CHAPTER 11

■ ■ ■

# Garbage Collection

## QUESTIONS AND EXERCISES

1. What is the difference between explicit and implicit memory allocation and memory reclamation?

**Answer:**

In explicit memory allocation, the programmer decides how much memory is needed, requests that amount of memory from the *allocator and returns it to memory pool* when the allocated memory is no longer needed.

In implicit memory allocation, the runtime computes the memory needed to store the type of data indicated by the programmer, allocates it to the running program. When the task is completed, the runtime will automatically reclaim all memory blocks. The process of returning memory to the pool is known *as memory reclamation or memory recycling.*

2. What is a dangling reference?

**Answer:**

A dangling reference is a memory reference that points to memory location that has already been reclaimed. Reading data using a dangling reference results in unpredictable behavior.

3. What is memory leak?

**Answer:**

Situation wherein a memory block is not returned to memory pool and thereby never reused is called memory leak.

4. What is garbage collection and a garbage collector?

**Answer:**

The process of automatic reclamation of unused memory is known as *garbage collection*. The program that performs garbage collection is known as a *garbage collector* or a *collector*.

5. Show two ways to call the garbage collector in your program.

**Answer:**

- `Runtime.getRuntime().gc()`
- `System.gc()`

6. What is the `finalize()` method? How is it used by the garbage collector?

**Answer:**

The garbage collector automatically invokes the `finalize()` method of an object before reclaiming the object's memory. You can perform cleanup work in the `finalize()` method.

Because all Java classes inherit from the `Object` class, the `finalize()` method can be invoked on all Java objects. Any class can override and implement its own version of the `finalize()` method. In JDK 9, the `finalize()` method of the `Object` class has been deprecated.

7. What does the Java runtime do before throwing an `OutOfMemoryError`?

**Answer:**

The JVM tries its best to free up memory of all unreachable objects before it throws a `OutOfMemoryError` error.

8. What is an object resurrection in Java?

**Answer:**

An object resurrection is a process in which, during finalization, an object resurrects itself by placing its reference in a reachable reference variable. Once it is reachable through an already reachable reference variable, the object is back to life.

9. How do you request that the Java runtime run the garbage collection?

**Answer:**

```
Runtime.getRuntime().gc()
```

10. Describe the uses of the `WeakReference<T>`, `SoftReference<T>`, and `PhantomReference<T>` classes.

**Answer:**

All these classes reference weak reference with a varying degree of weakness.

`WeakReference` objects does not prevent their referents from being garbage collected.

`SoftReference` objects are garbage collected at the discretion of the garbage collector in response to memory demand. They are typically used to implement memory-sensitive caches.

`PhantomReference` objects are enqueued after the garbage collector determines that their referents may be reclaimed. They are typically used to schedule post-mortem cleanup actions.

11. When do you use a `ReferenceQueue`?

**Answer:**

The `ReferenceQueue` class is used to place the references of `SoftReference`, `WeakReference`, and `PhantomReference` objects in a queue.

12. How is the `Cleaner` class, which was introduced in JDK9, used?

**Answer:**

`Cleaner` class helps to setup up and perform the cleanup work for objects when the objects become phantom reachable.