Types of References in Java

Difficulty Level : Medium • Last Updated : 06 Dec, 2019

In Java there are four types of references differentiated on the way by which they are garbage collected.

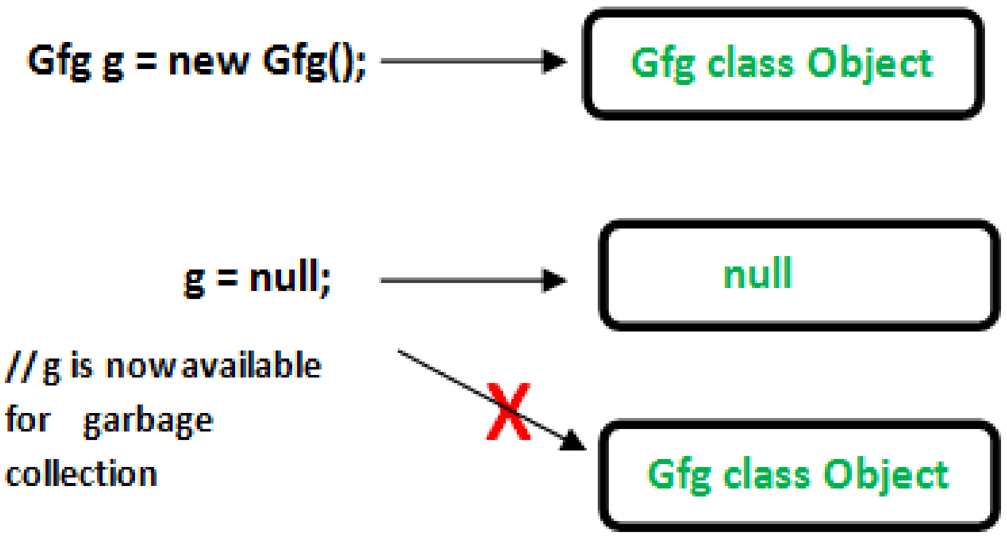
1. Strong References
2. Weak References
3. Soft References
4. Phantom References

Prerequisite: Garbage Collection

- **Strong References:** This is the default type/class of Reference Object. Any object which has an active strong reference are not eligible for garbage collection. The object is garbage collected only when the variable which was strongly referenced points to null.

```
MyClass obj = new MyClass ();
```

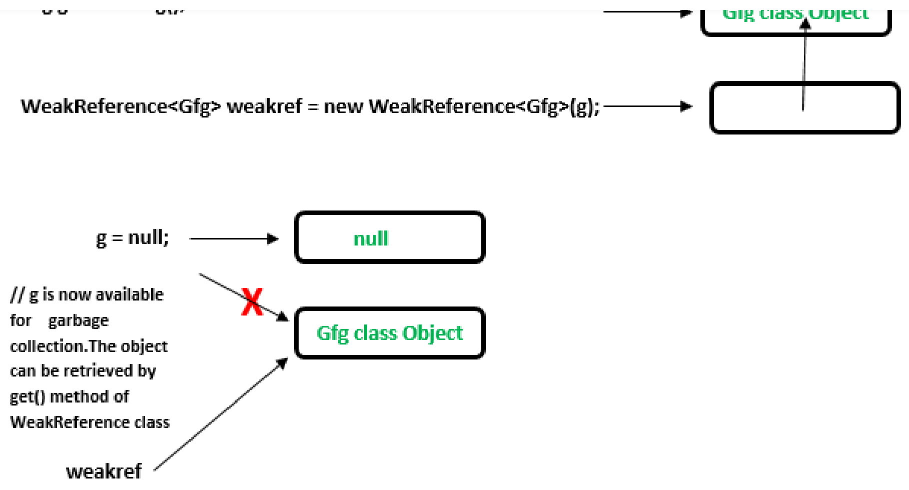
```
obj = null;
//'obj' object is no longer referencing to the instance.
So the 'MyClass type object is now available for garbage collection.
```



```
// Java program to illustrate Strong reference
class Gfg
```

```
public class Example
{
    public static void main(String[] args)
    {
         //Strong Reference - by default
        Gfg g = new Gfg();

        //Now, object to which 'g' was pointing earlier is
        //eligible for garbage collection.
        g = null;
    }
}
```

- **Weak References:** Weak Reference Objects are not the default type/class of Reference Object and they should be explicitly specified while using them.
  - This type of reference is used in WeakHashMap to reference the entry objects .
  - If JVM detects an object with only weak references (i.e. no strong or soft references linked to any object object), this object will be marked for garbage collection.
  - To create such references java.lang.ref.WeakReference class is used.
  - These references are used in real time applications while establishing a DBConnection which might be cleaned up by Garbage Collector when the application using the database gets closed.


Gfg class Object
WeakReference<Gfg> weakref = new WeakReference<Gfg>(g);
g = null;
null
// g is now available for garbage collection.The object can be retrieved by get() method of WeakReference class
X
Gfg class Object
weakref

```
//Java Code to illustrate Weak reference
import java.lang.ref.WeakReference;
class Gfg
{
    //code
    public void x()
    {
        System.out.println("GeeksforGeeks");
    }
}

public class Example
{
    public static void main(String[] args)
    {
        // Strong Reference
        Gfg g = new Gfg();
        g.x();
```

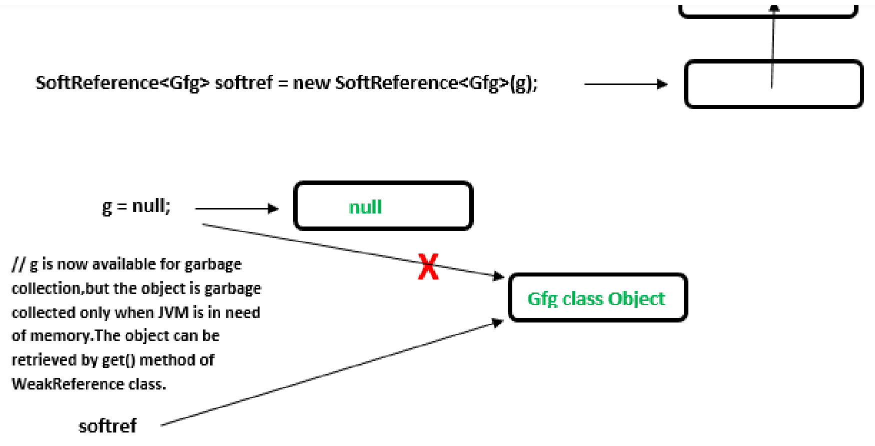
```
            //Now, Gfg-type object to which 'g' was pointing earlier
            //is available for garbage collection.
            //But, it will be garbage collected only when JVM needs memory.
            g = null;

            // You can retrieve back the object which
            // has been weakly referenced.
            // It successfully calls the method.
            g = weakref.get();

            g.x();
        }
    }
```

Output:

```
GeeksforGeeks
GeeksforGeeks
```

Two different levels of weakness can be enlisted: Soft and Phantom

- **Soft References:** In Soft reference, even if the object is free for garbage collection then also its not garbage collected, until JVM is in need of memory badly.The objects gets cleared from the memory when JVM runs out of memory.To create such references java.lang.ref.SoftReference class is used.


SoftReference<Gfg> softref = new SoftReference<Gfg>(g);
g = null;
null
X
// g is now available for garbage collection,but the object is garbage collected only when JVM is in need of memory.The object can be retrieved by get() method of WeakReference class.
Gfg class Object
softref

```
//Code to illustrate Soft reference
import java.lang.ref.SoftReference;
class Gfg
{
    //code..
    public void x()
    {
        System.out.println("GeeksforGeeks");
    }
}

public class Example
{
    public static void main(String[] args)
    {
        // Strong Reference
        Gfg g = new Gfg();
        g.x();

        // Creating Soft Reference to Gfg-type object to which 'g'
```

```
            // Now, Gfg-type object to which 'g' was pointing
            // earlier is available for garbage collection.
            g = null;

            // You can retrieve back the object which
            // has been weakly referenced.
            // It successfully calls the method.
            g = softref.get();

            g.x();
        }
    }
```

Output:

```
GeeksforGeeks
GeeksforGeeks
```

- **Phantom References:** 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.To create such references java.lang.ref.PhantomReference class is used.

```
//Code to illustrate Phantom reference
import java.lang.ref.*;
class Gfg
{
    //code
    public void x()
}

public class Example
{
    public static void main(String[] args)
    {
        //Strong Reference
        Gfg g = new Gfg();
        g.x();

        //Creating reference queue
        ReferenceQueue<Gfg> refQueue = new ReferenceQueue<Gfg>();

        //Creating Phantom Reference to Gfg-type object to which 'g'
        //is also pointing.
        PhantomReference<Gfg> phantomRef = null;

        phantomRef = new PhantomReference<Gfg>(g,refQueue);

        //Now, Gfg-type object to which 'g' was pointing
        //earlier is available for garbage collection.
        //But, this object is kept in 'refQueue' before
        //removing it from the memory.
        g = null;

        //It always returns null.
        g = phantomRef.get();

        //It shows NullPointerException.
        g.x();
    }
}
```

```
Exception in thread "main" java.lang.NullPointerException
    at Example.main(Example.java:31)
```

Output:

```
GeeksforGeeks
```

Previous Next

Related Articles

1. C/C++ Pointers vs Java References
2. Method References in Java with examples
3. Can an ArrayList Contain Multiple References to the Same Object in Java?
4. Difference Between Equality of Objects and Equality of References in Java
5. Bounded Types with Generics in Java
6. Comparison of double and float primitive types in Java
7. Types of JVM Garbage Collectors in Java with implementation details
8. How to Create Different Types of Cells in a Spreadsheet using Java?

10. Why Java Collections Cannot Directly Store Primitives Types?

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : nidhi_biet

Article Tags : Java

Practice Tags : Java

Improve Article Report Issue