Java™

ORACLE®

Live Java Heap, String Compaction and Future

Vaibhav Choudhary (@vaibhav_c)
Java Platforms Team
https://blogs.oracle.com/vaibhav

Java
Your
Next
(Cloud)

Safe Harbor Statement

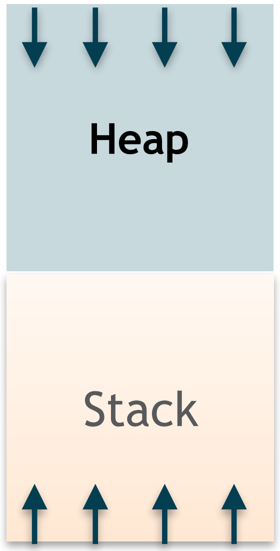
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

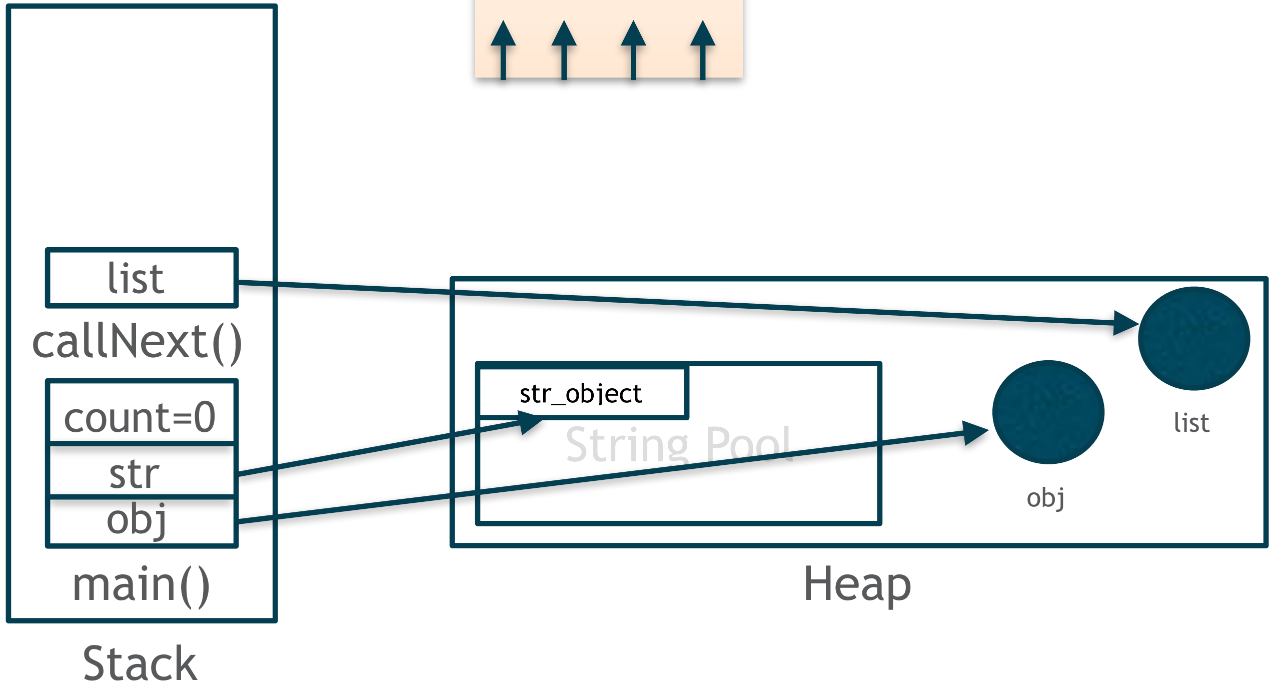
Agenda of the day …

1. Understanding Stack and Heap in Java Space
2. Data Structure to Heap Mapping
3. Code optimisation to reduce Memory Footprint
4. String changes in Java9 to reduce Java heap
5. Future Data Structures

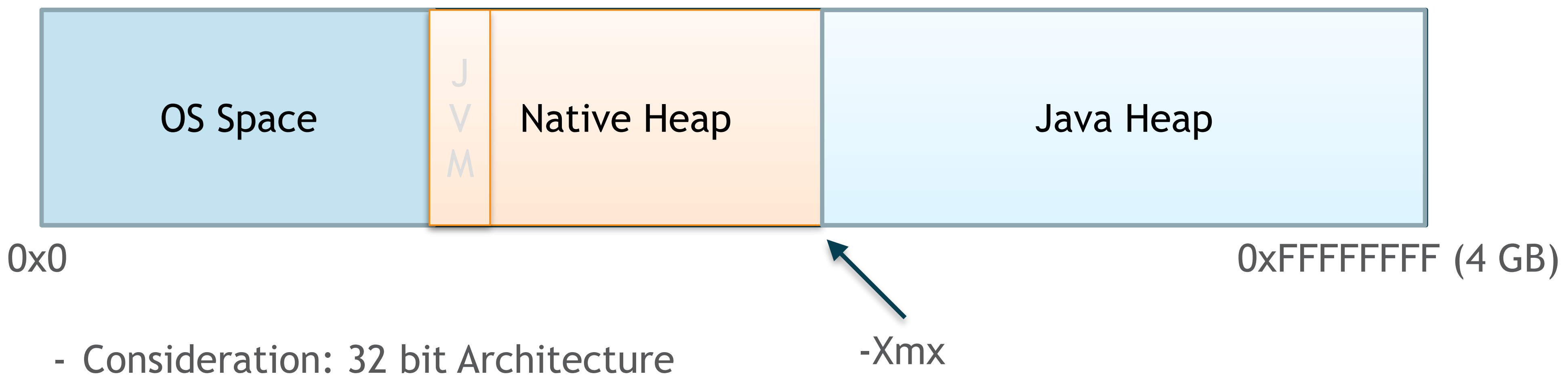
Stack and Heap in Java

```
public static void main(String[] args) {

     Object obj = new Object();
     String str = new String();
     int count = 0;
     // Do some work here
     callNext();

}

public void callNext() {

     List list = new ArrayList();
     // some work
}
```



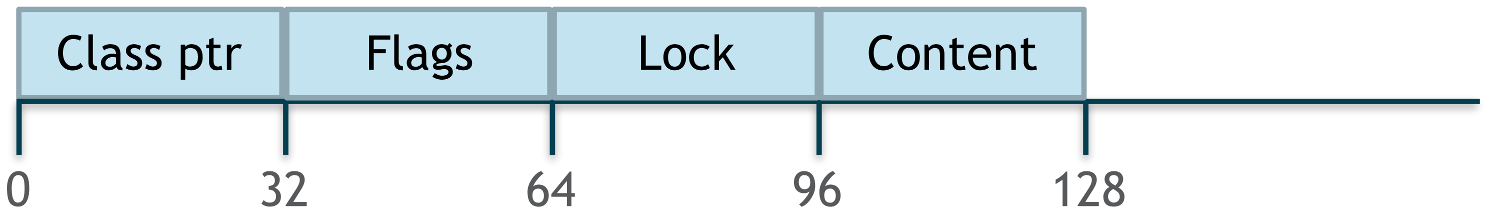


Java Heap And Native Heap



- Consideration: 32 bit Architecture
- Native Heap also covers JVM Heap

Java Heap Occupancy with Object - 32 bit



- Integer integer = new Integer(10);
- Total Space = 4*x
- Overhead Solutions ? (Interested party should see the work on value types)
- For 64 bit architecture, it will be 7x more. (Also learn about CompressedOops, Demo)

Collections - Memory overhead vs Functionality

- Memory Overhead

  ArrayList < LinkedList < HashMap < HashSet

- More in functionality

  ArrayList < LinkedList < HashMap < HashSet

- Be a smart selector

Heap Eaters

- Depends on application.
- String, char[], byte[] … Mostly takes a good amount of heap.
- Collections ?

- Analysis of heap is required.
- Any unknown “heap eater” is not good.

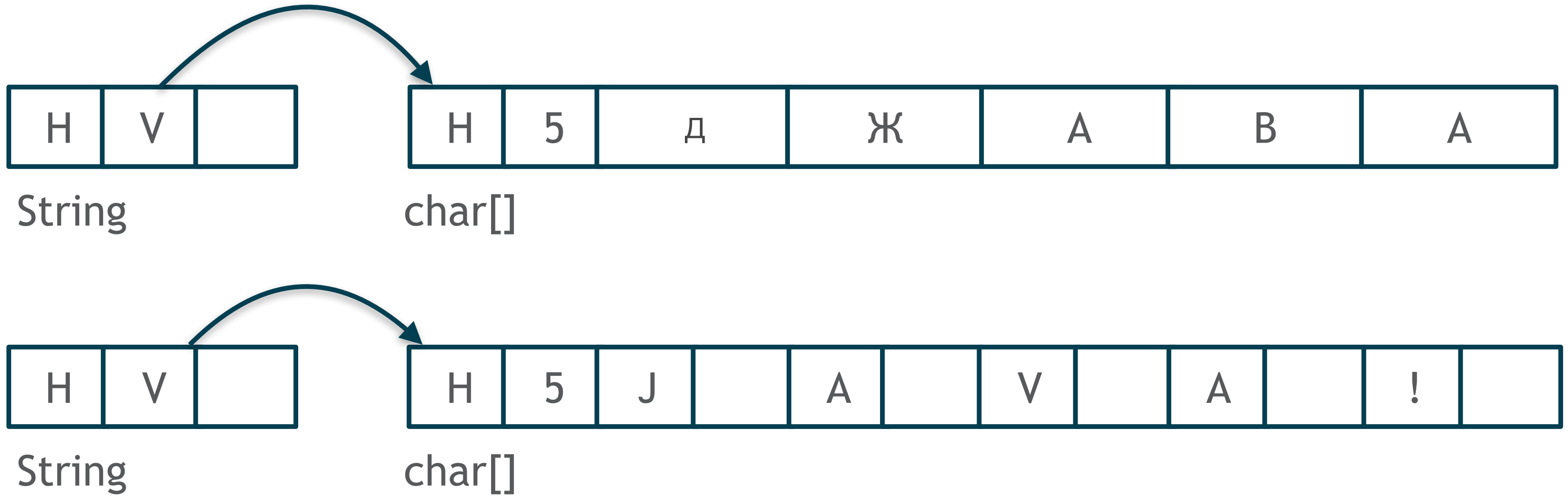
Minimizing Memory

- Lazy Initialization of Collections
  – Not a great idea to create till required.
- Don’t create single object collections.
- Too much expansion can be risky.
- Use the right choice of collection.
- Collections default sizing.
- Use Memory Analyzer Tools

Compact String - JEP 254

- Internal change in the String class
  - **String encoding based on the string content.**
  - **Purely implementation change, so no impact on API.**
- Good reduction in memory footprint.
- Demo

String internal Representation



It’s good to not have those empty spaces !

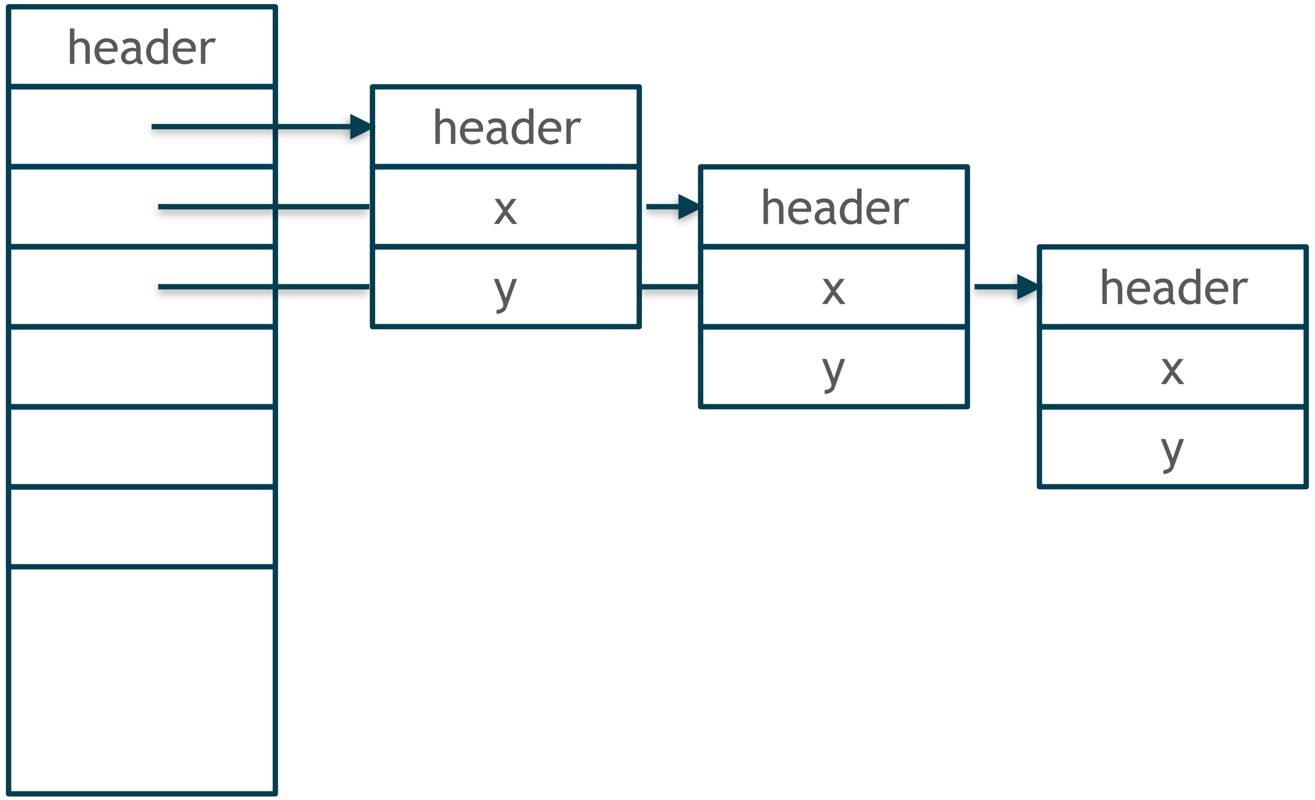
Compact String - JEP 254

- Internal change in the String class
  - **String encoding based on the string content.**
  - **Purely implementation change, so no impact on API.**
- Good reduction in memory footprint.
- Demo

A future resolution - Value types

- Under the name Project Valhalla

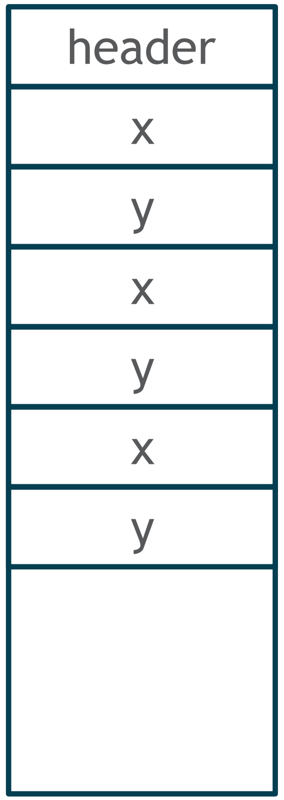
```
class Point {
    final int x;
    final int y;
}

Point[] point;
```



A future resolution - Value types

- Overhead should be very less if

```
class Point {
    final int x;
    final int y;
}

Point[] point;
```



A future resolution - Value types

- Backward compatibility ?
- if(point[i] == p) ?
- synchronized (point[i]) ?
- Proposal for the new data type

```
value class Point {
      int x;
      int y;
}
```

Important References

- Must View links :-
  - Aleksey Shipilëv on Compact Strings
  - Beyond Java 9 by Mark Reinhold
  - From Java Code to Java Heap

- I write at : https://blogs.oracle.com/vaibhav