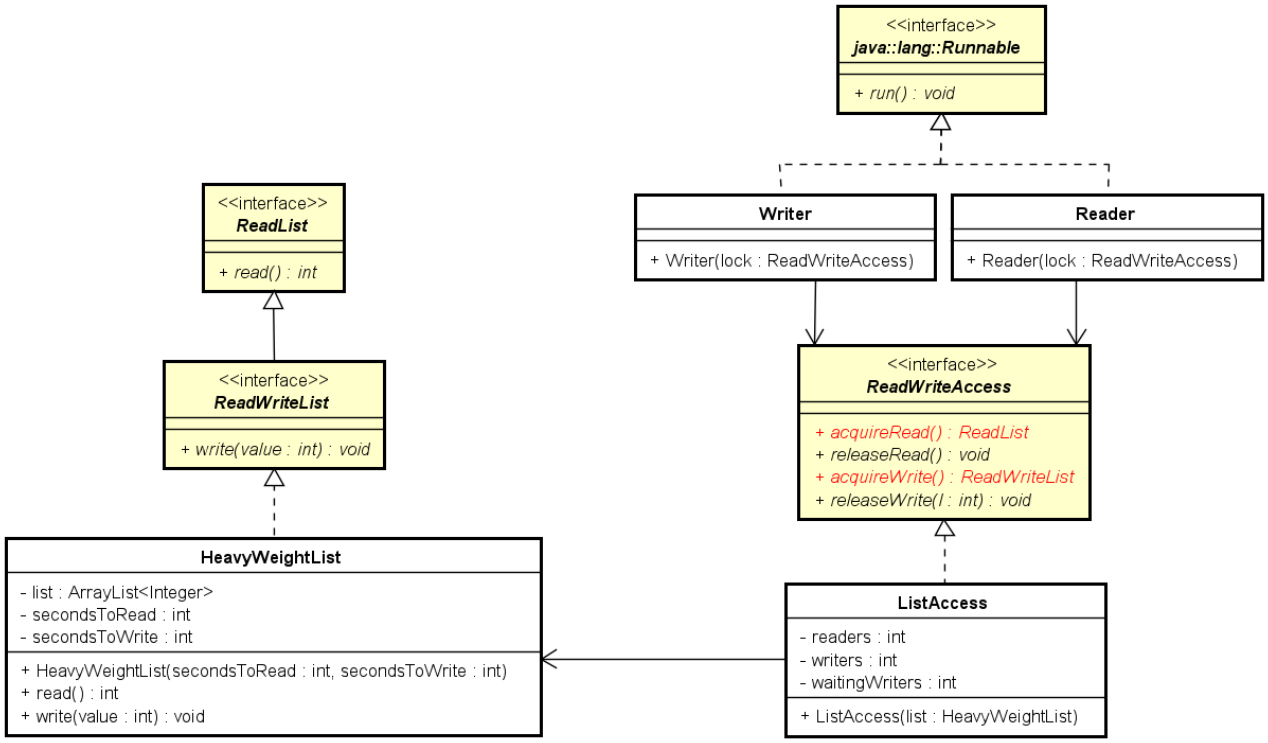
Exercise: Readers-Writers with access to a model object

Implement the following Readers-Writers problem exercise. Class `HeavyWeightList` and interfaces `ReadList` and `ReadWriteList` are given at the end of this document.



Part 1

Create interface `ReadWriteAccess`.

Part 2

Implement class `ListAccess`. In the diagram above it is shown with Writer preference, but It is also ok to implement it with either Reader preference or Fair.
Both of the `acquire` methods return the `HeavyWeightList` object.
Insert proper printouts, each including action, name of thread, and values for the instance variables (not including the HeavyWeightList instance variable)

Part 3

Implement class `Writer`. In the `run` method loop 3 times waiting a random time between 5 and 10 seconds, acquire write access (to get a ReadWriteList object), call method write, and release write access.

Part 4

Implement class `Reader`. In the `run` method loop 3 times waiting a random time between 1 and 10 seconds, acquire read access (to get a ReadList object), call method read, and release read access.

Part 5

Implement a class with a `main` method in which you 1) Create a HeavyWeightList with secondsToRead set to 3 and secondsToWrite to 5 seconds, 2) Create the shared resource (declared as a `ReadWriteAccess` and created as a `ListAccess`) and 3) Create and start 25 Reader threads and 2 Writer threads.

```java
public interface ReadList
{
  int read();
}
```

```java
public interface ReadWriteList extends ReadList
{
  void write(int value);
}
```

```java
import java.util.ArrayList;

public class HeavyWeightList implements ReadWriteList
{
  private ArrayList<Integer> list;
  private int secondsToRead;
  private int secondsToWrite;

  public HeavyWeightList(int secondsToRead, int secondsToWrite)
  {
    list = new ArrayList<>();
    this.secondsToRead = secondsToRead;
    this.secondsToWrite = secondsToWrite;
  }

  @Override public void write(int value)
  {
    list.add(value);
    if (list.size() > 10000)
    {
      list.remove(0);
    }
    simulateThatItTakesTime(secondsToWrite);
  }

  @Override public int read()
  {
    int sum = 0;
    for (int i=0; i<list.size(); i++)
    {
      if (sum > Integer.MAX_VALUE - list.get(i))
      {
        sum = Integer.MAX_VALUE;
      }
      else
      {
        sum += list.get(i);
      }
    }
    simulateThatItTakesTime(secondsToRead);
    return sum;
  }

  private void simulateThatItTakesTime(int seconds)
  {
    try {Thread.sleep(seconds*1000);}
    catch (InterruptedException e) {  /* empty*/ }
  }
}
```