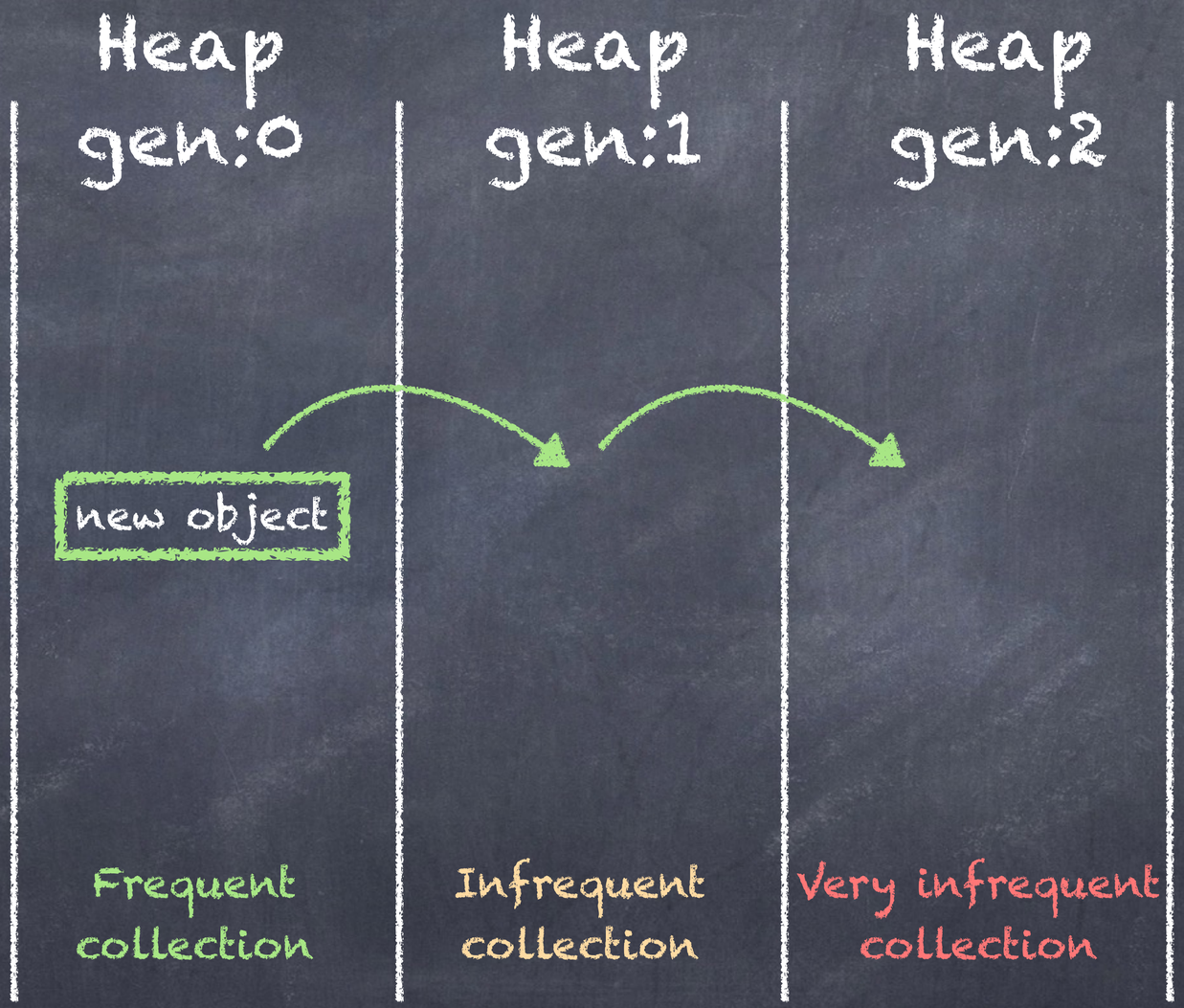
Fast garbage collection part 2

Recap

- The Garbage Collector uses a mark & sweep/compact cycle
- There are 2 heaps: the Small Object Heap (SOH) and the Large Object Heap (LOH)
- The Small Object Heap has 3 generations. Objects are allocated in gen 0 and progress towards gen 2.
- Gen 0 is collected frequently, gens 1 and 2 infrequently

Generations help to reduce the number of objects in gen 0



Gen:0 optimizations

The more objects in generation 0, the more work the Garbage Collector has to do. So:

- Limit the number of objects you create
- Allocate, use, and discard objects as quickly as possible

```
StringBuilder s = new StringBuilder();
for (int i=0; i < 10000; i++)
{
    s.Append(i.ToString() + "KB");
}
```

```
StringBuilder s = new StringBuilder();
for (int i=0; i < 10000; i++)
{
    s.Append(i.ToString() + "KB");
}
```

```
StringBuilder s = new StringBuilder();
for (int i=0; i < 10000; i++)
{
    s.Append(i);
    s.Append("KB");
}
```

```
ArrayList list = new ArrayList();
for (int i=0; i < 10000; i++)
{
   list.Add(i);
}
```

```
ArrayList list = new ArrayList();
for (int i=0; i < 10000; i++)
{
    list.Add(i);
}
```

```
List<int> list = new List<int>();
for (int i=0; i < 10000; i++)
{
    list.Add(i);
}
```

```
public static MyObject obj = new MyObject();
...
// lots of other code
...
UseTheObject(obj);
```

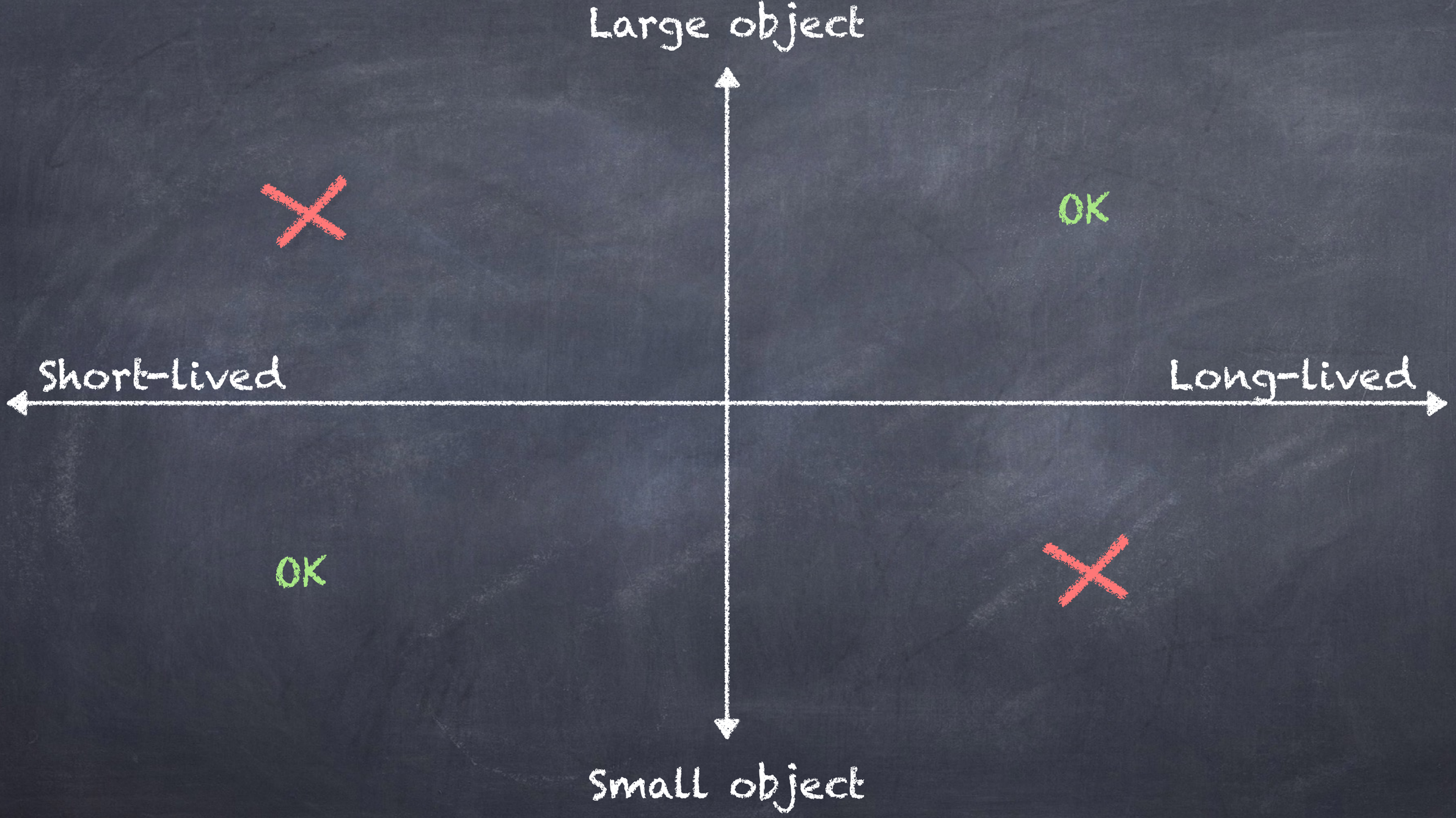
```
public static MyObject obj = new MyObject();
...
// Lots of other code
...
UseTheObject(obj);
```

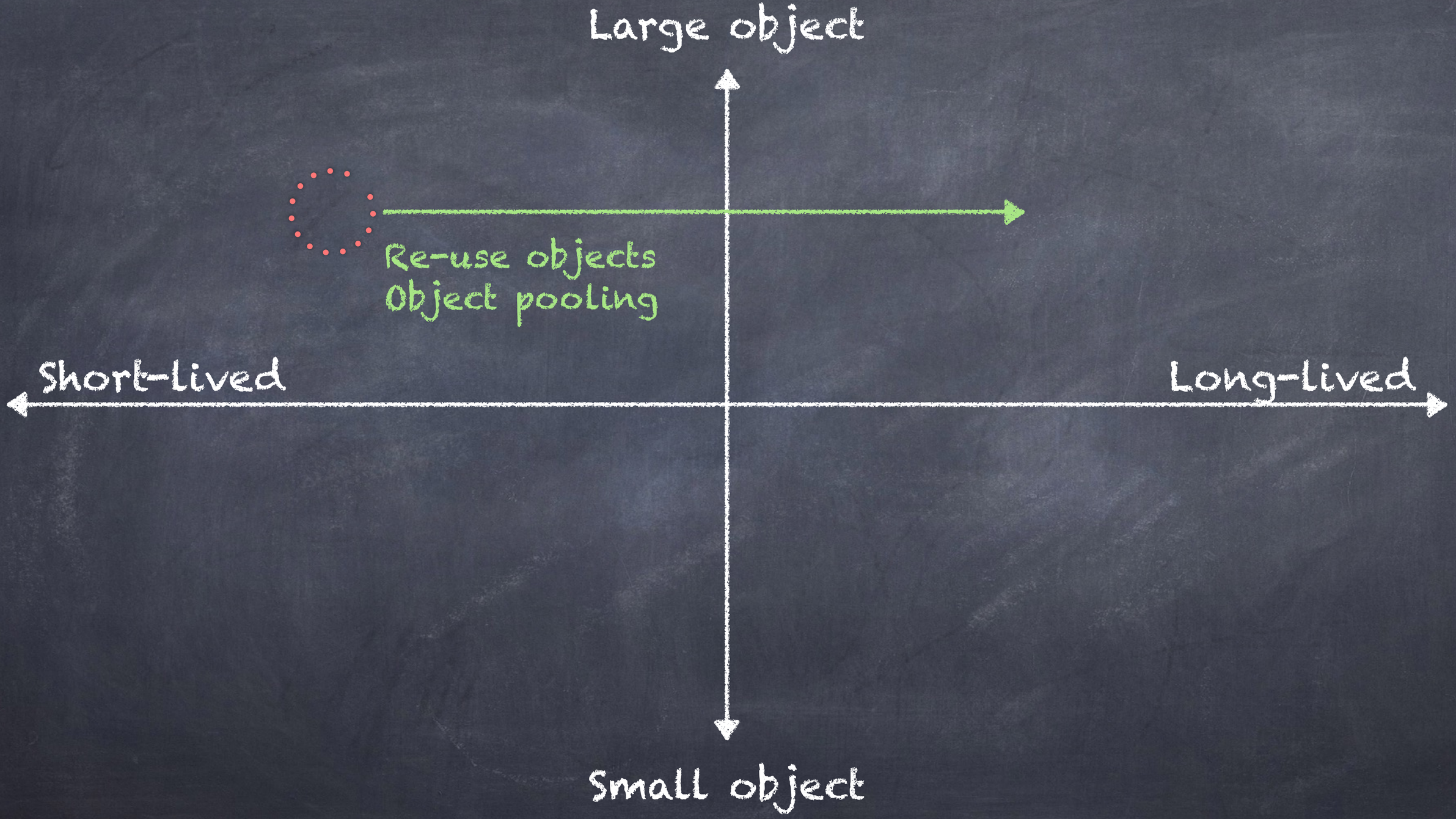
```
.....
MyObject obj = new MyObject();
UseTheObject(obj);
obj = null;
.....
```

Lifetime optimizations

The Garbage Collector assumes 90% of all small objects are short-lived, and all large objects are long-lived. So:

- Avoid large short-lived objects
- Avoid small long-lived objects

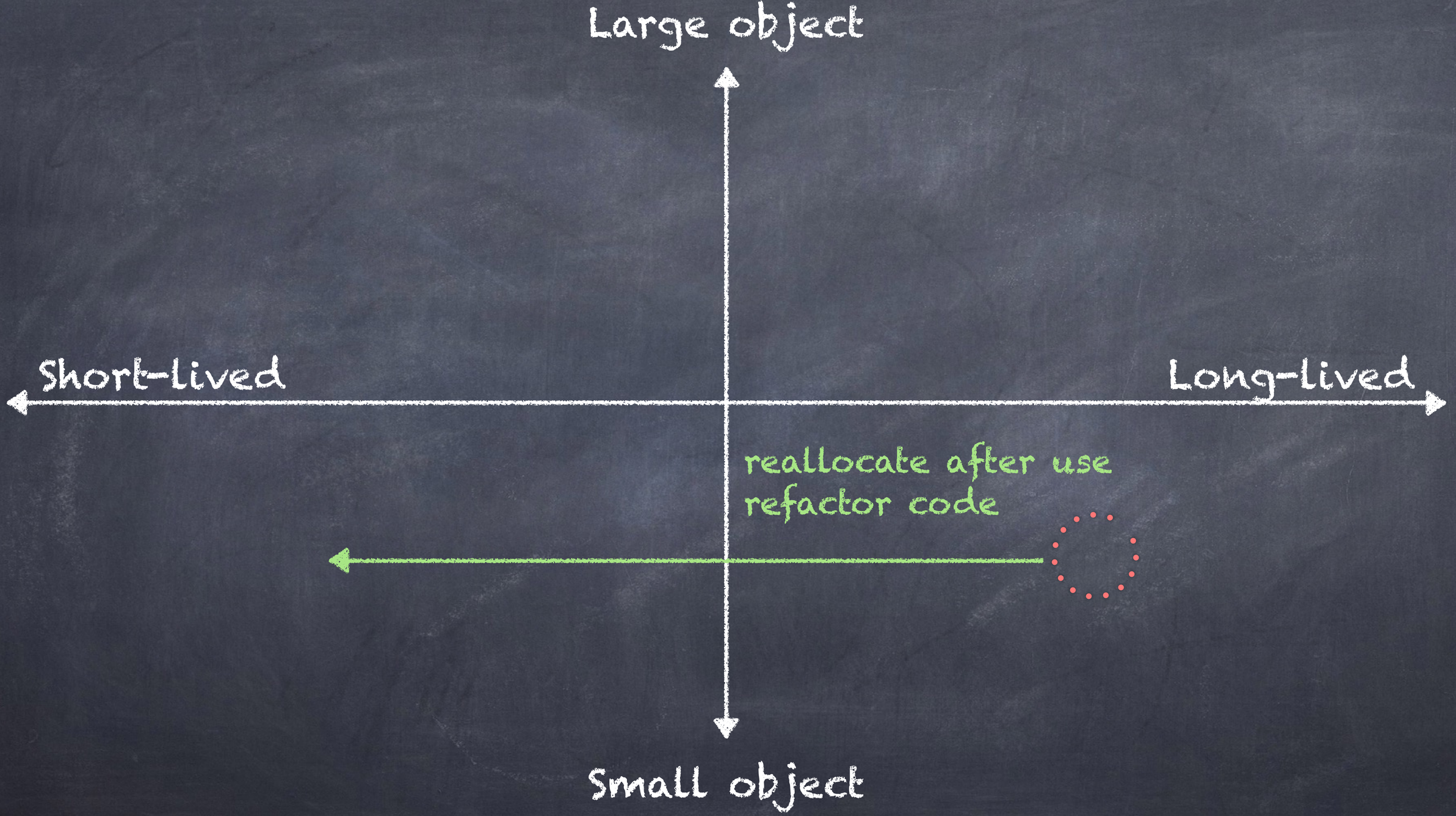

Large object
OK
Short-lived
Long-lived
OK
Small object


Large object
Re-use objects
Object pooling
Short-lived
Long-lived
Small object

```
ArrayList list = new ArrayList(85190);
UseTheList(list);
...
list = new ArrayList(85190);
UseTheList(list);
```

```
ArrayList list = new ArrayList(85190);
UseTheList(list);
...
list = new ArrayList(85190);
UseTheList(list);
```

```
ArrayList list = new ArrayList(85190);
UseTheList(list);
...
list.Clear();
UseTheList(list);
```


Large object
Short-lived
Long-lived
reallocate after use
refactor code
Small object

```
ArrayList list = new ArrayList(85190);
for (int i=0; i < 10000; i++)
{
   list.Add(new Pair(i, i+1));
}
```

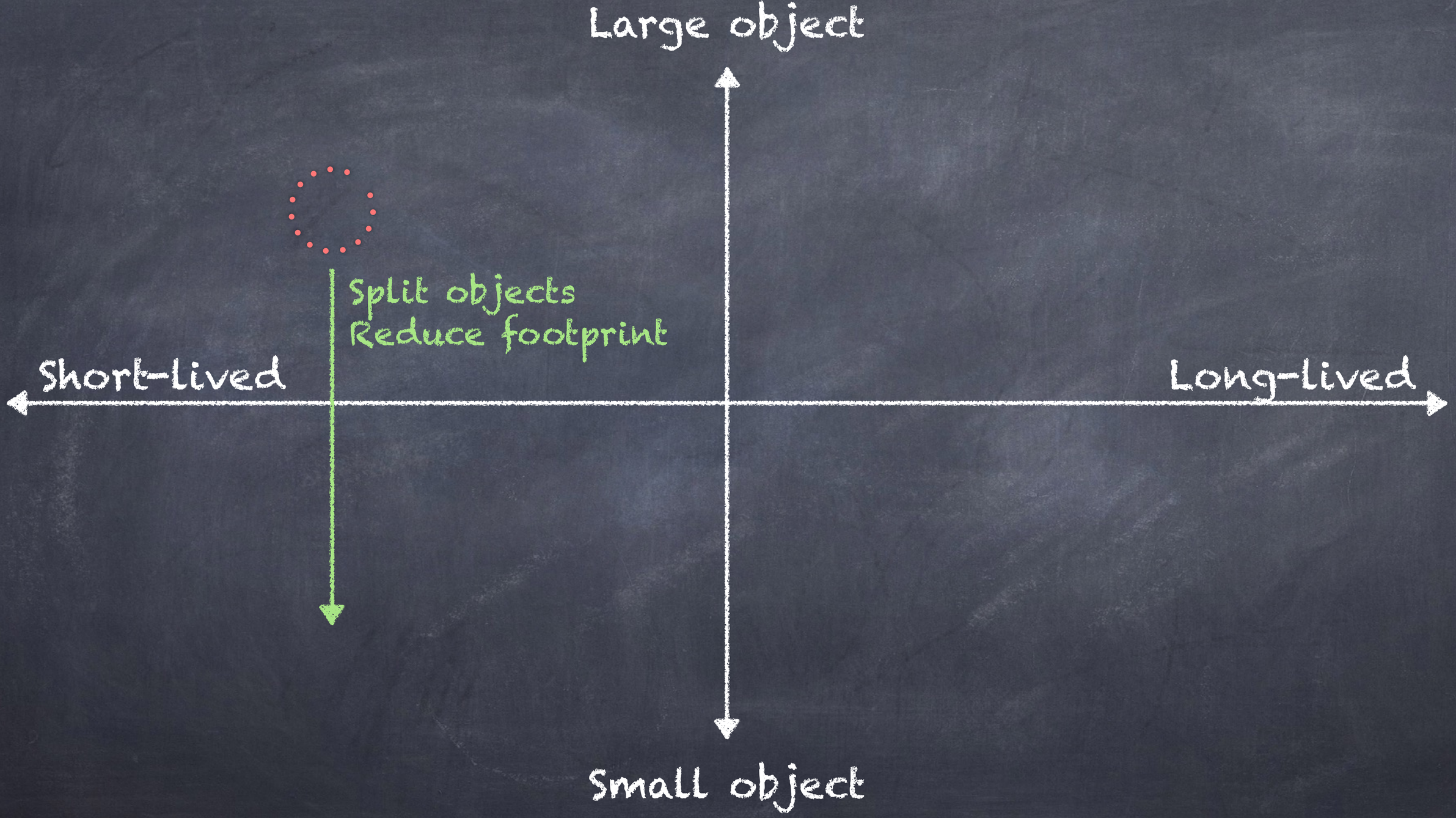
```
ArrayList list = new ArrayList(85190);
for (int i=0; i < 10000; i++)
{
   list.Add(new Pair(i, i+1));
}
```

```
int[] list_1 = new int[85190];
int[] list_2 = new int[85190];
for (int i=0; i < 10000; i++)
{
   list_1[i] = i;
   list_2[i] = i + 1;
}
```

Size optimizations

The Garbage Collector assumes 90% of all small objects are short-lived, and all large objects are long-lived. So:

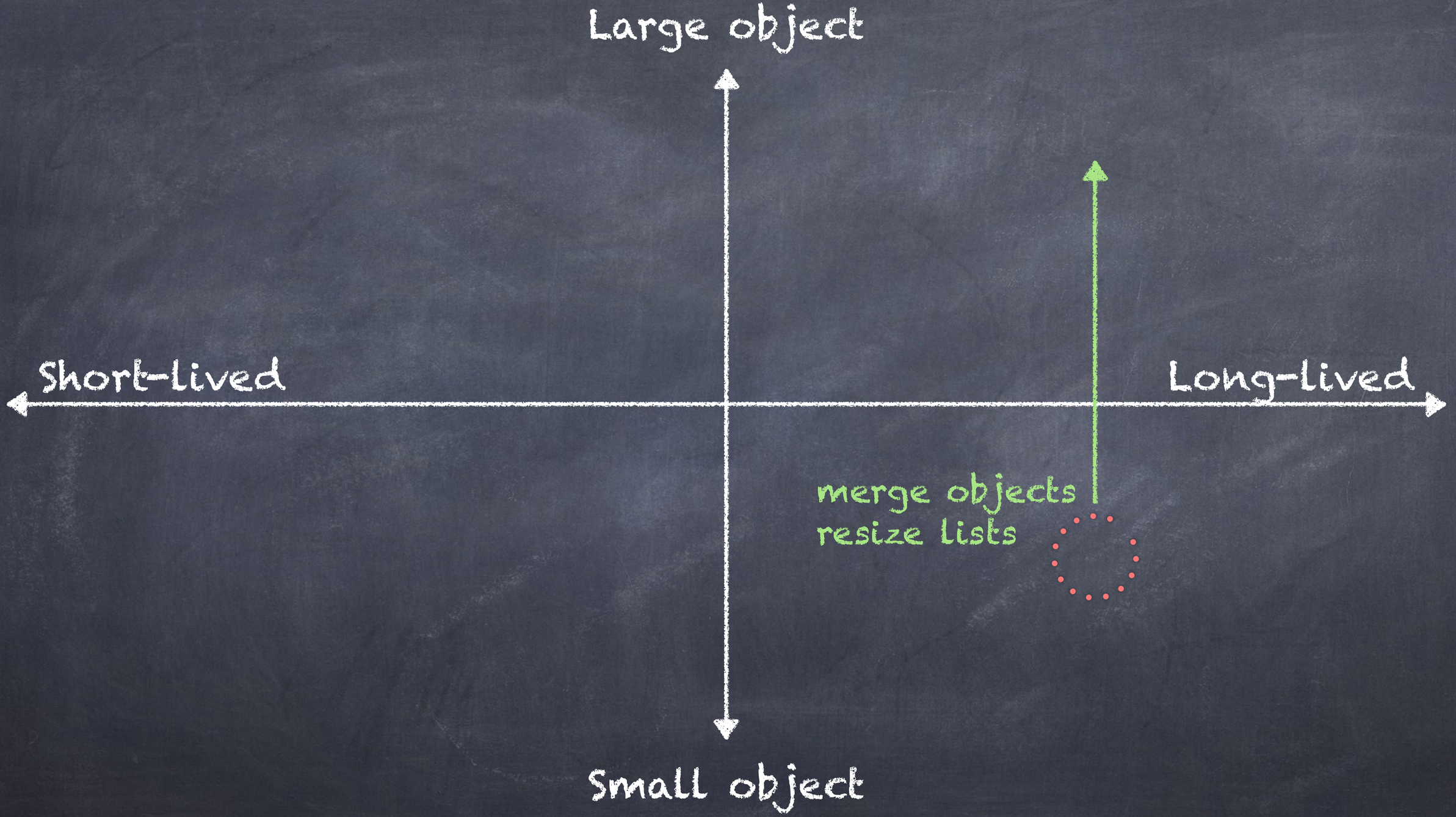
- Avoid large short-lived objects
- Avoid small long-lived objects


Large object
Split objects
Reduce footprint
Short-lived
Long-lived
Small object

```
int[] buffer = new int[32768];
for (int i=0; i < buffer.Length; i++)
{
   buffer[i] = GetByte(i);
}
```

```
int[] buffer = new int[32768];
for (int i=0; i < buffer.Length; i++)
{
   buffer[i] = GetByte(i);
}
```

```
byte[] buffer = new byte[32768];
for (int i=0; i < 10000; i++)
{
   buffer[i] = GetByte(i);
}
```


Large object
Short-lived
Long-lived
merge objects
resize lists
Small object

```
public static ArrayList list = new ArrayList();
...
// lots of other code
...
UseTheList(list);
```

```
public static ArrayList list = new ArrayList();
...
// lots of other code
...
UseTheList(list);
```

```
public static ArrayList list = new ArrayList(85190);
...
// lots of other code
...
UseTheList(list);
```

What have we learned

- Limit the number of objects you create
- Allocate, use, and discard small objects as fast as possible
- Re-use large object

- Use only small short-lived, and large long-lived objects

- Increase lifetime or decrease size of large short-lived objects
- Decrease lifetime or increase size of small long-lived objects