Types of references

- Different references
- Strong
- Weak
- Soft
- Phantom

Different references

Java provides two different types/classes of Reference Objects: strong and weak.

Weak Reference Objects can be further divided into soft and phantom.

These references are different solely because of the existence of a garbage collection mechanism in the JVM.

The decision of reclaiming memory from the object heap depends not only on the fact that there are active references to an object, but also on the type of reference to the object.

So, type of reference determine how the objects are garbage collected.

Strong reference

A normal one.

These type of references we use daily while writing the code. Any object in the memory which has active strong reference is not eligible for garbage collection.

I an object can’t be reached by chain of string references from the stack it it eligible for garbage collection.

Weak reference

WeakReference<String> weakString = new WeakReference<>("Hello Java");

Weak Reference Objects are not the default type/class of Reference Object and to be used they should be explicitly specified.

When an object in memory is reachable only by Weak Reference Objects, it becomes automatically eligible for GC.

So, after GC weakString will be *null*

Soft reference

A soft Reference Object is basically a weak Reference Object that remains in memory a bit more: normally, it resists GC cycle until memory is available and there is no risk of OutOfMemoryError (in that case, it can be removed).

So, soft reference is a Weak reference, but if an object can be reached by soft references it is not eligible for collection till OutOfMemoryError.

Phantom reference

A phantom Reference Object is useful only to know exactly when an object has been effectively removed from memory.

The objects which are being referenced by phantom references are eligible for garbage collection.

But, before removing them from the memory, JVM puts them in a queue called ‘reference queue’ .

They are put in a reference queue after calling finalize() method on them.

You can’t retrieve back the objects which are being phantom referenced. That means calling get() method on phantom reference always returns null.