Garbage Collection

- What is garbage and how can we deal with it?

- Garbage collection schemes

  - Reference Counting
  - Mark and Sweep
  - Stop and Copy

How Objects are Created in Java

- An object is created in Java by invoking the new() operator.

- Calling the new() operator, the JVM will do the following:

  - allocate memory;
  - assign fields their default values;
  - run the constructor;
  - a reference is returned.

How Java Reclaims Objects Memory

- Java does not provide the programmer any means to destroy objects explicitly
- The advantages are
  - No *dangling reference* problem in Java
  - Easier programming
  - No *memory leak* problem

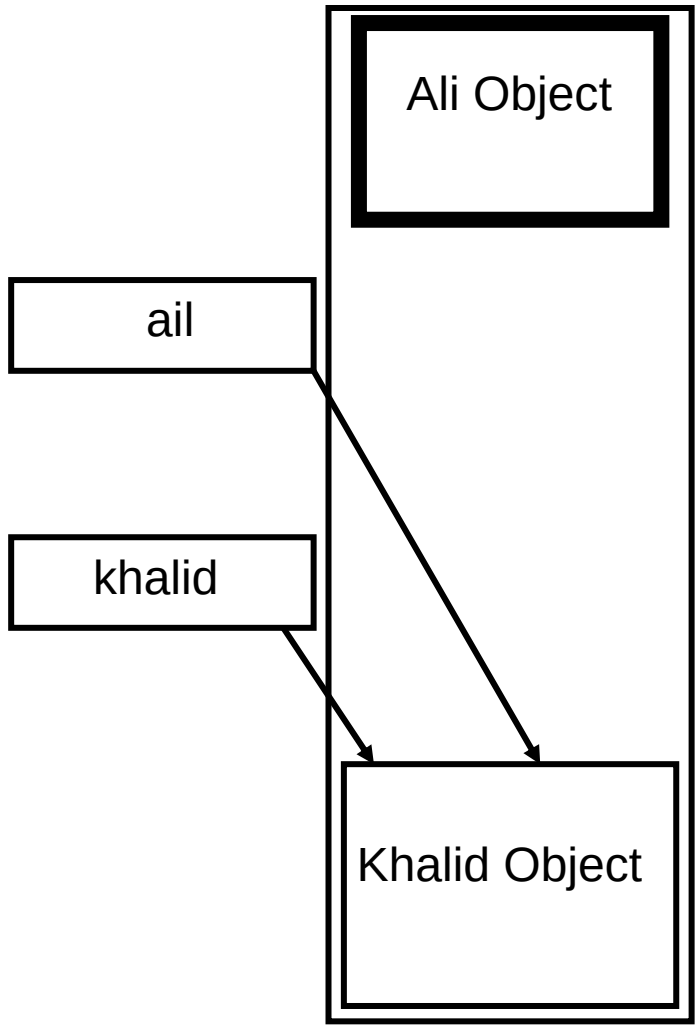
What is Garbage?

Garbage: unreferenced objects

```
Student ali= new Student();
Student khalid= new Student();
ali=khalid;
```

Now ali Object becomes a garbage,

It is unreferenced Object



What is Garbage Collection?

- *What is Garbage Collection?*
  - Finding garbage and reclaiming memory allocated to it.
- *Why Garbage Collection?*
  - the heap space occupied by an un-referenced object can be recycled and made available for subsequent new objects
- *When is the Garbage Collection process invoked?*
  - When the total memory allocated to a Java program exceeds some threshold.
- *Is a running program affected by garbage collection?*
  - Yes, the program suspends during garbage collection.

Advantages of Garbage Collection

- GC eliminates the need for the programmer to deallocate memory blocks explicitly
- Garbage collection helps ensure program integrity.
- Garbage collection can also dramatically simplify programs.

Disadvantages of Garbage Collection

- Garbage collection adds an overhead that can affect program performance.
- GC requires extra memory.
- Programmers have less control over the scheduling of CPU time.

Helping the Garbage Collector

- Reuse objects instead of generating new ones.
  - This program generates one million objects and prints them out, generating a lot of garbage

```
for (int i=0;i<1000000; ++i) {
    SomeClass obj= new SomeClass(i);
    System.out.println(obj);
}
```

  - Using only one object and implementing the setInt() method, we dramatically reduce the garbage generated.

```
SomeClass obj= new SomeClass();
for (int i=0;i< 1000000; ++i)   {
   obj.setInt(i);
  System.out.println(onj);
}
```

- Eliminate all references to objects that are no longer needed
  - This can be done by assigning null to every variable that refers to an object that is no longer needed

Common Garbage Collection Schemes

- Three main methods of garbage collection:
  - Reference counting
  - Mark-and-sweep
  - Stop-and-copy garbage collection.

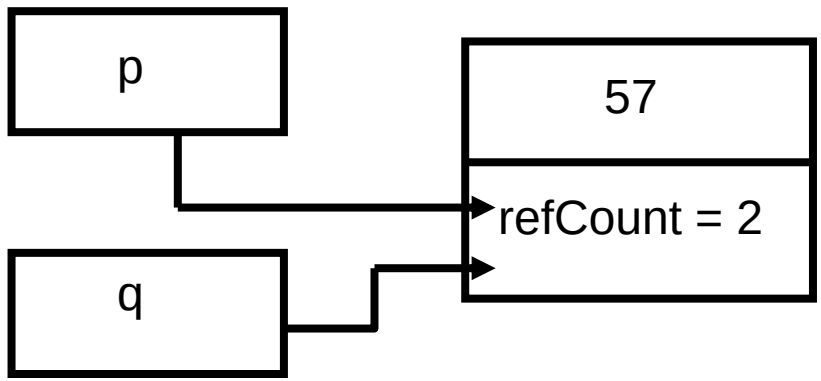
Reference Counting Garbage Collection

- Main Idea: Add a reference count field for every object.

- This Field is updated when the number of references to an object changes.

Example

```
Object p= new Integer(57);
Object q = p;
```

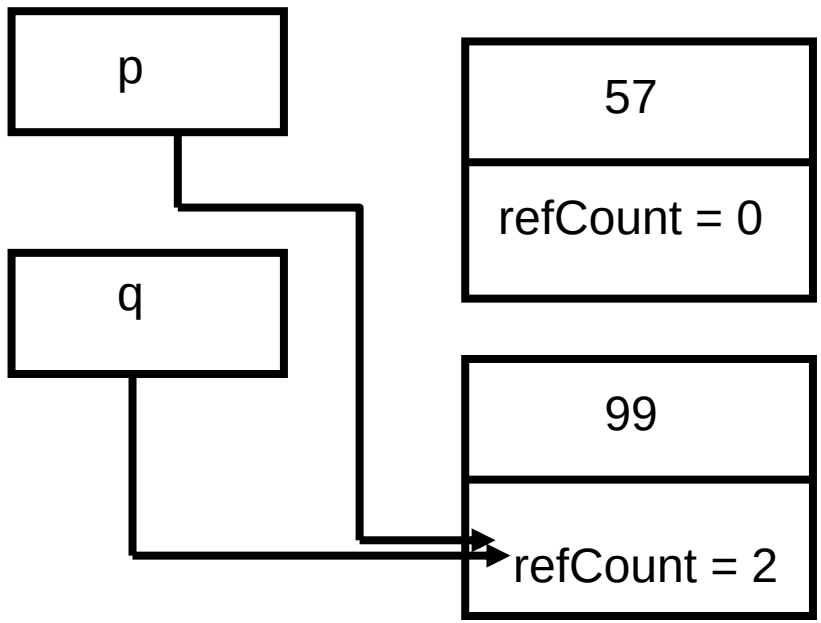


Reference Counting (cont'd)

- The update of reference field when we have a reference assignment (i.e p=q) can be implemented as follows

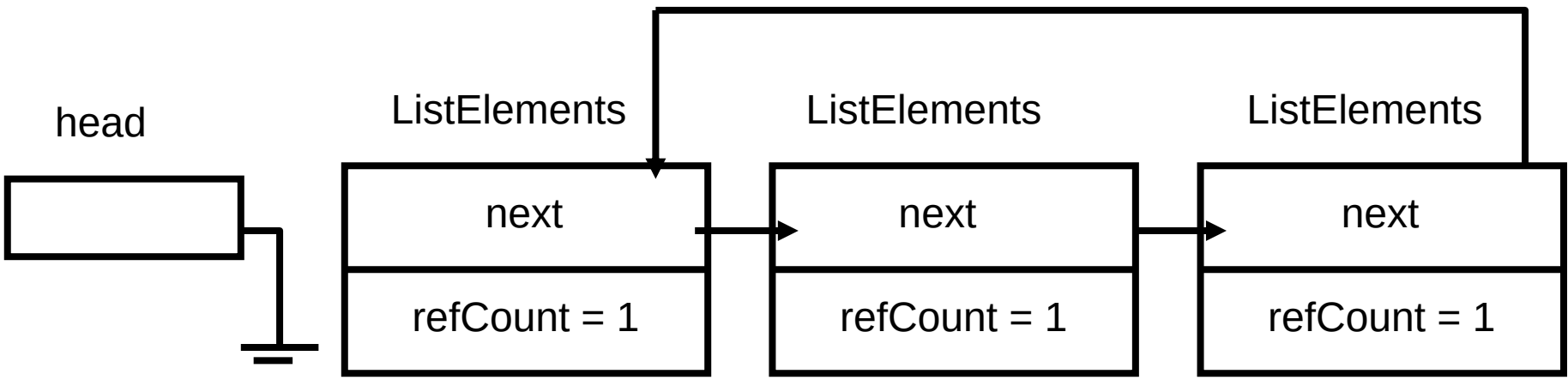
```
if (p!=q)
{
    if (p!=null)
      --p.refCount;
    p=q;
    if (p!=null)
      ++p.refCount;
}
```

Example:
Object p = new Integer(57);
Object q= new Integer(99);
p=q



Reference Counting (cont'd)

- *Reference counting will fail whenever the data structure contains a cycle of references and the cycle is not reachable from a global or local reference*



Reference Counting (cont'd)

- Advantages
  - Conceptually simple: Garbage is easily identified
  - It is easy to implement.
  - Immediate reclamation of storage
  - Objects are not moved in memory during garbage collection.

- Disadvantages
  - Reference counting does not detect garbage with cyclic references.
  - The overhead of incrementing and decrementing the reference count each time.
  - Extra space: A count field is needed in each object.
  - It may increase heap fragmentation.

Mark-and-Sweep Garbage Collection

- The mark-and-sweep algorithm is divided into two phases:
  - **Mark phase**: the garbage collector traverses the graph of references from the ***root nodes*** and marks each heap object it encounters. Each object has an extra bit: the mark bit – initially the mark bit is 0. It is set to 1 for the ***reachable objects*** in the mark phase.
  - **Sweep phase**: the GC scans the heap looking for objects with mark bit 0 – these objects have not been visited in the mark phase – they are garbage. Any such object is added to the free list of objects that can be reallocated. The objects with a mark bit 1 have their mark bit reset to 0.

Mark and Sweep (cont'd)

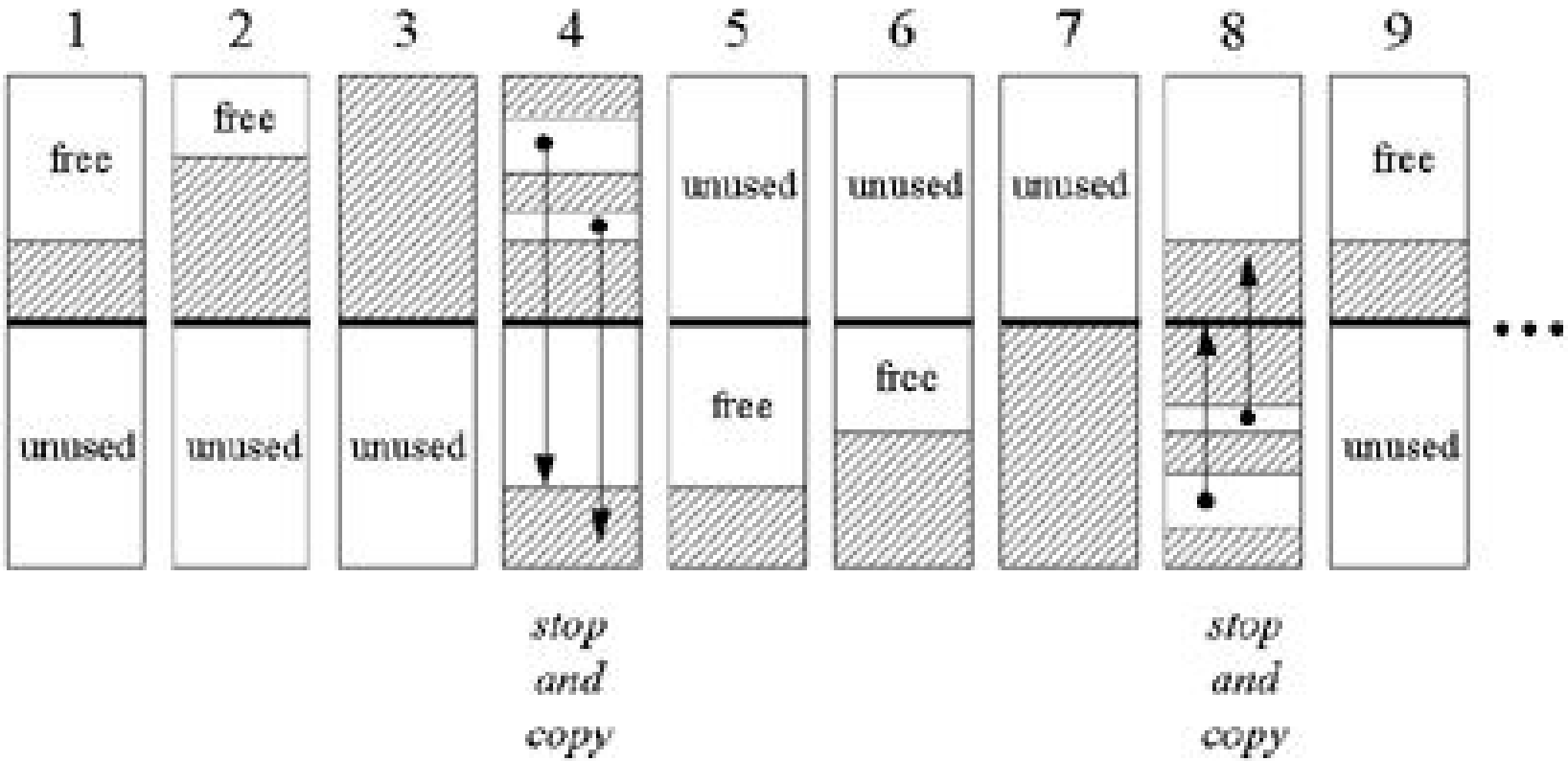
- Advantages
  - It is able to reclaim garbage that contains cyclic references.
  - There is no overhead in storing and manipulating reference count fields.
  - Objects are not moved during GC – no need to update the references to objects.
- Disadvantages
  - It may increase heap fragmentation.
  - It does work proportional to the size of the entire heap.
  - The program must be halted while garbage collection is being performed.

Stop-and-Copy Garbage Collection

- The heap is divided into two regions: Active and Inactive.
- Objects are allocated from the active region only.
- When all the space in the active region has been exhausted, program execution is stopped and the heap is traversed. Live objects are copied to the other region as they are encountered by the traversal. The role of the two regions is reversed, i.e., swap (active, inactive). …

Stop-and-Copy Garbage Collection (cont'd)

- A graphical depiction of a garbage-collected heap that uses a stop and copy algorithm. This figure shows nine snapshots of the heap over time:



Stop-and-Copy Garbage Collection (cont'd)

- Advantages
  - Only one pass through the data is required.
  - It de-fragments the heap.
  - It does work proportional to the amount of live objects and not to the memory size.
  - It is able to reclaim garbage that contains cyclic references.
  - There is no overhead in storing and manipulating reference count fields.

Stop-and-Copy Garbage Collection (cont'd)

- Disadvantages
  - Twice as much memory is needed for a given amount of heap space.
  - Objects are moved in memory during garbage collection (i.e., references need to be updated)
  - The program must be halted while garbage collection is being performed.