**Task 5: React Water**

**Introduction**

Reentrant Locks are achieved through thread synchronization in Java using synchronized keywords. It uses methods to share resources. The methods they use are the lock and unlock methods. With Condition they are statements that allow two options between true and false. With an if-statement the statement is typically true and the else-statement is the false one.

Pseudocode - ReactWater.java

public class ReactWater{

/\*\*

\* Constructor of ReactWater

\*\*/

int H, O; // initialize variables

private final Lock l = new ReentrantLock();

private final Condition notEmpty = lock.newCondition();

public ReactWater() {

} // end of ReactWater()

/\*\*

\* When H element comes, if there already exist another H element

\* and an O element, then call the method of Makewater(). Or let

\* The H element waits in line.

\*\*/

public void hReady() throws InterruptedException {

if (H == 1 && O == 1){ then, Makewater(); }

else {

Call lock() constructor

try {

while(isEmpty()) { notEmpty.await()}

} finally{

Call unlock() constructor

}

}

} // end of hReady()

/\*\*

\* When O element comes, if there already exist another two H

\* elements, then call the method of Makewater(). Or let O element

\* wait in line.

\*\*/

public void oReady() {

if (H == 2){ then, Makewater(); }

else {

Call lock() constructor

try {

while(isEmpty()) {

notEmpty.await()}

}finally{

Call unlock() constructor

}

}

} // end of oReady()

/\*\*

\* Print out the message of "water was made!".

\*\*/

public void Makewater() {

printf(“water was made!");

} // end of Makewater(

} // end of class ReactWater

**Test Cases:**

1. Await(); for waiting the O and H elements
2. lock(); helps invoke the await method
3. unlock(); in case the await method is not empty, which then releases the threads.