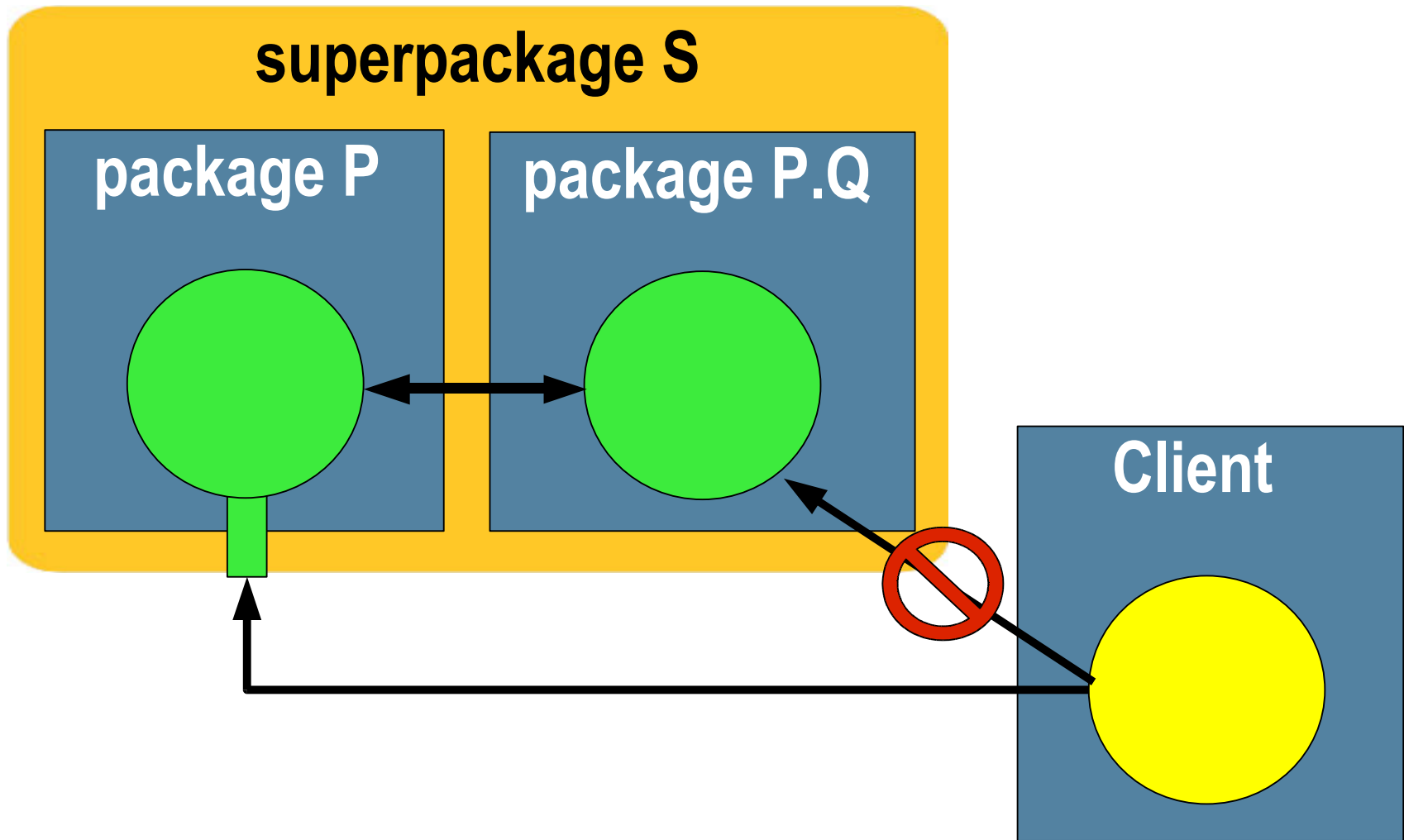


Modularity: Superpackages

- Aim: Language-level modules, independent of deployment mechanism

```
super package java {  
  
    member java.util;  
    member java.io;  
    member sun.misc;           // Impl detail  
  
    export java.util.*;       // Public API  
    export java.io.*;  
  
}
```

Modularity: Superpackages



Modularity: Status

- JSR 294 Expert Group about to start work
- May also handle separate compilation
- Tool support will be key
- Mailing lists could be publicly readable

- Related to other JSRs
 - > JSR 294: development-time modules
 - > JSR 277, 291: deployment-time modules

```
Iterator<String> it = strings.iterator();
while (it.hasNext()) {
    System.out.format("\"%s\"", size=%s%n",
                    it.next(),
                    it.next().length());
}
```

```
Iterator<String> it = strings.iterator();  
while (it.hasNext()) {  
    System.out.format("\"%s\"", size=%s%n",  
                      it.next(),  
                      it.next().length());  
}
```

```
for (String s : strings) {  
    System.out.format("\'%s\'", size=%s%n",  
                      s,  
                      s.length());  
}
```

```
InputStream is = ...;  
try {  
    ...  
} finally {  
    is.close();  
}
```

```
InputStream is = ...;
OutputStream os = ...;
try {
    ...
} finally {
    os.close();
    is.close();
}
```

```
InputStream is = ...;
OutputStream os = ...;
try {
    ...
} finally {
    os.close();
    is.close();
}
```

Abstraction: Motivation

- Boilerplate code becomes unwieldy
- Need for new statements
 - > Libraries aren't up to the task
- Code blocks in Java are not first-class
 - > Cannot pass blocks of code as arguments

```
InputStream is = ...;
try {
    OutputStream os = ...;
    try {
        ...
    } finally {
        os.close();
    }
} finally {
    is.close();
}
```

```
with (InputStream is : ...)
with (OutputStream os : ...) {
    ...
}
```

Abstraction: Status

- Numerous proposals, different aims
 - > BGGGA
 - > CICE
 - > C3S
 - > FCM

Black Hole Theory of Design

“Closures feel to me like one of these design Black Holes. When inner classes were designed, we wanted to avoid the complexity of closures. But this brought about oddness and tension of it's own that has left me less than happy with inner classes. So, should we just give in to the force of gravity and go the rest of the way?”

James Gosling
Sun Microsystems

Generic Types and Compatibility

- Source compatibility
 - > Nice to have...
- Binary compatibility
 - > Must have!
- Migration compatibility
 - > Existing programs must continue to work
 - > Existing libraries must be able to use generic types
 - > Must have!

Erasure

```
new ArrayList<String>()
```



```
new ArrayList()
```

Disadvantages of Erasure

- Dual model makes learning harder
- Lack of generic types at runtime prevents
 - > Some type tests
 - > Arrays of generic types (creation, not declaration)
 - > Accurate reflection/serialization

Reify

Regard an abstract concept as real

Other features

- Property declaration and access
- Annotations on types (JSR 308)
- XML literals
 - > Part of a wider XML feature set, including navigation
- Shorter syntax for common operations
 - > Collections, BigDecimal
- Improved switch statement
 - > Strings, coverage checking of enum constants

Local variable declarations

- `m := new HashMap<String,Integer>();`
 - `var m = new HashMap<String,Integer>();`
 - `final m = new HashMap<String,Integer>();`
- `Map<> m = new HashMap<String,Integer>();`
 - `Map<String,Integer> m = new HashMap<>();`
 - `Map<String,Integer> m = new HashMap();`
 - `Map<String,Integer> m = HashMap.new();`
- `Map<String,Integer> m = new();`
 - `HashMap<String,Integer> m();`

New features: Our view

- Code abstraction
- Local variable declarations
- Integration with scripting languages
 - > Multiline literals
 - > JSR 292 *invokedynamic* bytecode instruction
 - > JSR 223 Scripting for the Java Platform
- Worth tidying up some loose ends

Agenda

- Where is Java?
- Potential language features
- **Language design constraints**
- The Kitchen Sink Language

Design constraints

- Ask yourself: Could I not do it in the language?
- If it's in the language, everyone has to learn it

Design constraints: Syntax

- Unambiguous grammar
 - > Ideally context-free
- Syntax does matter
 - > Tasteful
 - > Do not introduce new keywords (assert, enum)
 - > Context-sensitive keywords may be acceptable
 - import java.util.List<String> **as** StringList;
 - super package ... { **export** ... }

Design constraints: The right balance

- Syntax does **not** matter
 - > Idea – Syntax = Idea
- Language constructs should be concepts, not just syntactic sugar
- Concepts should be language constructs, not annotations
 - > Program metadata != Language data
 - > Annotations are convenient for prototyping language features

Design constraints: Implications

- Semantics should be context-free
 - > "Tennent's Correspondence Principle"
 - > Orthogonality matters
- Classfile format changes
- VM support
- Toolability

Agenda

- Where is Java?
- Potential language features
- Language design constraints
- **The Kitchen Sink Language**

Kitchen Sink Language

“Throw stuff into the kitchen sink without thinking too hard about whether or not its a good idea. Let folks kick the tires. Those experiences then inform the choice of which features go into the standard.”

James Gosling
Sun Microsystems

Kitchen Sink Language

- <https://ksl.dev.java.net/>
- A serious language proposal has:
 - > Specification
 - > Implementation
 - > Test cases
- OpenJDK and KSL do not bypass JCP

Final Thoughts

- Java 7 is likely to increase functionality and consistency in significant ways
- Continued focus on broadest possible market
- Feature list will be determined through the JCP
- Serious open source proposals are good

Links

- <http://blogs.sun.com/abuckley/>
- <http://blogs.sun.com/ahe/>
- <http://planetjdk.org/>
- <https://openjdk.dev.java.net/>
- <https://ksl.dev.java.net/>

- Superpackages: JavaOne2006 TS-3885
- XML literals: JavaOne2006 TS-3441



JAVA™ SE 7 LANGUAGE FEATURES

Peter.Ahe@Sun.COM

Alex.Buckley@Sun.COM