CECS 277 – Lecture 20 – Thread Synchronization

When threads share resources, it may be necessary to sometimes lock out one of the threads in order to allow another thread to access the resource. In Java, we have the ability to synchronize blocks of code that access the shared resource in order to block threads.

There are three methods that are useful when synchronizing:

1. wait() – tells the thread to pause for a while and wait to be notified when the resource is free for them.
2. notify() – wakes up the first thread that was told to wait for this resource.
3. notifyAll() – wakes up all threads that were told to wait for this resource.

Example: Synchronized Cookie Producer / Consumer Program.

```
public class CookieJar {
    private int numCookies = 0;
    private int max = 3;
    public synchronized void add() throws InterruptedException{
        while(numCookies == max){
            System.out.println("Can't fit any cookies in the jar");
            wait();
        }
        numCookies++;
        notifyAll();
    }
    public synchronized int eat() throws InterruptedException{
        while(numCookies == 0){
            System.out.println("No cookies to eat...");
            wait();
        }
        notifyAll();
        return --numCookies;
    }
    public int getNum(){
        return numCookies;
    }
}
public class Baker implements Runnable{
    private CookieJar jar;
    public Baker(CookieJar j){
        jar = j;
    }
    public void run() {
        for (int i = 1;i<=12;i++){
            try{
                Thread.sleep((int)(Math.random()*5000));
                jar.add();
                System.out.println("The Baker made cookie " + i);
            }catch(InterruptedException e){
                System.out.println("Interrupted");
```

```
            }
        }
        System.out.println("The Baker baked a dozen cookies");
    }
}
public class Monster implements Runnable{
    private CookieJar jar;
    private int ate;
    public Monster(CookieJar j){
        jar = j;
        ate = 0;
    }
    public void run() {
        while(ate<12){
            try{
                jar.eat();
                System.out.println("jar.getNum()+ " cookies left.");
                ate++;
                Thread.sleep((int)(Math.random()*5000));
            }catch(InterruptedException e){
                System.out.println("Monster Interrupted");
            }
        }
        System.out.println("Monster ate "+ ate+" cookies.");
    }
}
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
public class Cookies {
    public static void main (String [] args){
        ExecutorService cookies =
        Executors.newCachedThreadPool();
        CookieJar jar = new CookieJar();
        cookies.execute(new Baker(jar));
        cookies.execute(new Monster(jar));
        cookies.shutdown();
    }
}
/*No cookies to eat...
The Baker made cookie 1
0 cookies left.
No cookies to eat...
The Baker made cookie 2
0 cookies left.
The Baker made cookie 3
0 cookies left.
The Baker made cookie 4
The Baker made cookie 5
1 cookies left.
0 cookies left...
```