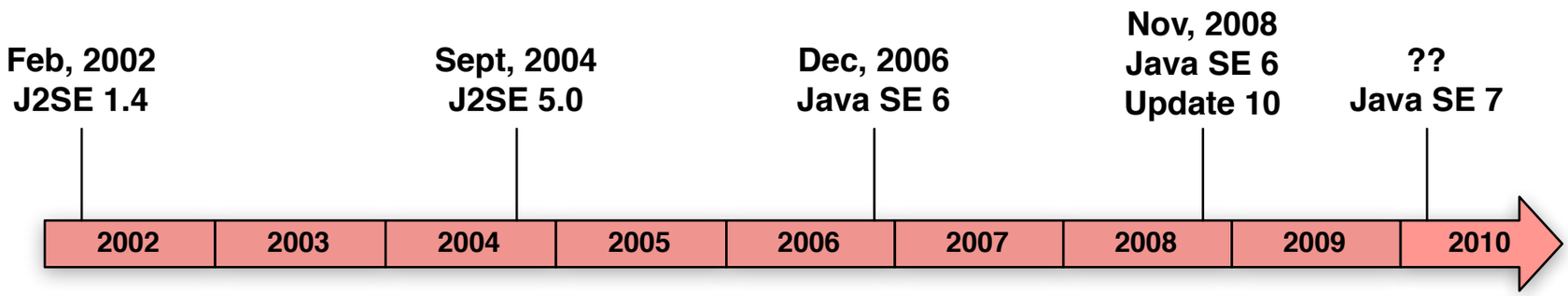
Java SE 7 Preview

Alex Miller

Blog: http://tech.puredanger.com
Twitter: http://twitter.com/puredanger

History



assert
regular expressions
IPv6
NIO
Logging API
Image I/O API
XML/XSLT
Security/crypto
Java Web Start

generics
annotations
autoboxing
enumerations
synth LAF
enhanced for loop
varargs
Java Memory Model
concurrent library
static imports

scripting support
JAX-WS
Java compiler API
pluggable annotations
SwingWorker, etc

Java Kernel
Java Quick Starter
Nimbus LAF
Java Plugin NG

Full Menu

Modularity
- Project Jigsaw
- JSR 294 Superpackages

Libraries
- JSR 203 NIO2
- JSR 275 Units and Quantities
- JSR 310 Date and Time API
- JSR 166 Concurrency Utilities
- JSR 225 XQuery API for Java
- JSR 284 Resource Consumption Mgmt

Swing
- JSR 296 Swing Application Framework
- JSR 295 Beans Binding
- JSR 303 Beans Validation
- Java Media Components

JMX
- JSR 255 JMX 2.0
- JSR 262 Web Services Connector

Tools
- JSR 326 Post-mortem JVM Diagnostics API
- JSR 260 Javadoc Update

Types and Generics
- Reified Generics
- Type Literals
- JSR 308 Annotations on Java Types
- Type Inference

Language Proposals
- Closures
- Automatic Resource Mgmt Blocks
- Language level XML support
- JavaBean property support
- BigDecimal operator support
- Strings in switch statements
- Comparisons for Enums
- Chained invocation
- Extension methods
- Improved catch
- Null handling improvements

JVM
- invokedynamic
- Tiered compilation
- G1 garbage collector

Who Decides?

Danny Coward

...and of course the
Java SE 7 JSR
Expert Group

Status du jour

Modularity
- Project Jigsaw YES
- JSR 294 Superpackages YES

Libraries
- JSR 203 NIO2 YES
- JSR 275 Units and Quantities NO?
- JSR 310 Date and Time API HMM
- JSR 166 Concurrency Utilities YES
- JSR 225 XQuery API for Java YES?
- JSR 284 Resource Consumption Mgmt NO

Swing
- JSR 296 Swing Application Framework YES
- JSR 295 Beans Binding NO
- JSR 303 Beans Validation HMM
- Java Media Components YES?

JMX
- JSR 255 JMX 2.0 YES
- JSR 262 Web Services Connector YES

Tools
- JSR 326 Post-mortem JVM Diagnostics HMM
- JSR 260 Javadoc Update NO

Types and Generics
- Reified Generics NO
- Type Literals NO
- JSR 308 Annotations on Java Types YES
- Type Inference YES?

Language Proposals
- Closures NO
- Automatic Resource Mgmt Blocks NO
- Language level XML support NO
- JavaBean property support NO
- BigDecimal operator support NO
- Strings in switch statements HMM
- Comparisons for Enums HMM
- Chained invocation HMM
- Extension methods HMM
- Improved catch YES?
- Null handling improvements YES?

JVM
- invokedynamic YES
- Tiered compilation HMM
- G1 garbage collector YES
- Compressed pointer 64 bit VM YES

Focus

Modularity

- Project Jigsaw
- JSR 294 Superpackages

Libraries

- JSR 203 NIO2
- JSR 310 Date and Time API
- JSR 166 Concurrency Utilities

Swing

- JSR 296 Swing Application Framework
- JSR 303 Beans Validation

JMX

- JSR 255 JMX 2.0
- JSR 262 Web Services Connector

Language Proposals

- Strings in switch
- Comparisons for Enums
- Chained invocation
- Extension methods
- Improved catch
- Null handling
- Type Inference

JVM

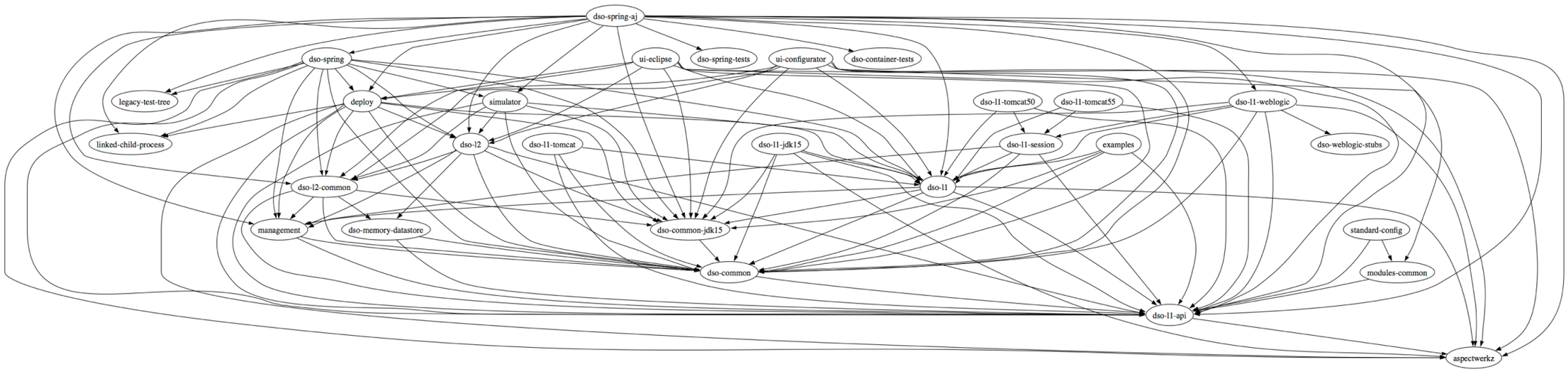
- invokedynamic

Java Modularity

- JSR 294: http://jcp.org/en/jsr/detail?id=294
- Project: http://openjdk.java.net/projects/modules

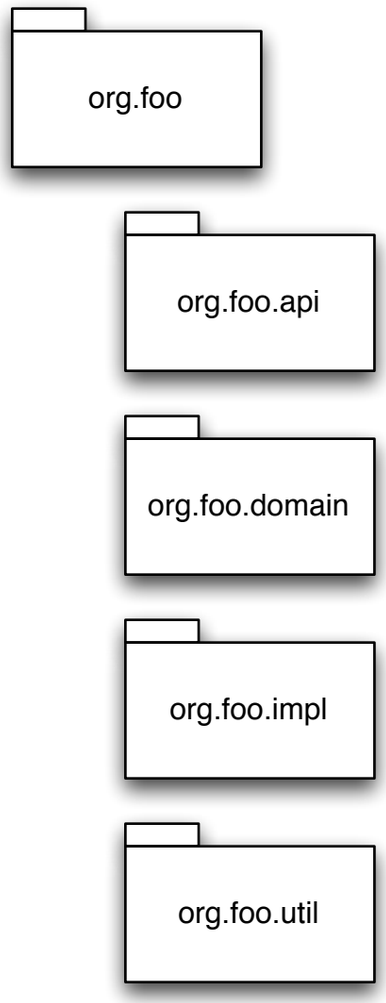
JAR HELL

Jar Hell in Action



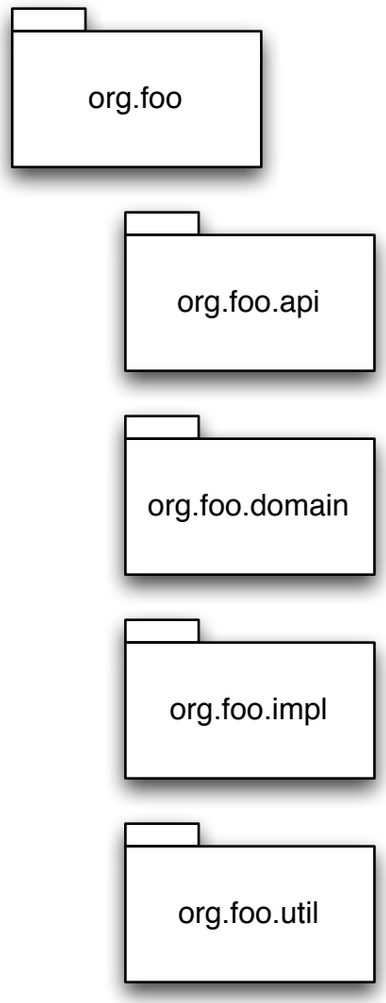
Dependency management is one of the most important (and challenging) features of modern software development

Module Development



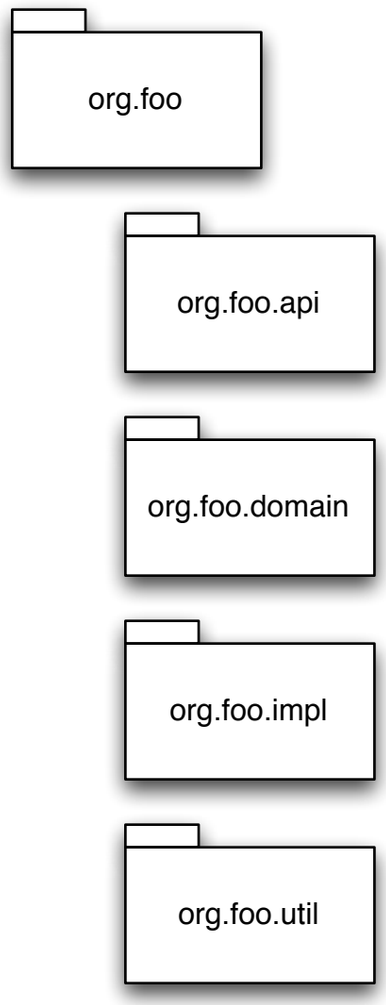

```
// org/foo/api/FooFighters.java:

module  org.foo;
package org.foo.api;

public  class FooFighters {
  public static FooFighters newBand() {
    return new org.foo.impl.FooFightersImpl();
  }
}
```

Module Development




```
// org/foo/impl/FooFightersImpl.java:

module  org.foo;
package org.foo.impl;

module  class FooFightersImpl
   implements FooFighters {

   // etc
}
```

Module Development




```
// org/foo/module-info.java:

@Version("2.3")
@MainClass("org.foo.Foosball")
@ImportModules {
  @ImportModule(name="java.se.core",
                version="1.7+")
  @ImportModule(name="org.bar",
                version="1.0", reexport="true")
}
@ExportResources({"org/foo/icons/**})
module org.foo;
```

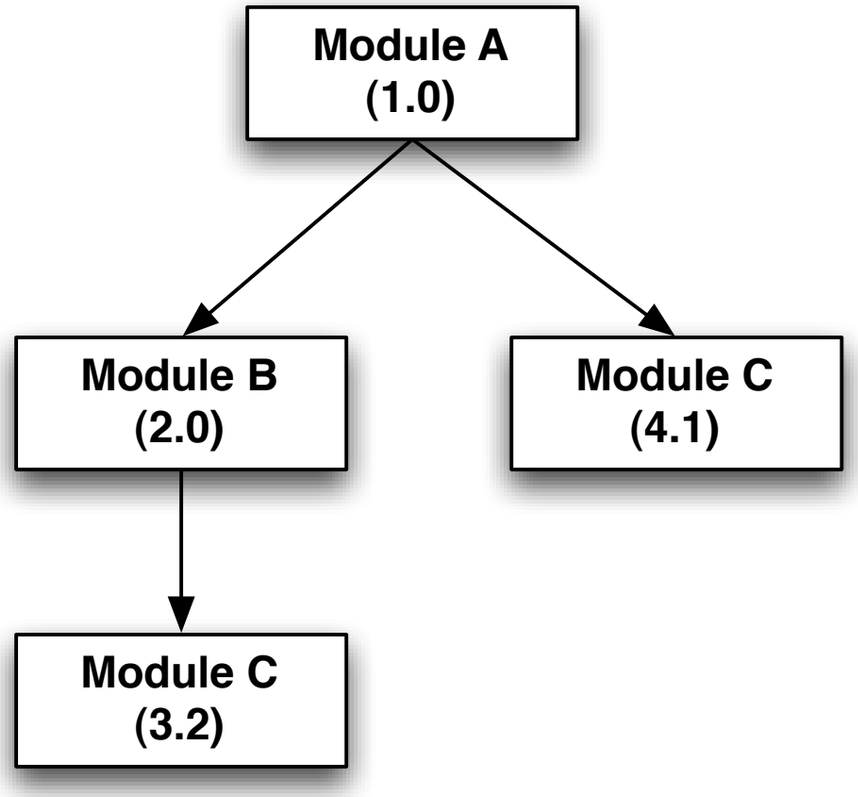
Packaging

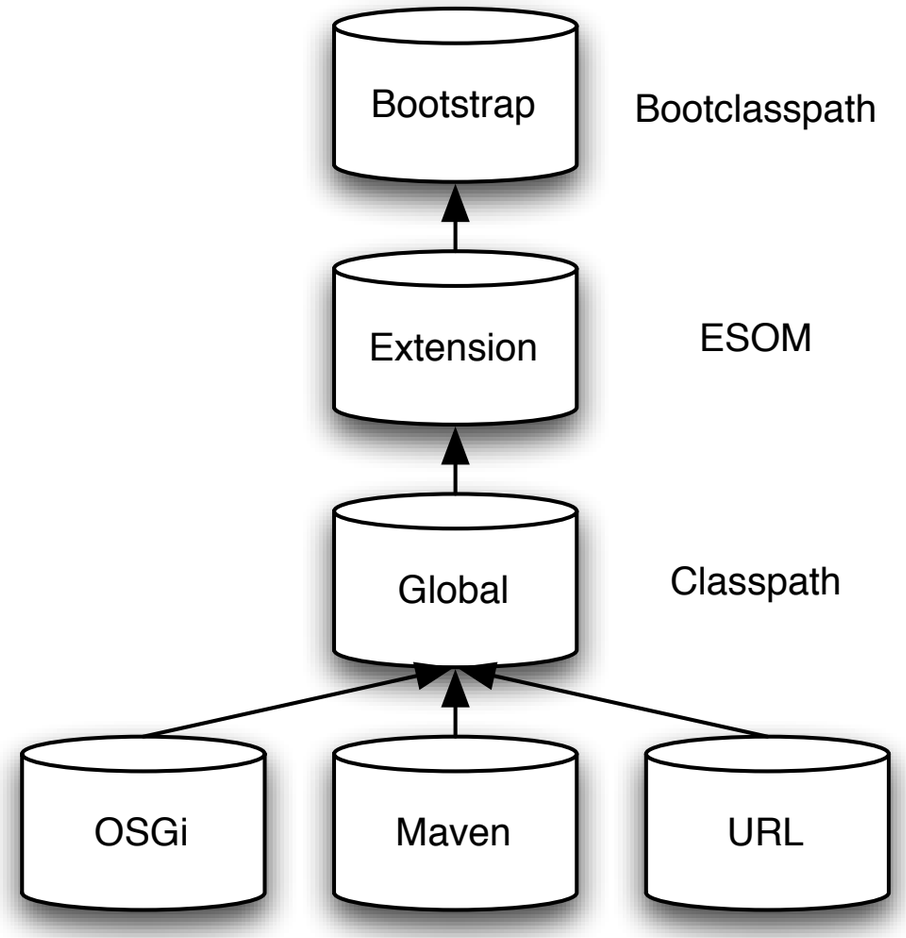
```
javac ...  // as usual

jam cvf org.foo-2.3.jam
  org/foo/*
  org/foo/api/*
  org/foo/impl/*
  org/foo/domain/*
  org/foo/util/*
```

JAM Contents

```
/META-INF/MANIFEST.MF              // Normal jar manifest
/MODULE-INF/MODULE.METADATA        // Module metadata
/MODULE-INF/bin/xyz-windows.dll    // Native libs
/MODULE-INF/bin/xyz-linux.so
/MODULE-INF/bin/yyz.jar            // Other jars
/org/foo/module-inf.class          // Compiled module info
/org/foo/**.class                  // Normal class files
/org/foo/icons                     // Resources
```

Module Resolution





Project Jigsaw

- Modularize the JDK itself
- Integrate at low level
- Integrate with native packaging
- NOT a JSR but part of JDK 7

NIO 2

- JSR 203: http://jcp.org/en/jsr/detail?id=203
- Project: http://openjdk.java.net/projects/nio

NIO 2 Themes

- New file system API
- Asynchronous I/O on sockets and files
- Completion of socket channel work

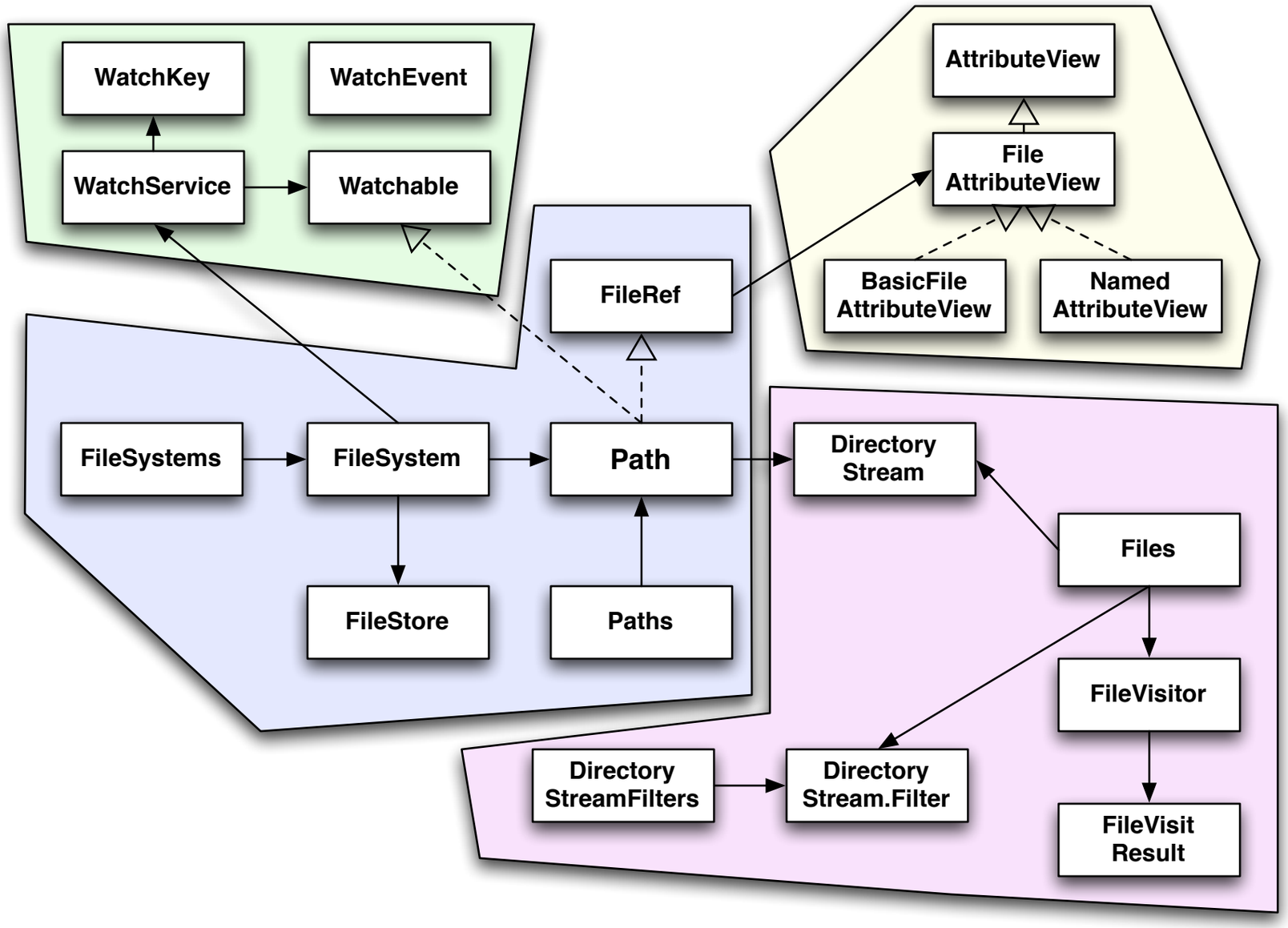
java.io.File problems

- Many methods return boolean
- No copy / move support
- No symbolic link support
- No change notification support
- Limited support for file attributes
- Not extensible

New file system API

- FileRef - represents file object in system
- Path - extends FileRef, binds a file to a system-dependent location
- FileSystem - interface to file system
- FileStore - underlying storage system

File System API



Using Path

```
import java.nio.file.*;

// FileSystems -> FileSystem -> Path
FileSystem fileSystem = FileSystems.getDefault();
Path homeDir = fileSystem.getPath("/Users/amiller");

// Shortcut with Paths helper class
Path homeDir = Paths.get("/Users/amiller");

// Resolve one path in terms of another
Path relativeTemp = Paths.get("temp");
Path absoluteTemp = relativeTemp.resolve(homeDir);

// Get relative path from a base
Path absoluteProfile = Paths.get("/Users/amiller/.profile");
Path relativeProfile = absoluteProfile.relativize(homeDir);
assert relativeProfile.isRelative();
assert relativeProfile.getNameCount() == 1;
```

Appending to a file

```
import java.io.*;
import java.nio.file.*;
import static java.nio.file.StandardOpenOption.*;

Path journal = Paths.get("/Users/amiller/journal.txt");

OutputStream stream =
    journal.newOutputStream(CREATE, APPEND);

try {
    writeEntry(stream);   // normal stuff
} finally {
    stream.close();
}
```

Copying and Moving

```
import java.nio.file.*;

Path home = Paths.get("/Users/amiller");
Path secrets = home.resolve("secrets.txt");

// Steal secrets
secrets.copyTo(home.resolve("stolenSecrets.txt"));

// Hide secrets
secrets.moveTo(Paths.get("/Users/dvader/secrets.txt"));
```

Walking Directories

```
Path music = Paths.get("/Users/amiller/files/music");

// External iterator
DirectoryStream<Path> mp3s =
    music.newDirectoryStream(“*.mp3”);

try {
   for(Path entry : mp3s)
      System.out.println(entry.getName());
} finally {
   mp3s.close();
}

// Internal iterator
Files.withDirectory(music, “*.mp3”, new FileAction<Path>() {
  public void invoke(Path entry) {
    System.out.println(entry.getName());
  }
});
```

Recursive Walk

```
Path itunes =
  Paths.get("/Users/amiller/Music/iTunes/iTunes Music");

public class Mp3Visitor extends SimpleFileVisitor<Path> {
  private Path root;
  public Mp3Visitor(Path root) {
    this.root = root;
  }

  public FileVisitResult visitFile(Path file,
    BasicFileAttributes attrs) {

    System.out.println(root.relativize(file));
  }
}

Files.walkFileTree(itunes, new Mp3Visitor(itunes));
```

File Attributes

```
Path file = Paths.get("/usr/bin/perl");

// true here means follow symbolic links
BasicFileAttributes attrs =
  Attributes.readPosixFileAttributes(file, true);
Set<PosixFilePermission> perms = attrs.permissions();

System.out.format("%s %s %s",
  PosixFilePermission.toString(perms),
  attrs.owner(),
  attrs.group());

// rwxr-xr-x root wheel
```

Watchers

```
import static java.nio.file.StandardWatchEventKind.*;

Path deploy = Paths.get("deploy");
WatchService watcher=FileSystems.getDefault().newWatchService();
WatchKey key = deploy.register(watcher,
  ENTRY_CREATE, ENTRY_DELETE, ENTRY_MODIFY);

for(;;) {
  key = watcher.take();   // blocks, also can poll
  for(WatchEvent<?> ev : key.pollEvents()) {
    switch(ev.kind()) {
      case ENTRY_CREATE:
        Path file = (Path)ev.getContext(); //relative to deploy
        //  deploy new stuff
      case ENTRY_MODIFY: ...
      case ENTRY_DELETE: ...
    }
  }
  key.reset();  // reset after processing
}
```

What else?

- NetworkChannel
  - finish work on channel to network socket
- Multicasting
  - DatagramChannel (now supports),
  - AsynchronousDatagramChannel (new)
- Asynchronous I/O for files and sockets
  - Future or callback style
  - Group support - manage thread pools

Date & Time

- JSR 310: http://jcp.org/en/jsr/detail?id=310
- Project: https://jsr-310.dev.java.net
- Wiki: http://wiki.java.net/bin/view/Projects/DateTimeAPI

A simple example

```
Date xmasEve = new Date(2008, 12, 25, 23, 59, 59);
```

Is this right?

A simple example

```
Date xmasEve = new Date(2008, 12, 25, 23, 59, 59);
```

Is this right?

NO!

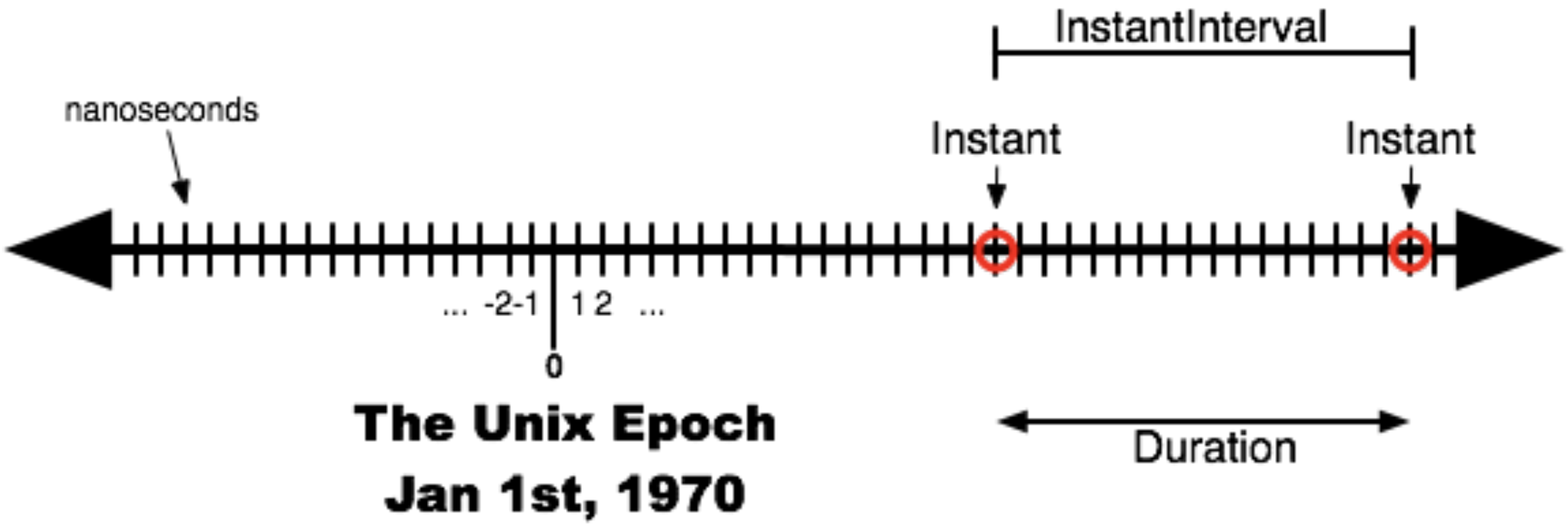
Correction:

```
int year = 2008 - 1900;
int month = 12 - 1;
Date xmasEve = new Date(year, month, 25, 23, 59, 59);
```

Other Problems

- Date, Calendar, SimpleDateFormatter are mutable
- TimeZone.getInstance() exception handling
- GregorianCalendar time + zone constructor
- Calendars can’t be formatted
- java.sql.Date, Time, Timestamp subclass poorly
- Formatters can’t format Timestamp nanos
- Date does not represent a date

Continuous


InstantInterval
nanoseconds
Instant
Instant
... -2-1 1 2 ...
0
The Unix Epoch
Jan 1st, 1970
Duration

Instants, Intervals, and Durations

```
// Create some instants in seconds
Instant start2008 = Instant.instant(1199167200);
Instant start2009 = Instant.instant(1230789600);
assert start2008.isBefore(start2009);

// Create an interval - [inclusive, exclusive] by default
InstantInterval year2008 =
  InstantInterval.intervalBetween(start2008, start2009);
assert year2008.contains(start2008);
assert ! year2008.contains(start2009);

// Create a duration in seconds
Duration minute = Duration.duration(60);
Duration hour = minute.multipliedBy(60);
Duration duration2008 = Duration.durationBetween(
  start2008, start2009);
```

Human


MAY FLOWERZ, LET ME SHOW U THEM
May 2008
Monday
Tuesday
Wednesday
Thursday
Friday
Caturday
Sunday
1
2
3
4
5
6
7
8
9
10
11
Mommy Cat Day
12
13
14
15
16
17
18
19
20
21
22
23
24
25

+

Local*, Offset*, Zoned*

```
// Local human-scale (not tied to TimeZone or Instant)
LocalDate myBirthday =
  LocalDate.date(1974, MonthOfYear.May, 1);
LocalTime quittingTime = LocalTime.time(17, 0);
LocalDateTime start2008 = LocalDateTime.dateMidnight(
  2008, MonthOfYear.JANUARY, 1);

// Tie to time zone offset of -6 hours from UTC
OffsetDateTime start2008Offset = OffsetDateTime.dateTime(
  start2008, ZoneOffset.zoneOffset(-6));

// Tie to current local time zone
ZonedDateTime start2008Zoned = ZonedDateTime.dateTime(
  start2008, Clock.system().timeZone());
```

Clock

```
// Use pre-defined system clock usually
Clock clock = Clock.system();

// Create instant
Instant now = clock.instant();

// Create human date / time
LocalDate today = clock.today();
```

Controlling time

```
LocalDateTime y2k = LocalDateTime.dateTime(
  1999, MonthOfYear.DECEMBER, 31,
  23, 59, 59, 999999999);

ZonedDateTime y2kHere = ZonedDateTime.dateTime(
  y2k, Clock.system().timeZone() );

Instant y2kInstant = y2kHere.toInstant();

// Mock around the Clock to test y2k transition
Clock clock = new my.test.ControlTimeClock(y2kInstant);
```

Other cool stuff

- Matchers, Adjusters, Resolvers
- Full time zone rule support
- Periods: “8 years, 2 months”
- Formatting and parsing

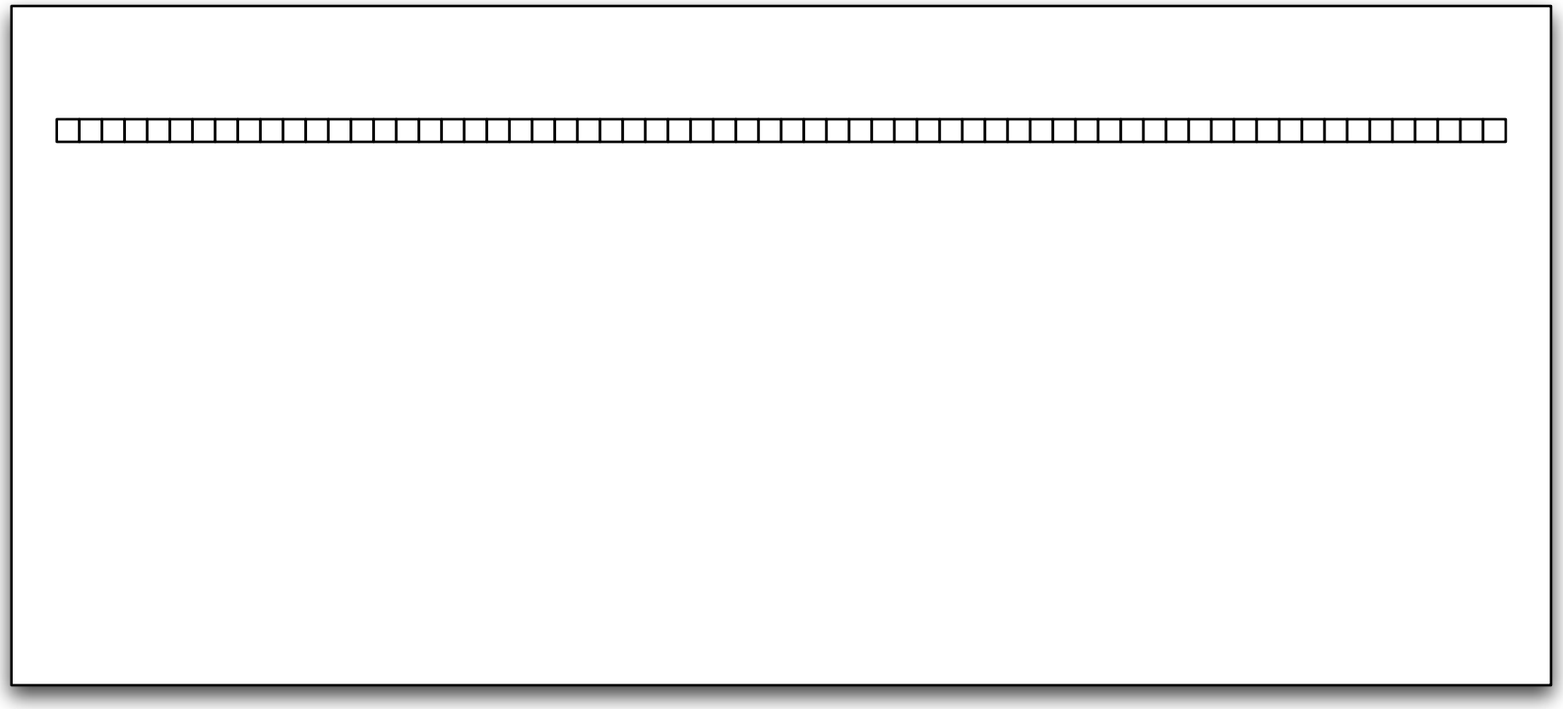
Integration

- Date, Calendar, etc - retrofit with interfaces
- JDBC - map to SQL types
- XML Schema - based on same standard so should be straightforward

Fork / join

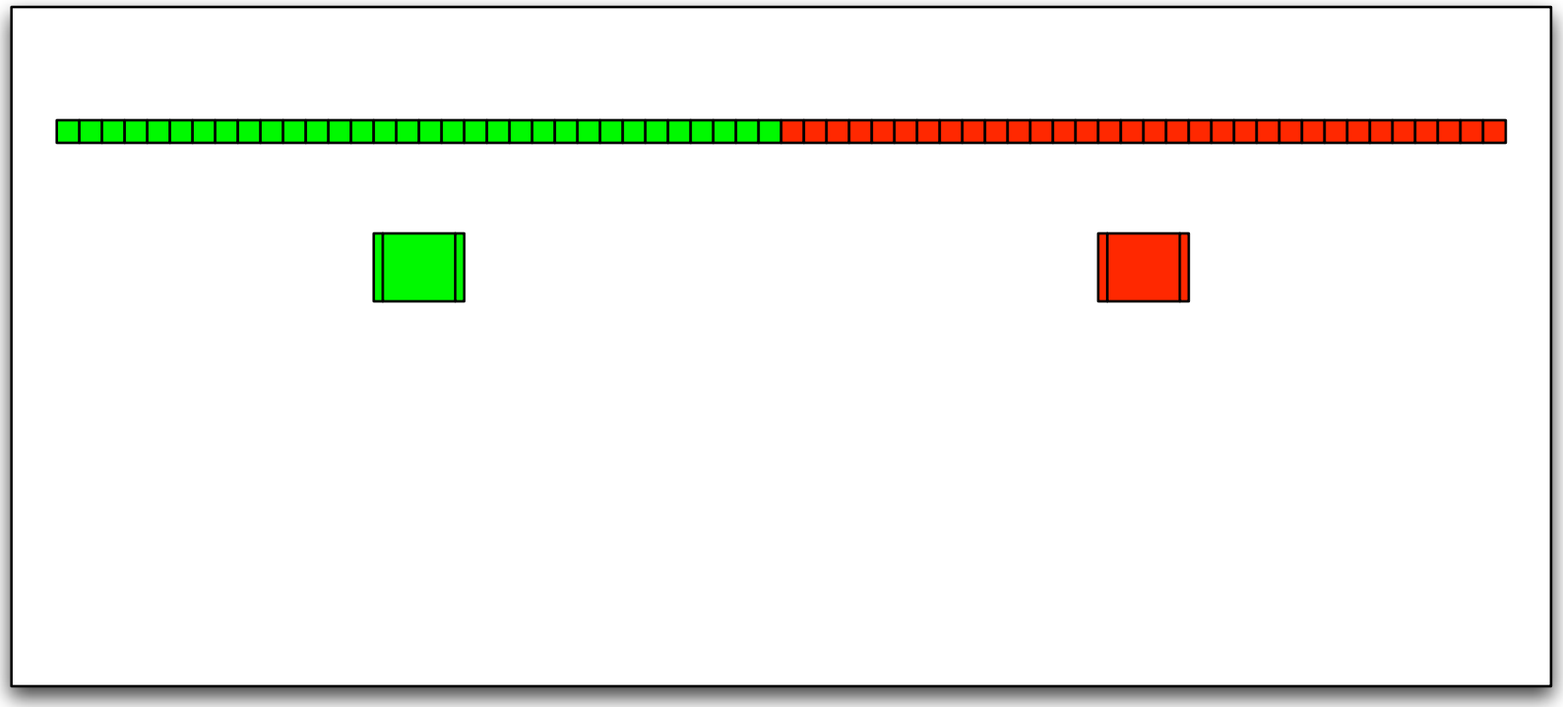
- JSR 166y: http://jcp.org/en/jsr/detail?id=166
- Project: http://gee.cs.oswego.edu/dl/concurrency-interest/index.html

Divide and Conquer

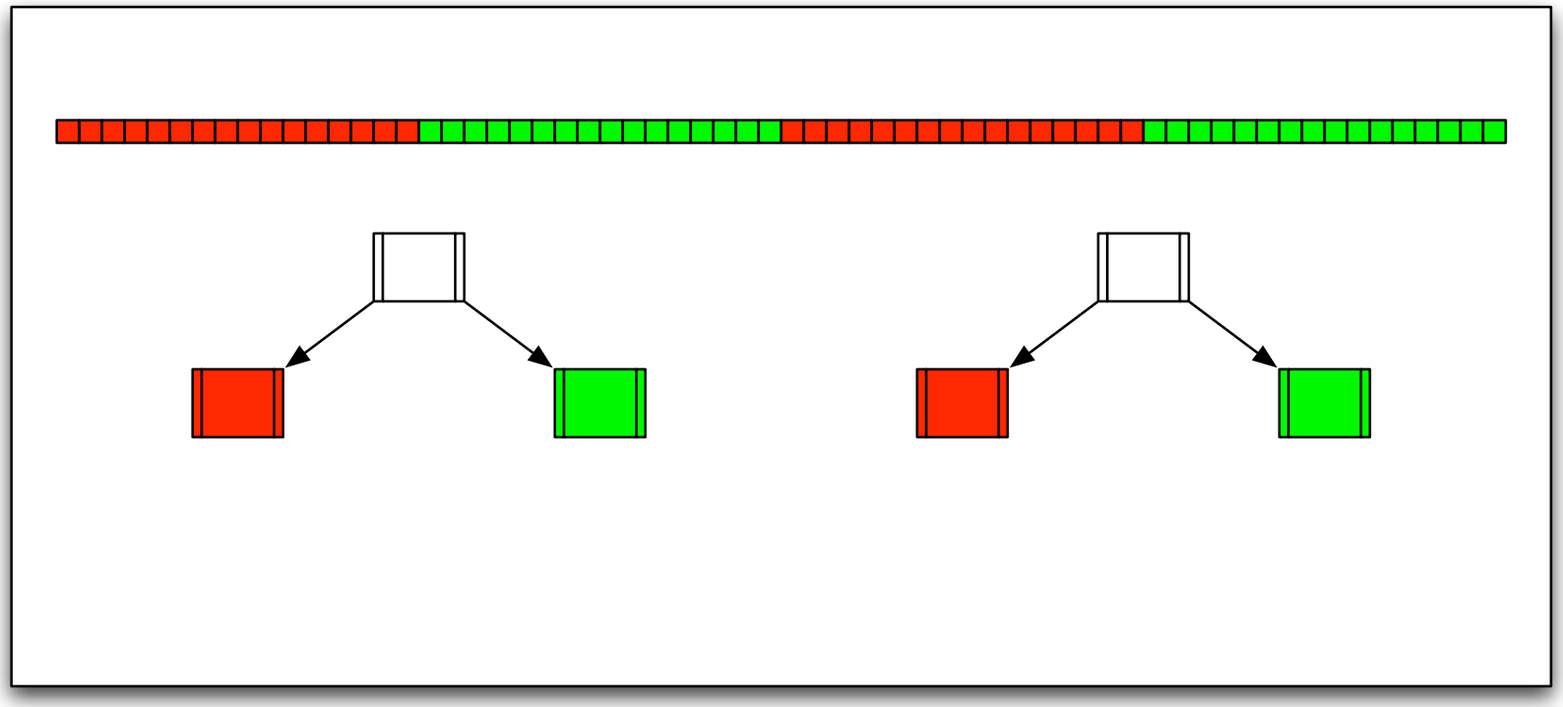


Task: Find max value in an array

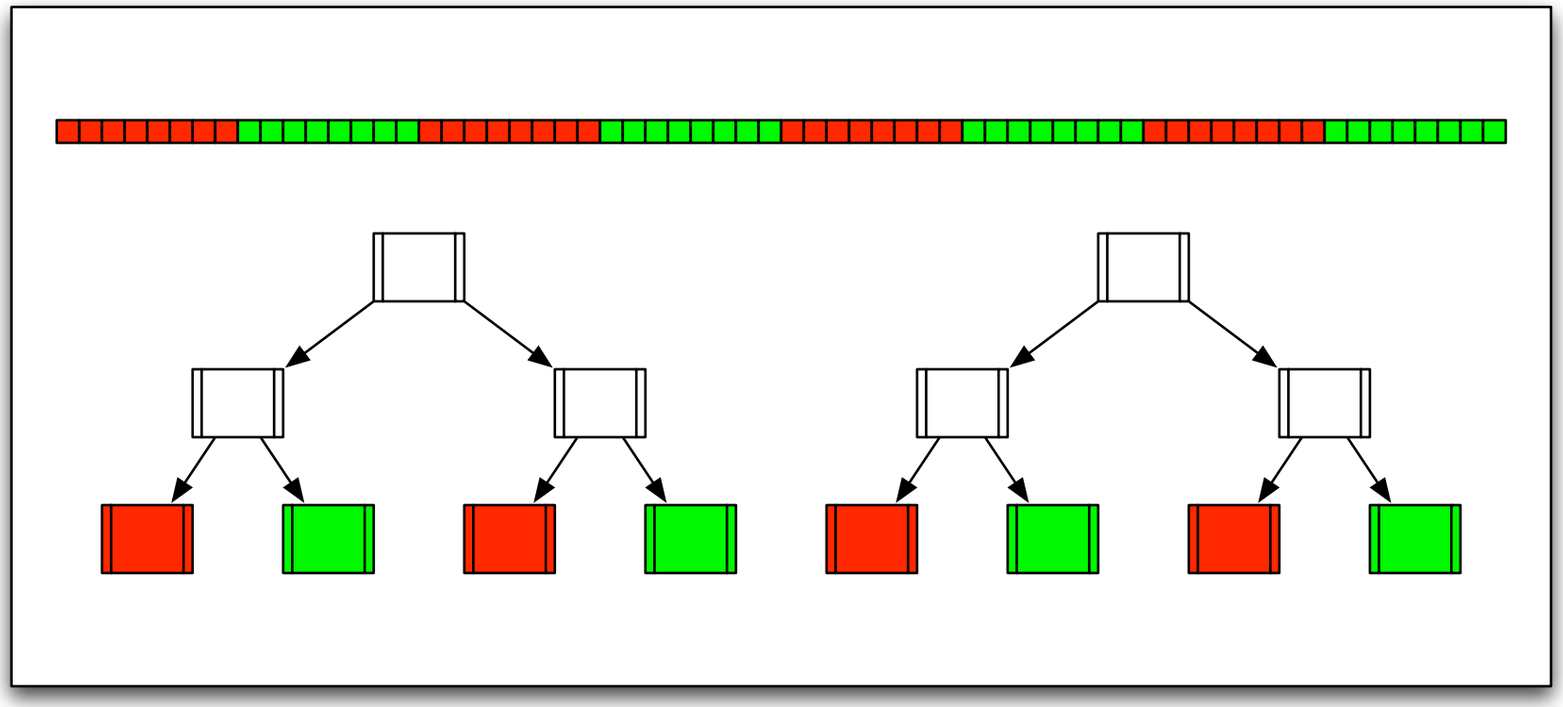
Divide and Conquer



Divide and Conquer



Divide and Conquer



ParallelArray

```
import static Ops.*;

ForkJoinPool fj = new ForkJoinPool(10);
Order[] data = ...
ParallelArray<Order> orders = new ParallelArray(fj, data);
```

ParallelArray

```
import static Ops.*;

// Filter
Ops.Predicate<Order> isLate = new Ops.Predicate<Order>() {
  public boolean op(Order o) {
    return o.due() < new Date();
  }
};

// Map
Ops.Predicate<Order> daysOverdue =
  new Ops.ObjectToInt<Order>() {
    public int op(Order o) {
      return daysOverdue(o.due());
    }
};
```

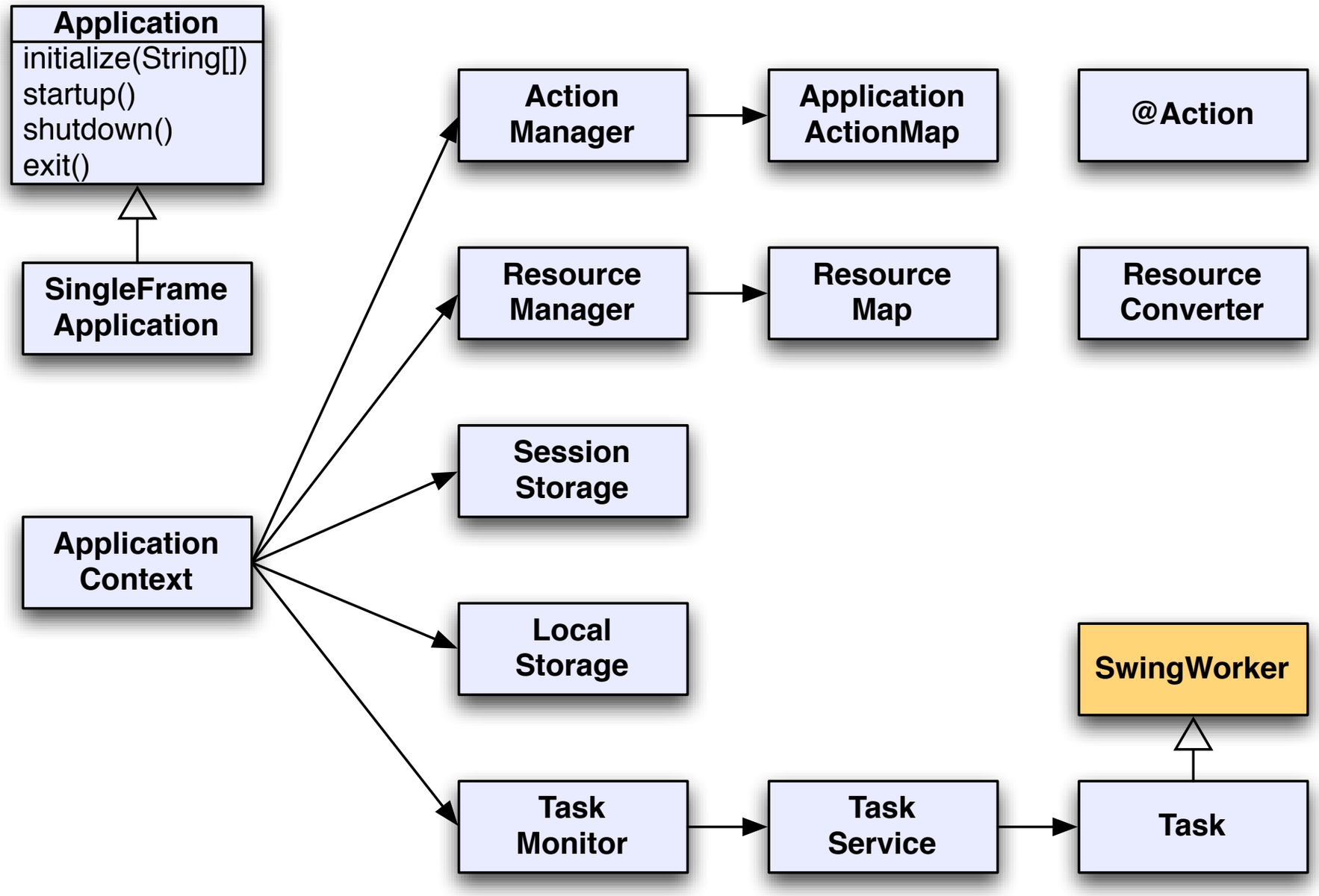
ParallelArray

```
SummaryStatistics<Integer> summary =
  orders.withFilter(isLate)
        .withMapping(daysOverdue)
        .summary();

System.out.println("overdue: " + summary.size());
System.out.println("avg by: " + summary.average());
```

Swing App Framework

- JSR 296: http://jcp.org/en/jsr/detail?id=296
- Project: https://appframework.dev.java.net/

Swing Application Framework



Beans Validation

- JSR 303: http://jcp.org/en/jsr/detail?id=303

Constraints and Validators

- JavaBeans property validation
- Meta-annotation for defining constraint annotations
- Core Constraint classes

Constraints

```
@ZipCodeCityCoherenceChecker    // works on Address itself
public class Address {
  @NotNull @Length(max=30)
  private String addressline1;

  @Length(max=30)
  private String addressline2;
  private String zipCode;
  private String city;

  @NotNull @Valid                    // check object graph
  private Country country;

  @Length(max=30) @NotNull
  public String getCity() { ... }

  // normal getters and setters
}
```

Validation

```
Validator<Address> addressValidator = ...
Address address = ...

// Validate all properties
Set<InvalidConstraint<Address>> invalidItems =
    validator.validate(address);

// Validate specific property
Set<InvalidConstraint<Address>> invalidItems =
    validator.validateProperty(address, “city”);

// Validate potential value for a property
Set<InvalidContraint<Address>> invalidItems =
    validator.validateValue(“city”, “St. Louis”);
```

JMX 2.0

- JSR 255: http://jcp.org/en/jsr/detail?id=255
- JSR 262: http://jcp.org/en/jsr/detail?id=262
- Project: https://ws-jmx-connector.dev.java.net

Features

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

Language Changes

- Project Coin

Strings in switch

```
static boolean isStooge(String stooge) {
  switch(input) {
    case "Moe":
    case "Curly":
    case "Larry":
    case "Shemp":
      return true;
    default:
      return false;
  }
}
```

Enum Comparisons

```
enum Rank {
  LIEUTENANT, CAPTAIN, MAJOR, COLONEL, GENERAL
}
```

Enum Comparisons

```
enum Rank {
  LIEUTENANT, CAPTAIN, MAJOR, COLONEL, GENERAL
}
```

```
// Compare using either compareTo() or ordinals
if(rank1.compareTo(rank2) < 0) ...
if(rank1.ordinal() < rank2.ordinal()) ...
```

Enum Comparisons

```
enum Rank {
  LIEUTENANT, CAPTAIN, MAJOR, COLONEL, GENERAL
}
```

```
// Compare using either compareTo() or ordinals
if(rank1.compareTo(rank2) < 0) ...
if(rank1.ordinal() < rank2.ordinal()) ...
```

```
// With enum comparison support can instead do:
if(rank1 < rank2) ...
```

Chained Invocation

```
// Construction with setters
DrinkBuilder margarita = new DrinkBuilder();
margarita.add("tequila");
margarita.add("orange liqueur");
margarita.add("lime juice");
margarita.withRocks();
margarita.withSalt();
Drink drink = margarita.drink();
```

Chained Invocation

```
// Construction with setters
DrinkBuilder margarita = new DrinkBuilder();
margarita.add("tequila");
margarita.add("orange liqueur");
margarita.add("lime juice");
margarita.withRocks();
margarita.withSalt();
Drink drink = margarita.drink();

// Construction with chained invocation
Drink margarita = new DrinkBuilder()
  .add("tequila")
  .add("orange liqueur")
  .add("lime juice")
  .withRocks()
  .withSalt()
  .drink();
```

Extension Methods

```
// I wish List had a sort() method...
List list = new ArrayList();
...
Collections.sort(list);    // works for now
list.sort();               // ERROR but cleaner
```

Extension Methods

```
// I wish List had a sort() method...
List list = new ArrayList();
...
Collections.sort(list);    // works for now
list.sort();               // ERROR but cleaner
```

```
// Maybe we can add it later!
import static java.util.Collections.sort;

List list = new ArrayList();
...
list.sort();     // now this works!
// equivalent to Collections.sort(list);
```

Exception boilerplate

```
// too much boilerplate!!!
} catch(RedException e) {
  LOGGER.info(e.getMessage(), e);
  throw e;
} catch(BlueException e) {
  LOGGER.info(e.getMessage(), e);
  throw e;
}
```

Exception boilerplate

```
// too much boilerplate!!!
} catch(RedException e) {
  LOGGER.info(e.getMessage(), e);
  throw e;
} catch(BlueException e) {
  LOGGER.info(e.getMessage(), e);
  throw e;
}


// Catch common superclass???
} catch(Exception e) {
  LOGGER.info(e.getMessage(), e);
  throw e;
}
```

Multi-catch

```
public void foo()
throws RedException, BlueException {

   try {
      ...

   // Use , to catch multiple types
   } catch(RedException, BlueException e) {
      LOGGER.info(e.getMessage(), e);
      throw e;
   }
}
```

Safe Rethrown

```
public void foo()
throws RedException, BlueException {

   try {
      ...

   } catch(final Throwable e) {
      LOGGER.info(e.getMessage(), e);
      throw e;
   }
}
```

Null Handling

```
Car car = ...
Integer tilt = null;
if(car != null) {
   Sunroof sunroof = car.getSunroof()(;
   if(sunroof != null) {
      tilt = sunroof.getTilt();
   }
}
```

```
Integer tilt = car?.getSunroof()?.getTilt();
```

Type Inference

```
Map<String, Integer> map =
   new HashMap<String, Integer>();
```

```
Map<String, Integer> map =
   new HashMap<>();
```

JVM

- Dynamic language support (JSR 292)
  - Method handles
  - Interface injection?
  - Tail recursion?
- G1 garbage collector

Favorites?

What was your favorite library change?

What was your favorite language change?

What's missing?

Learn more...

- http://tech.puredanger.com/java7
- http://java7.tumblr.com

Find Me...

- http://tech.puredanger.com
- Twitter: @puredanger