Counter

Implement the class **`Counter`** with attribute **`value`**, initialized to zero, and the methods **`increment`** and **`decrement`**.

```
public class Counter {









}
```

Resource

Implement the class **`Resource`** with attribute **`available`**, initialized to true, and the methods **`acquire`** and **`release`**.

```
public class Resource {









}
```

Implement the class **`User`** that extends the **`Thread`** class. The class contains a static attribute of type **`Resource`**, a resource shared among users. In its run method, it acquires and subsequently releases that resource.

```
public class User extends Thread {









}
```

The readers-writers problem

The readers and writers problem, due to Courtois, Heymans and Parnas, is a classical concurrency problem. It models access to a database. There are many competing threads wishing to read from and write to the database. It is acceptable to have multiple threads reading at the same time, but if one thread is writing then no other thread may either read or write. A thread can only write if no thread is reading.

```
public class Reader extends Thread {
  private Database database;

  public Reader(Database database) {
    this.database = database;
  }

  public void run() {
    this.database.read();
  }
}
```

```
public class Writer extends Thread {
  private Database database;

  public Writer(Database database) {
    this.database = database;
  }

  public void run() {
    this.database.write();
  }
}

public class Database {
  ...
  public Database() { ... }
  public void read() { ... }
  public void write() { ... }
}

final int READERS = 5;
final int WRITERS = 2;
Database database = new Database();
for (int r = 0; r < READERS; r++) {
  (new Reader(database)).start();
}
for (int w = 0; w < WRITERS; w++) {
  (new Writer(database)).start();
}
```

1. If we make both methods synchronized, does that solve the problem?

2. Is it a satisfactory solution? Explain your answer.

3. When does a reader have to wait until it can start reading?

4. In which method does a reader have to wait: **`read`** or **`write`**?

5. When does a writer have to wait until it can start writing?

6. In which method does a writer have to wait: **`read`** or **`write`**?

7. Of which *type* of information do we need to keep track so that we can determine
   - whether a writer is writing, and
   - whether a writer is writing or a reader is reading.

8. What are appropriate names for these two attributes?

9. ```
   public class Database {
     private boolean writing;
     private boolean reading;

     ...
   }
   ```

   Where and how are the attributes **`writing`** and **`reading`** initialized?

10. In

    ```
    public void read() {
    ...
    \\ read
    ...
    }
    ```

    how do we express that a thread has to wait if a writer is writing?

11. When invoking **`object.wait()`**, the current thread must own the lock (or monitor) of **`object`**. If that is not the case, a **`IllegalMonitorStateException`** is thrown.

    How can we ensure that the current thread owns the lock of the database when executing **`wait`** within the **`read`** method?

12. Where and how do we modify the value of the attribute **`writing`**?
    ```

13. In

```
public void write() {
...
\\ write
...
}
```

how do we express that a thread has to wait if a writer is writing or a reader is reading?

14. Where and how do we modify the value of the attribute **`reading`**?

15. We need more fine-grained information than a boolean that captures whether readers are reading. From this more fine-grained information we should be able to derive whether readers are reading.

What *type* of more fine-grained information is needed?

16. What is an appropriate name for this attribute?

17.
```
public class Database {
  private boolean writing;
  private int readers;
  ...
}
```

Where and how are the attributes **`writing`** and **`readers`** initialized?

18. In

```
public void write() {
  this.beginWrite();
  ...
}
```

how do we express that a thread has to wait if a writer is writing or a reader is reading?

19. Where and how do we modify the value of the attribute **`readers`**?