Concurrency

Franck van Breugel

March 13, 2018

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

5. 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.

6. What are appropriate names for these two attributes?

7.

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

...
}
```

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

8. In

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

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

9. The **`wait`** method throws an **`InterruptedException`** if any thread interrupted the current thread before or while the current thread was waiting for a notification.

   Since an **`InterruptedException`** is a checked exception, it needs to be specified or caught. How do we handle the **`InterruptedException`**?

10. 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?

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

12. 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?

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