/**
* **JCache** is here. Say Goodbye to proprietary
* Caching APIs!
*
*
* @author Jaromir Hamala
* @since 18-03-2014
*/

Who Am I?

- Jaromir Hamala
- Passionate Developer
- Working for Hazelcast
- Open Source Fan

Warning: May Get Extremely Silly!

Why?

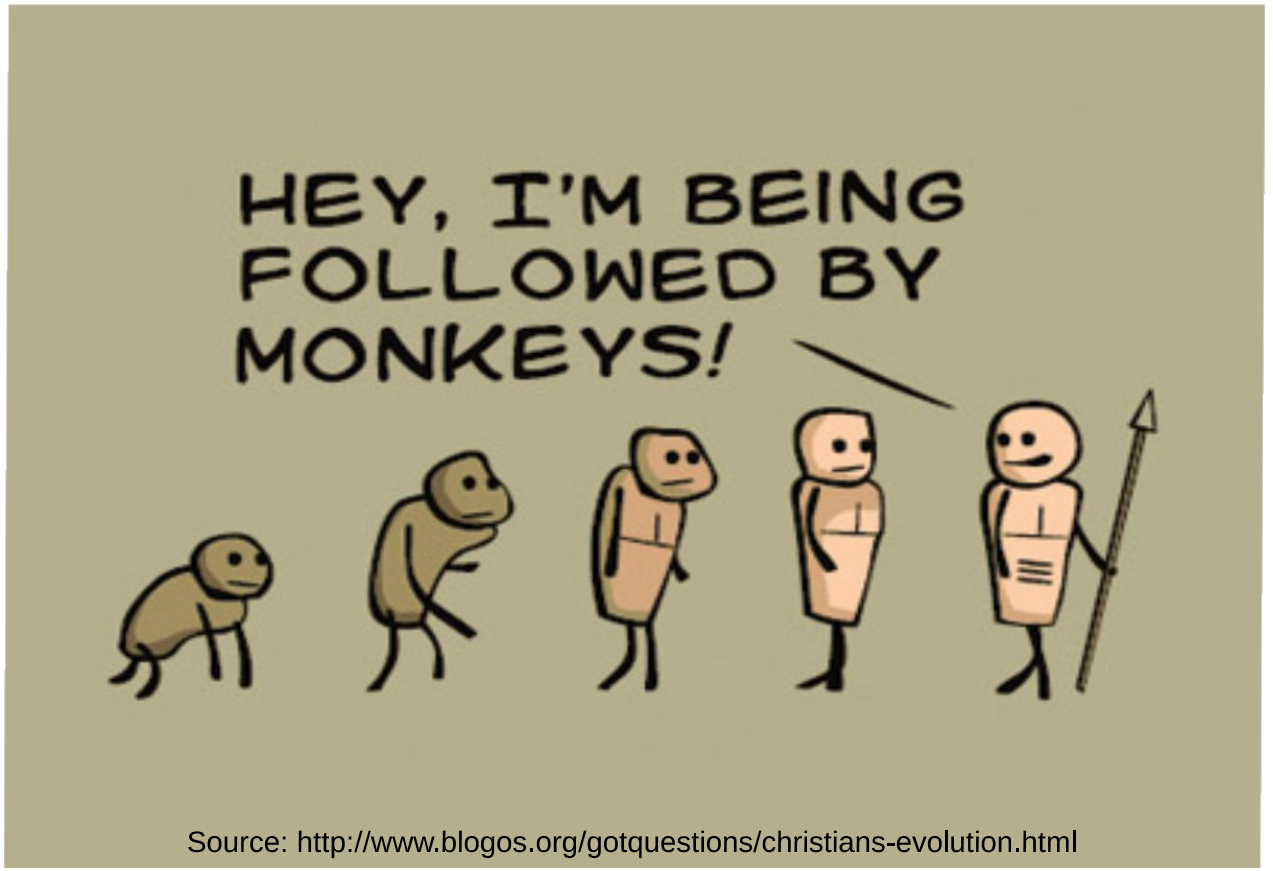
L1 cache reference	0.5	ns	
Branch mispredict	5	ns	
L2 cache reference	7	ns	
Mutex lock/unlock	25	ns	
Main memory reference	100	ns	
Compress 1K bytes with Zippy	3,000	ns	
Send 1K bytes over 1 Gbps network	10,000	ns	0.01 ms
Read 4K randomly from SSD*	150,000	ns	0.15 ms
Read 1 MB sequentially from memory	250,000	ns	0.25 ms
Round trip within same datacenter	500,000	ns	0.5 ms
Read 1 MB sequentially from SSD*	1,000,000	ns	1 ms
Disk seek	10,000,000	ns	10 ms
Read 1 MB sequentially from disk	20,000,000	ns	20 ms
Send packet CA->Netherlands->CA	150,000,000	ns	150 ms

Sources: https://gist.github.com/jboner/2841832 http://commons.wikimedia.org/wiki/File:Roaring_Lion_Travis_Jervey.jpg

Evolutions of Caching

- Roll your own :-(
- Use a proprietary API :-/
- Use JCache! :-)

Do. Not. Do. That.



Source: http://www.blogos.org/gotquestions/christians-evolution.html

What is JCache

- Standard API for Caching
- Think of “JDBC for caching” (only better:)
- JSR-107
  - The longest running JSR ever (?)
- *“The best thing since sliced bread”*
- Targetted to Java SE 6+

Stage	Start
Final Release	18 Mar, 2014
Final Approval Ballot	04 Mar, 2014
Proposed Final Draft	24 Oct, 2013
Public Review Ballot	27 Aug, 2013
Public Review	05 Jul, 2013
Early Draft Review	23 Oct, 2012
Expert Group Formation	20 Mar, 2001
JSR Review Ballot	06 Mar, 2001

What *really* is JCache

- *Basic* Caching - *somewhat similar to a j.u.Map*
- Events
- Computations

Be Aware: It is *just* an API!

- Think of JMS (only better:)
- It doesn't say anything about data distribution, network topology or wire-level protocol!

Existing Implementations

- Coherence
- EHCache
- Hazelcast
- Infinispan

Source: http://www.mtbeer.com/how-beer-tap-handles-are-made/

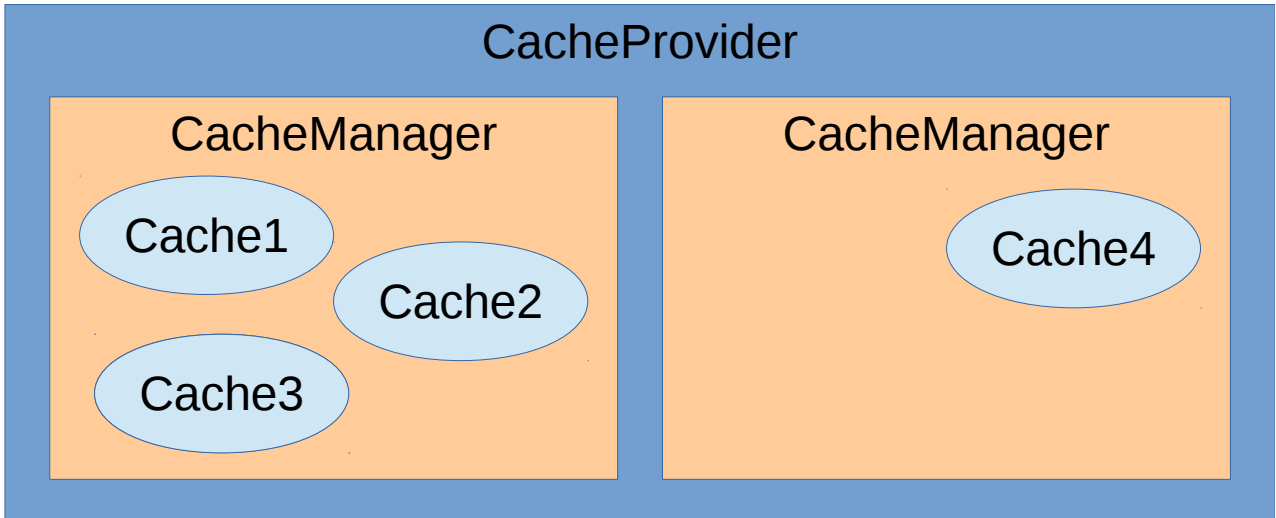
Quick Start

1. Add JCache API JAR on a classpath
2. Choose your implementation
3. Code!

Source: http://commons.wikimedia.org/wiki/File:Start_Jeremy_Wariner_2007.jpg

Basic Terminology

- Cache
- CacheManager – ~~*“provides a means of establishing, configuring acquiring, closing and destroying uniquely named Caches.”*~~
  - Just think of CacheFactory (for now)
- CacheProvider – manages lifecycle of CacheManagers

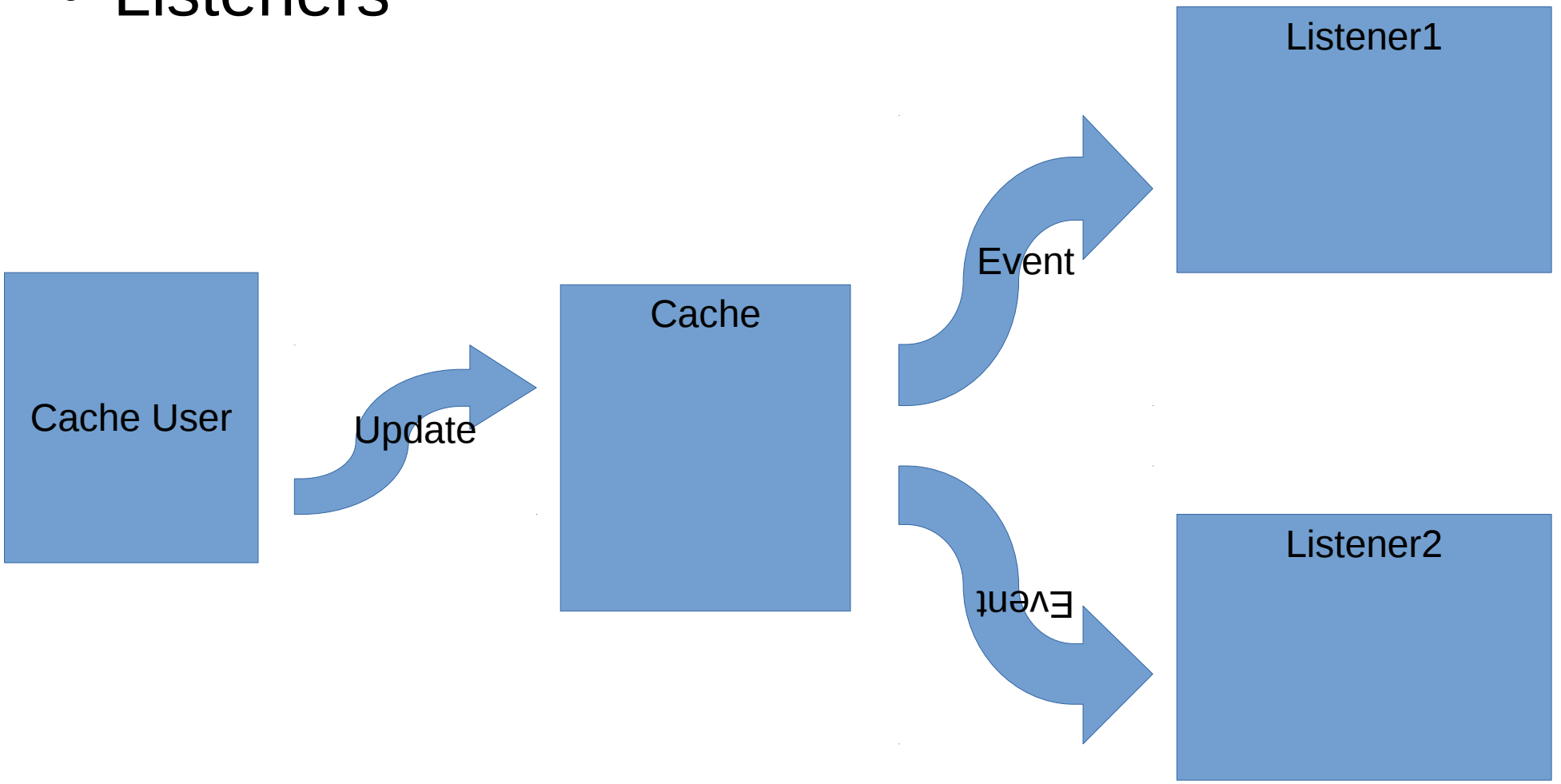


Interesting Features I

- Entry Processors
  - Data Transformations
  - Computing

Interesting Features II

- Listeners



Listeners

- `CacheEntryCreatedListener`
  - `onCreated();`
- `CacheEntryRemovedListener`
  - `onRemoved();`
- `CacheEntryUpdatedListener`
  - `onUpdated();`
- `CacheEntryExpiredListener`
  - `onExpired();`

Cache Loaders / Writers

Source: http://commons.wikimedia.org/wiki/File:Stipula_fountain_pen.jpg

CacheLoader

```
public interface CacheLoader<K, V> {

  V load(K key) throws CacheLoaderException;


  Map<K, V> loadAll(Iterable<? extends K>
keys) throws CacheLoaderException;

}
```

CacheWriter

```
public interface CacheWriter<K, V> {

  void write(Cache.Entry<? extends K, ? extends V>
entry) throws CacheWriterException;

  void writeAll(Collection<Cache.Entry<? extends K,
? extends V>> entries) throws CacheWriterException;


  void delete(Object key) throws
CacheWriterException;

  void deleteAll(Collection<?> keys) throws
CacheWriterException;

}
```

Other Features

- Annotations
  - `@CacheResult`
  - `@CachePut`
  - `@CacheRemove`
  - `@CacheRemoveAll`
- Expiry Policy
- Statistics
- Store-By-Value vs. Store-By-Reference

Integrations

- Spring Framework
- Java EE 8 (*maybe*)
- Payara

Source: http://towson.shownbyphotos.com/20060818-towson-univ-0003-800.jpg-large.html

Tips & Tricks

- Isolate Implementation-Specific Stuff
- Be carefull with the default CachingProvider
- EntryProcessors are awesome!

http://en.wikipedia.org/wiki/File:1_bowl_skate_boarding.JPG

http://www.mylifeonboard.net/2011/03/04/is-it-dangerous-yeah-danger-is-the-middle-name-of-our-sport/

Missing Bits

- No Async Interface in the spec
  - Proprietary extensions
- Eviction
- Query

$25 REWARD!

I will pay the above reward for the recovery of the following described Horse that was Stolen from Snyder's pasture on White River, on or about June 19, 1891, the same belonging to me:

ONE BLACK HORSE

SIX YEARS OF AGE.
WEIGHT, 1,000 POUNDS.
SADDLE MARKS.
BRANDED, "71" RIGHT SHOULDER.
ROUND BUILD.

WIRE ME IF FOUND.

BOLTON ROGERS,
SEATTLE, WASH. Chief of Police.

Northwestern Print, Occidental Block, Seattle.

Further Sources

- https://jcp.org/en/jsr/detail?id=107 - 130 pages
- https://github.com/jsr107
- https://groups.google.com/forum/#!forum/jsr107

Questions?

```
String viking =
dictionary.get("Thank you");

assert("tack", viking);
```