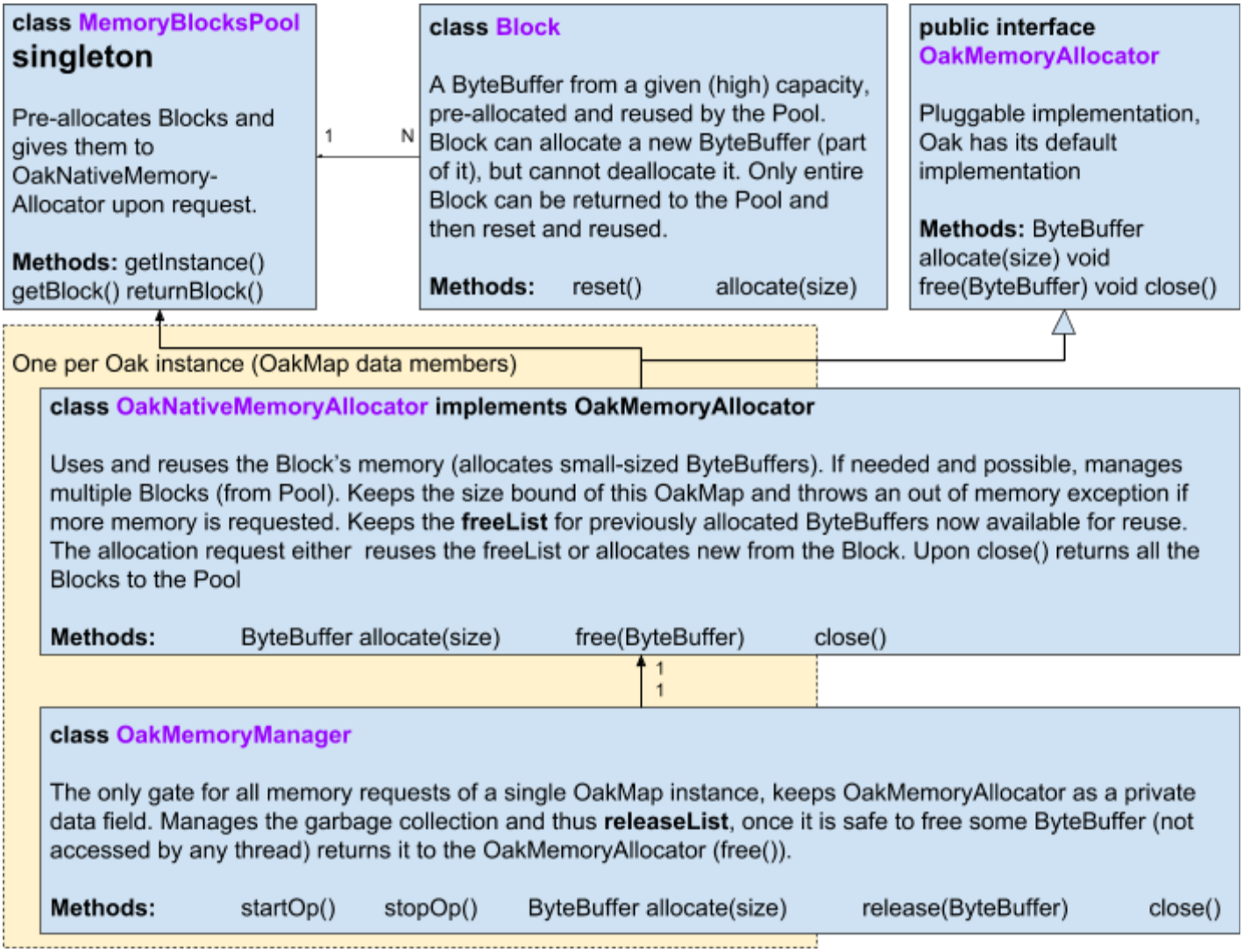
New Memory Management Structure

[It is assumed the reader is familiar with OakMap data structure. This document speaks only about memory management.]

Hereby please find the class diagram of the new memory management structure for Oak. It shows the default memory allocation schema and allows user to provide its own memory allocator. In case user-defined memory allocator is used, the off-heaping of the underlying ByteBuffers is up to the provided allocator.

Default Oak memory management schema is

1. Thread safe
2. Allows allocation of any amount of memory for multiple Oak instances (up to OS/HW boundaries)
3. Currently works only with ByteBuffers
4. Keeps a memory size bound per OakMap, meaning if more than predefined amount of memory (memory capacity) is requested per single OakMap instance the exception will be thrown.



How to create OakBuilder with user-defined MemoryAllocator or with default OakMemoryAllocator. As a reminder the way to create an OakMap instance is to create an OakBuilder and to use its *build()* method later. Here is the code example:

```
OakMapBuilder<K,V> builder = new OakMapBuilder()
        .setKeySerializer(new OakKeySerializerImplementation(...))
        .setValueSerializer(new OakValueSerializerImplementation(...))
        .setMinKey(...)
        .setKeysComparator(new OakKeyComparatorImplementation(...))
        .setMemoryCapacity(...);
```

```
OakMap<K,V> oak = builder.build();
```

In the above example an user-defined MemoryAllocator wasn't used, thus any OakMap build by this builder will be created with the Oak's native OakMemoryAllocator. In order to create a builder with user-defined MemoryAllocator see the code example below:

```
OakMapBuilder<K,V> builder = new OakMapBuilder()
        .setKeySerializer(new OakKeySerializerImplementation(...))
        .setValueSerializer(new OakValueSerializerImplementation(...))
        .setMinKey(...)
        .setKeysComparator(new OakKeyComparatorImplementation(...))
        .setMemoryCapacity(...)
       .setMemoryAllocator(new SpecialMemoryAllocator(...));

OakMap<K,V> oak = builder.build();
```