Java Reference Classes

Kevin Jones

@kevinrjones

Introduction

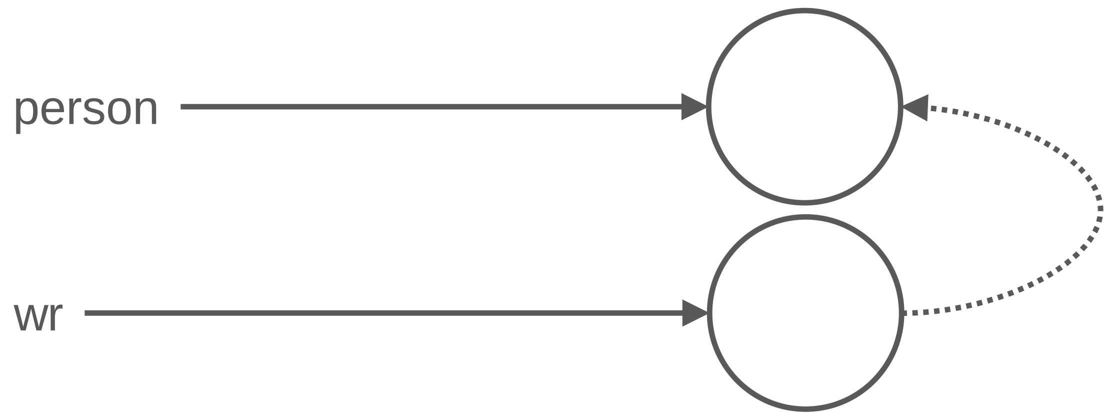
- Java has always had ‘strong’ references
  - Object not GC’d until references are released
- Other types of references are available
  - ‘Special’ class in java.lang.ref package
  - Soft, Weak and Phantom
- WeakHashMap, ReferenceQueue

Reference Rules

- Strong -> Soft -> Weak -> Phantom
- Object not GC if there is a strong reference
  - Can be GC'd if there is a Soft, Weak or Phantom reference
- Soft will be collected if there is memory pressure
- Weak will be collected immediately
- Phantom references different to the other two
  - Cannot retrieve the object through a phantom reference

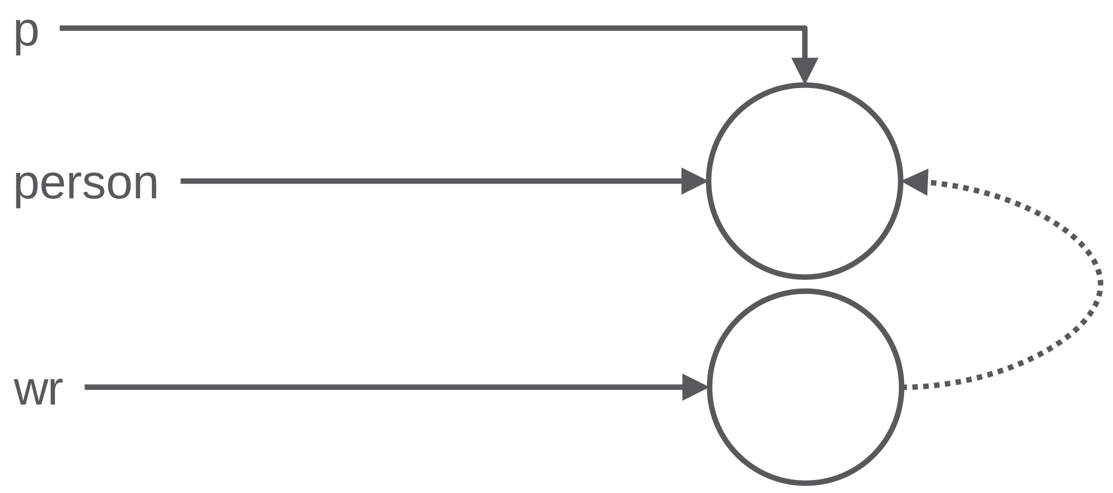
Using Reference Types

```
Person person = new Person();
WeakReference<Person> wr = new WeakReference<Person>(person);
```



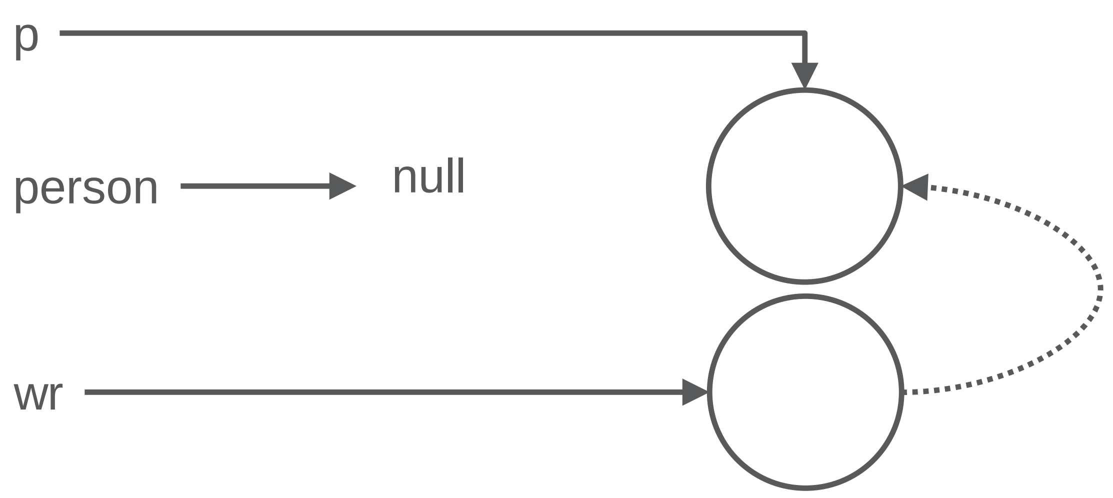
Using Reference Types

```
Person p = wr.get();
```



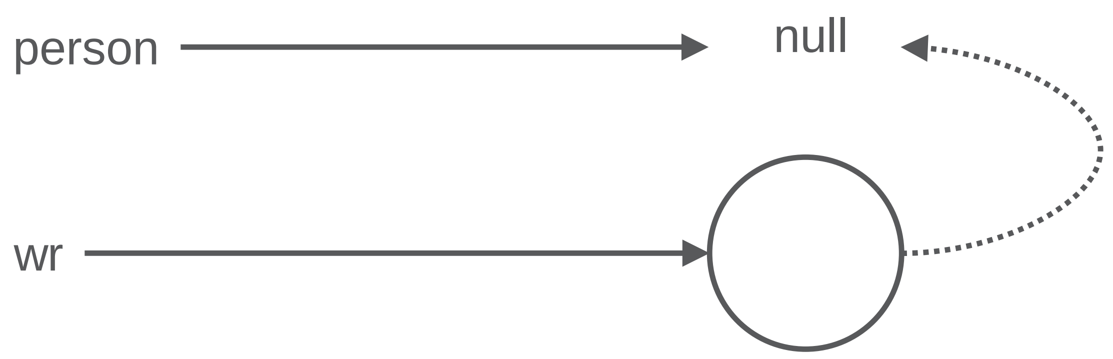
Using Reference Types

```
person = null;
Person p = wr.get();
```



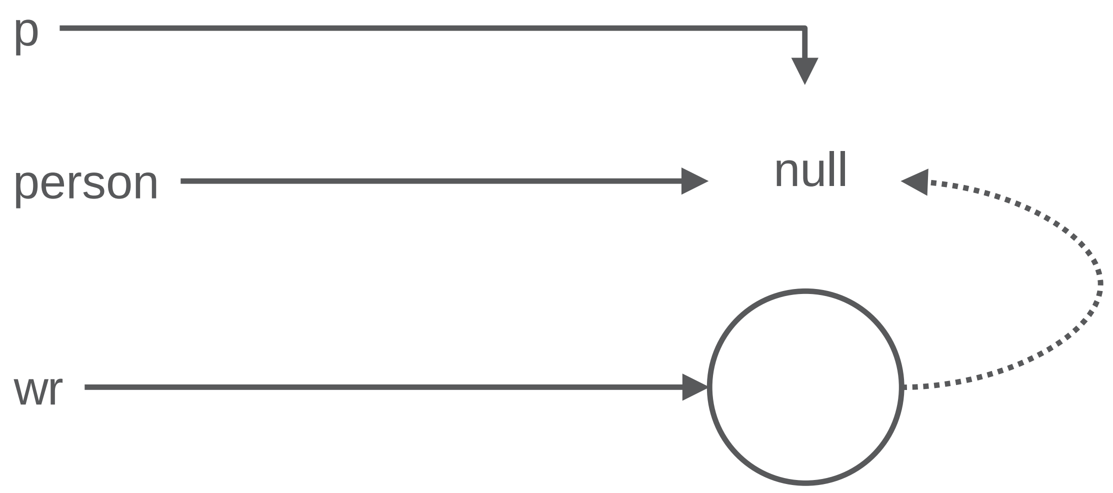
Using Reference Types

```
person = null;
System.gc();
```



Using Reference Types

```
Person p = wr.get();
```



Demo

- Reference types demo

Usages of Reference Types

- WeakReference
  - Associate meta data with another type
  - Use WeakHashMap
- SoftReference
  - Can be used for caching
- PhantomReference
  - Interaction with the garbage collector

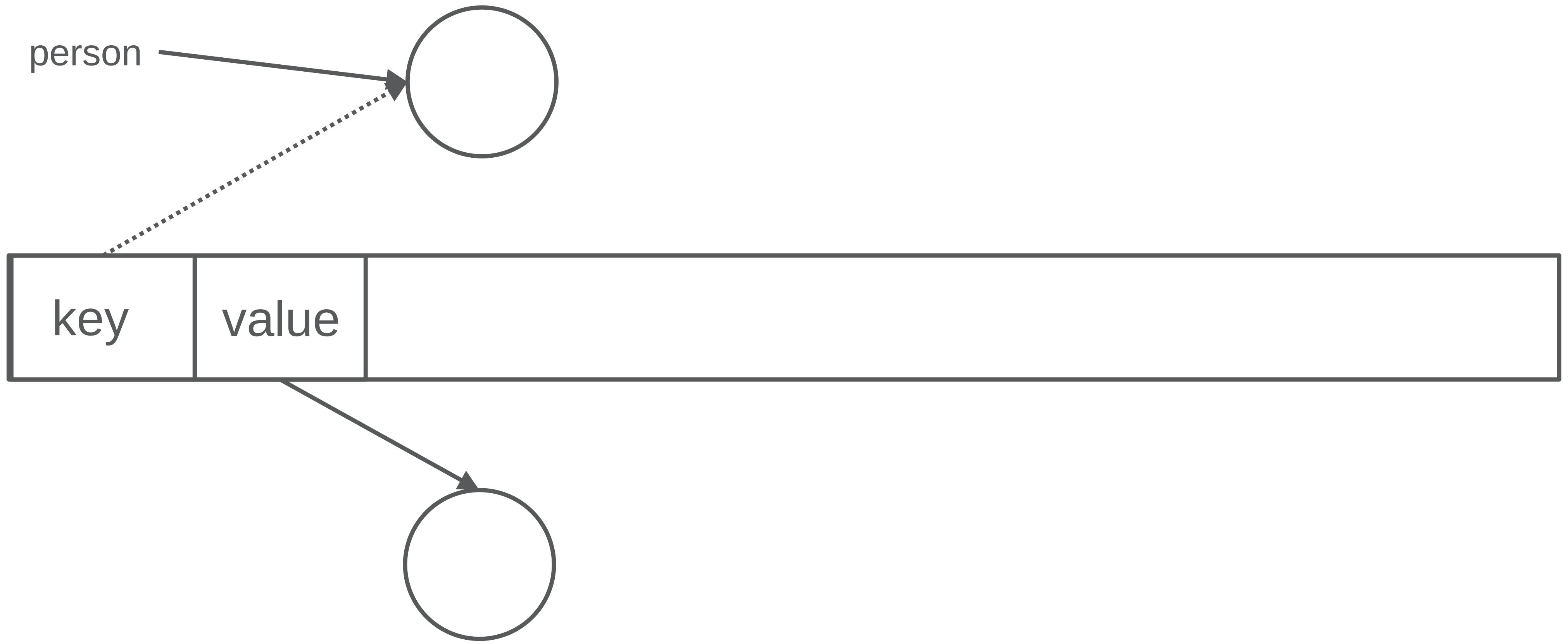
SoftReference Caching

- Hold a SoftReference to an object
  - As well as a strong reference
- When strong reference is cleared soft is still available
- Not always a great mechanism
  - No control over the cache
  - It’s all managed by the garbage collector

WeakHashMap

- Like a HashMap
- Key is a weak reference to an object
  - Store a weak reference to an object as a key
  - Value is the object's 'meta data'
- When object has no more strong references
  - The key is released
  - 'Meta data' goes away

WeakHashMap



WeakHashMap Demo

ReferenceQueue

- Pass a reference queue to constructor when creating the reference object
  - Optional except for PhantomReference
- References types enqueued to ReferenceQueue
- Useful when you want to associate some cleanup mechanism with an object

ReferenceQueue

```
Person person = new Person();
```

```
ReferenceQueue<Person>
   referenceQueue = new ReferenceQueue<Person>();
WeakReference<Person>
   wr = new WeakReference<Person>(person, referenceQueue);
```

Using the ReferenceQueue

- When all strong references cleared
  - Reference object is added to the reference queue
- ReferenceQueue has poll and remove methods
  - *poll* returns immediately
  - *remove* has a timeout
  - Both remove object from the queue

ReferenceQueue Example

- Can be used to attach clean up code
  - Extend reference type
- When all strong references cleared
  - Reference object added to the queue
- Dequeue object from the queue
  - Call its 'clean up' method

Demo

- ReferenceQueueDemo

PhantomReference

- Used instead of finalizers
- Finalizers have issues
  - Can be expensive
  - Not sure when they will be called

Demo