Task 1.2

To address this task, we need to implement two classes in Java: MapContainer<Value> and SimpleFileContainer<Value>.

MapContainer<Value>

Map<Long, Value> for sorting the objects in memory

```
Import java.util.HashMap;
Import java.util.Map;

Public class MapContainer<Value> implements
                    Java.util.Map<Long, Value> {
Private Map<Long, Value> container;
Private long, nextKey;

Public MapContainer(){
    this, container = new HashMap<>();
    this.nextKey = 0L;
}
```

```
Public Long put (value value) {
    Long key = nextKey++;
    contains.put (key, value);
    return key;
}

@override
public value get (Object key) {
    return contains.get (key);
}

@override
Public value remove (Object key) {
    return container.remove (key);
}
}
```

part (6): SimpleFileContainer<Value>

The SimpleFileContainer<Value> class will implement a file based container that uses serialization to write data to a file. Here are the requirements and steps.

Sample implementation of SimpleFileContainer<Value>

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;

public class SimpleFileContainer<Value> {
    private final File dataFile;
    private final File metadataFile;
    private final FixedSizeSerializer<Value>
    private long nextKey;                    serializer;
    private final Map<Long, Long> keyOffsetMap;

public SimpleFileContainer(File dataFile, File
                    metadataFile, FixedSizeSerializer<Value
this.dataFile = dataFile;
this.metadataFile = metadataFile;
```

```
        this.serializer = serializer;
        this.nextKey = 0;
        this.keyOffsetMap = new HashMap<>();
    }

    public long put(Value value) throws
                                IOException {
        try (RandomAccessFile raf = new RandomAccessFile(dataFile, "rw")) {
            long offset = raf.length();
            raf.seek(offset);
            raf.writeByte(1);
            raf.write(serializer.serialize(value));
            keyOffsetMap.put(nextKey, offset);
            return nextKey++;
        }
    }
```

```
Public value get (Long key) throws IOException {
    Long offset = keyoffsetmap.get (key);
    if (offset == null) return null;

    try (RandomAccessFile raf = new RandomAccessFile
                                (datafile, "r")) {
      raf.seek (offset);
    if (raf.readbyte () == 0) return null;
    byte [] data = new byte [serializer.getsize()];
    raf.readfully (data);
    return serializer.deserialize (data);
}}
}

Public void remove (Long key) throws IOException {
    Long offset = keyoffsetmap.get (key);
    if (offset != null) {
      try (RandomAccessFile raf = new RandomAccessFile (datafile, "rw")) {
        raf.seek (offset);
        raf.writeByte (0);
}}}
}
```