

# Solving Memory Leaks

---



**Richard Warburton**

@richardwarburto [www.monotonic.co.uk](http://www.monotonic.co.uk)





# Memory Leak

A memory leak occurs when memory that has been allocated and is no longer needed does not get released.



# Outline

## Memory Leaks?

Despite GC these can still happen in Java

## Heap Dumps

How to solve with heap dumps

## Memory Profiling

How to find with memory profiling



# Memory Leaks?

---



# Memory Leak

A memory leak occurs when memory that has been **allocated** and is no longer needed does not get **released**.



```
Public void foo() {  
    Object obj = new Object ();  
  
    //Method ends here  
}
```

---

## Allocation

**Calls new() allocate memory**





```
private Object obj;  
  
public void foo() {  
    obj = new Object();  
}
```

---

## Retaining a reference

**Assigning an object to a field keeps it alive.**





Garbage Collection does  
not prevent all memory  
leaks.



# Heap Dumps

---

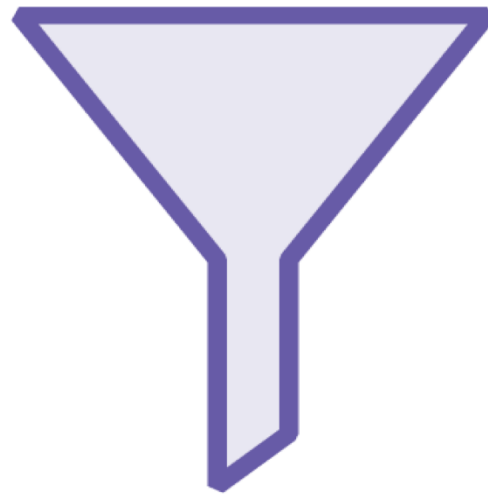


# Leak Finding Process



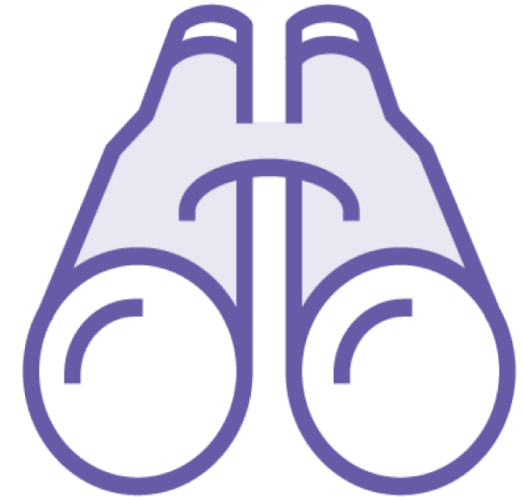
## Retained Heap

Find objects that  
cause the most heap  
to be retained



## Filter

Top objects will  
usually be noise



## Investigate

Look at what these  
objects are  
referencing or being  
referenced by



# Demo



Klassified is a klassified ads web service

We'll fix a memory leak in it

Using a heap dump



# Memory Profiling

---



# Generational Ages

## Age of Object

Number of Garbage  
Collections survived

## Generational Count

Number of different ages of  
objects surviving



Classes with increasing  
generational counts are leak  
candidates





# Demo



We'll fix a similar memory leak

This time in a simpler and more cut down application

Using it to illustrate the memory profiling approach



# Conclusions

---



# Summary



Memory can leak despite Garbage Collection

Find the unused references and break them.

They can be found using both memory profiling and heap dumps

