CECS 277 – Lecture 18 – Threads

Threaded programs often need to share resources between the threads. A shared object can be created and then passed to each of the threads to access whenever it is needed.

Example: Sharing Resources.

```
public class CookieJar {
   private int numCookies = 0;
   public void setNum(int c){
      numCookies = c;
   }
   public int getNum(){
      return numCookies;
   }
}
public class Monster implements Runnable{
   private CookieJar jar;
   private int ate;
   public Monster(CookieJar j){
      jar = j;
      ate = 0;
   }
   @Override
   public void run() {
      while(ate<12){
         if (jar.getNum()>0){
            try{
               jar.setNum(jar.getNum()-1);
               System.out.println("Monster ate cookie,
               "+jar.getNum()+ " left.");
               ate++;
               Thread.sleep((int)(Math.random()*5000));
            }catch(InterruptedException e){
               System.out.println("Monster Interrupted");
            }
         }else{
            try{
               System.out.println("Me want cookie.");
               Thread.sleep((int)(Math.random()*2000));
            }catch(InterruptedException e){
               System.out.println("Monster Interrupted");
            }
         }
      }
      System.out.println(ate+" cookies were eaten.");
   }
}
```

```
public class Baker implements Runnable{
   private CookieJar jar;
   public Baker(CookieJar j){
      jar = j;
   }
   @Override
   public void run() {
      for (int i = 1;i<=12;i++){
         try{
         Thread.sleep((int)(Math.random()*5000));
            jar.setNum(jar.getNum()+1);
            System.out.println("Baker made cookie #"+ i);
         }catch(InterruptedException e){
            System.out.println("Interrupted");
         }
      }
      System.out.println("Baker finished baking 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();
   }
}
/*Me want cookie.
Baker made cookie #1
Monster ate cookie, 0 left.
Baker made cookie #2
Monster ate cookie, 0 left.
Me want cookie.
Baker made cookie #3
Monster ate cookie, 0 left.
Me want cookie.
Me want cookie.
Baker made cookie #4
Monster ate cookie, 0 left.
Baker made cookie #5
Monster ate cookie, 0 left.
Baker made cookie #6
Monster ate cookie, 0 left.
Baker made cookie #7
Baker made cookie #8
Baker made cookie #9
Monster ate cookie, 2 left.
Baker made cookie #10
Baker made cookie #11
Monster ate cookie, 3 left.
Baker made cookie #12
Baker finished baking
cookies.
Monster ate cookie, 3 left.
Monster ate cookie, 2 left.
Monster ate cookie, 1 left.
Monster ate cookie, 0 left.
12 cookies were eaten.
*/
```