Java Reference Precise – 2022

**Soft references**

**SoftReferences will keep the objects alive until there is memory available on the heap, but it will discard them before an OutOfMemoryError**. **A Soft reference won't be collected until its memory is really needed**. i.e. garbage collects before OutOfMemoryError. Soft reference objects are cleared at the discretion of the garbage collector in response to memory demand. Soft references are most often used to implement memory-sensitive caches. SoftReferences can be used to implement a cache that can grow without risking an application crash. To do this, you need to implement a Map interface in which values are stored, wrapped inside a SoftReference.

SoftReference are like weakreference but they survive garbage collection cycle till memory is available in plenty.

In practice softly reachable objects are generally retained as long as memory is in plentiful supply. This makes them an excellent foundation for a cache, such as the image cache described above,

Counter prime = new Counter(); // prime holds a strong reference - line 2

SoftReference<Counter> soft = new SoftReference<Counter>(prime) ;

//soft reference variable has SoftReference to Counter Object created at line 2

prime = null; // 🡸 **now Counter object is eligible for garbage, collection but only be collected when JVM absolutely needs memory**

**Weak references**

**Weak references are collected eagerly.** WeakReferences can be used, for example, to store some information related to an object until the object gets finalized. To do this, you can implement a Map in which the keys are wrapped in a WeakReference. As soon as GC reclaims the key object, you can remove the value as well.

Counter counter = new Counter(); // strong reference - line 1

WeakReference<Counter> weakCounter = new WeakReference<Counter>(counter);

//weak reference

counter = null; // 🡸 **now Counter object is eligible for garbage collection**

Now as soon as you make strong reference counter = null, counter object created on line 1 becomes eligible for garbage collection.

On the other hand, had this been Soft Reference, Counter object is not garbage collected until JVM absolutely needs memory.

If GC finds that an object is weakly reachable (reachable only through weak references), it'll clear the weak references to that object immediately.

So, the decision depends on usage - if you're caching information that is expensive to construct, but nonetheless reconstructible from other data, use soft references - if you're keeping a reference to a canonical instance of some data, or you want to have a reference to an object without "owning" it (thus preventing it from being GC'd), use a weak reference.

* An object which has No Strong Reference and has only Soft or Weak Reference is Softly Reachable
* An object having only WeakReference and no Strong or soft references is Weekly Reachable

Note: **So in brief: Soft references try to keep the reference. Weak references don’t try to keep the reference. Phantom references don’t free the reference until cleared.**

**Stories**

To reuse (and stretch) our restaurant metaphor one last time: **A SoftReference is like a customer that says, "I’ll leave my table only when there are no other tables available.** **A WeakReference is like someone ready to leave as soon as a new customer arrives.** **A PhantomReference is like someone ready to leave as soon as a new customer arrives, but actually not leaving until the manager gives them permission.**

Analogy: Assume a JVM is a kingdom, Object is a king of the kingdom, and GC is an attacker of the kingdom who tries to kill the king(object).

• When King is Strong, GC can not kill him.

• When King is Soft, GC attacks him but King rule the kingdom with protection until resource are available.

• When King is Weak, GC attacks him but rule the kingdom without protection.

• When king is Phantom, GC already killed him but king is available via his soul.

**What is Softly Reachable Object**

Student strongRef = new Student();

SoftReference<Student> softReference = new SoftReference<>(strongRef);

Both the below cases The object in heap is softly reachable.

Student stRef = new Student();

SoftReference <Student> sfRef = new SoftReference<>(stRef);

stRef = null;

Or

SoftReference <Student> sfRef = new SoftReference<>(new Student());

**Reference<HeavyList> softRef = new SoftReference<>(curr, queue);**

**Reference<HeavyList> weakRef = new WeakReference<>(curr, queue);**

**Reference<HeavyList> phantomRef = new PhantomReference<>(curr, queue);**

PhantomReferences can be used to notify you when some object is out of scope to do some resource cleanup.

**What is ReferenceQueue**

A ReferenceQueue is a simple data structure onto which the garbage collector places reference objects when the reference field is cleared (set to null). You would use a reference queue to find out when an object becomes softly, weakly, or phantom reachable, so your program can take some action based on that knowledge.