Universidad de los Andes

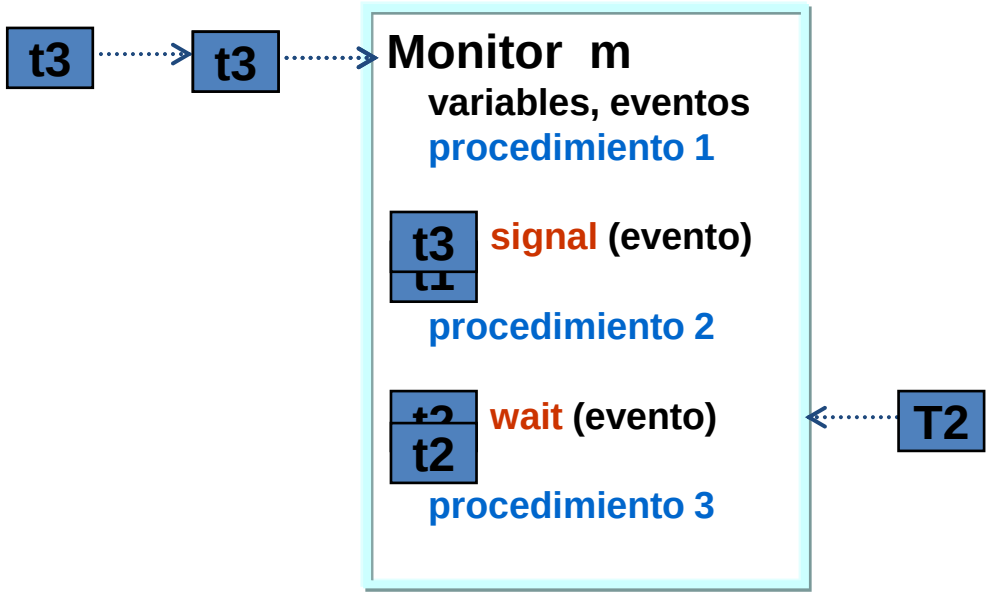
Infraestructura computacional

Concurrencia

Universidad de los Andes

Sincronización - eventos

- Monitores:



Universidad de los Andes

Sincronización - eventos

- Java: `wait`, `notify`, `notifyAll`
  - `wait( )`: el thread que lo invoca se suspende, libera al monitor, y se duerme esperando en el objeto en cuestión.
  - `notify( )`: despierta a uno de los threads dormidos en el objeto en cuestión.
  - `notifyAll( )`: despierta todos los threads que están dormidos esperando en el objeto.
  - Para que pueda usar estos métodos, un thread debe previamente apropiarse del objeto en cuestión (`synchronized`).
    - Nota: `wait( )` lanza excepciones, por lo cual debe ir en un bloque `try-catch` (o lanzar la excepción); en ocasiones, en los ejemplos lo obviamos para aligerar el código, pero en los programas reales es necesario tenerlo en cuenta.

Universidad de los Andes

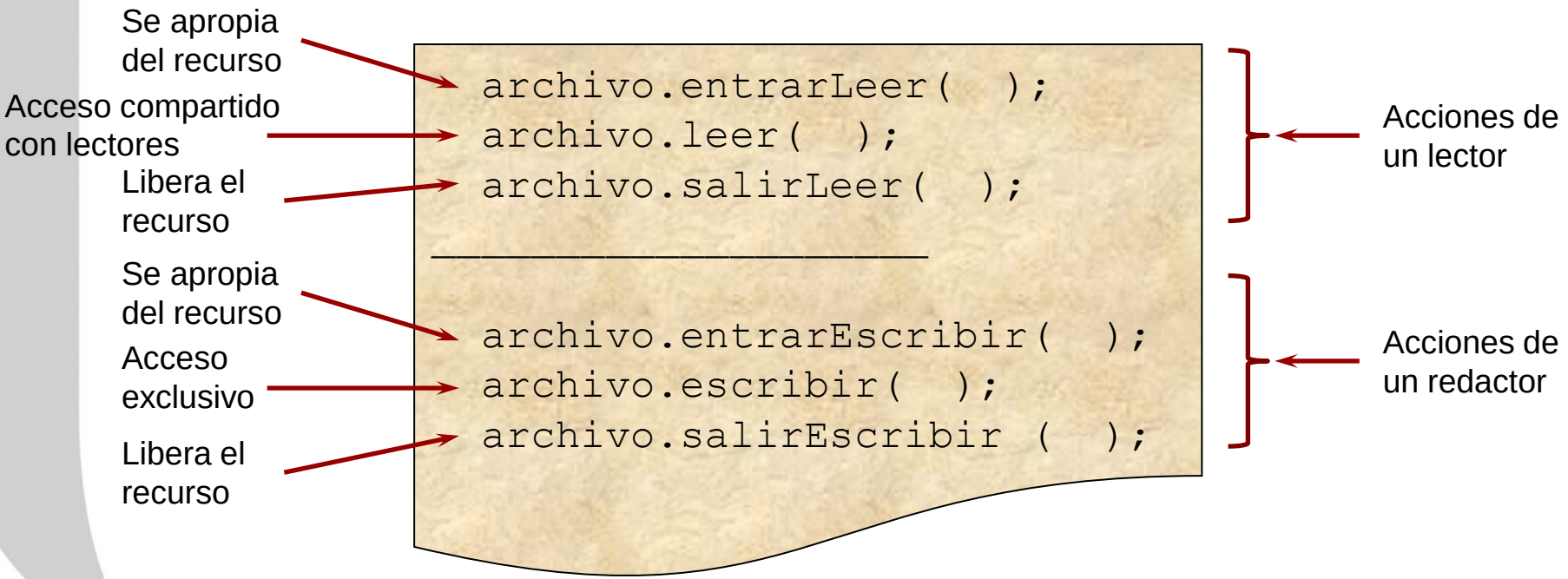
Sincronización - eventos

- Java – ejemplo: lectores – redactores
  - Se tiene un archivo y varios threads que acceden a él: unos en lecturas, otros en escritura
  - Si hay alguien escribiendo, ningún otro thread debe tener acceso al archivo
  - Si hay alguien leyendo, solo otros lectores pueden tener acceso
  - Cuando un proceso termina su operación, debe despertar a otros procesos que estén en espera (y que puedan continuar)
  - Para efectuar su operación, los threads solicitan permiso usando los métodos `entrarLeer( )` y `entrarEscribir( )`, según sea el caso
  - Al terminar, los threads informan esto usando los métodos `salirLeer( )` y `salirEscribir( )`

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: lectores - redactores



Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: lectores - redactores

```
public class Archivo {
  private int nRedactores = 0;
  private int nLectores = 0;

  public synchronized void entrarLeer(  ) {
    while( nRedactores != 0 ) wait(); //Conceptual
    nLectores++;
  }
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: lectores - redactores

```
public class Archivo {
  private int nRedactores = 0;
  private int nLectores = 0;

  public synchronized void entrarLeer(  ) {
    while ( nRedactores != 0 ) { //Implantación
      try { wait(); }
        catch ( InterruptedException e ) { }
    }
    nLectores++;
  }
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: lectores - redactores

```
public synchronized void entrarEscribir(  ) {
  while ( nRedactores != 0 || nLectores != 0 ) {
    try { wait(); }
      catch ( InterruptedException e ) { }
  }
  nRedactores++;
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: lectores - redactores

```
public synchronized void salirLeer(  ) {
  nLectores--;
  if ( nLectores == 0 ) notify();
}

public synchronized void salirEscribir(  ) {
  nRedactores--;
  notifyAll();
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejercicio. Lectores – redactores debilitado :
  - Resuelva el problema de los lectores-redactores con el siguiente ajuste:
    Los lectores pueden estar en compañía de otros lectores o de un redactor.

Universidad de los Andes

Sincronización - eventos

- Java – solución lectores – redactores debilitado:

```
public synchronized void entrarLeer(  ) {
}

public synchronized void entrarEscribir(  ) {
  while ( nRedactores != 0 ) {
    try { wait(); }
      catch ( InterruptedException e ) { }
  }
  nRedactores ++;
}
```

Universidad de los Andes

Sincronización - eventos

- Java – solución lectores – redactores debilitado:

```
public synchronized void salirLeer(  ) {
}

public synchronized void salirEscribir(  ) {
  nRedactores--;
  notify();
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejercicio. Lectores – redactores sin inanición:
  - Si están llegando continuamente lectores, los redactores no pueden entrar nunca.
  - Modificar el programa anterior para que no se presente esa situación; en lugar de esto, cuando un redactor se encuentra en espera, no se permite la entrada de más lectores (quedan en espera de un turno).

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor– consumidor
  - Un thread genera datos (el productor), otro los recibe y procesa (el consumidor).
  - Entre los dos hay un buffer de un tamaño limitado.
  - El buffer sirve para almacenar temporalmente datos cuando el consumidor no los puede procesar.
  - De esta manera se evita que el productor se bloquee.
    - Nota: este ejemplo pretende ilustrar el uso de objetos como eventos, pero se puede hacer sin ellos (usando al propio buffer para hacer los `wait`).

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public class Buffer {
  private ArrayList buff;
  private int n;
  Object lleno, vacío;

  public Buffer ( int n ) {
    this.n = n;
    buff = new ArrayList( );
    lleno = new Object();
    vacío = new Object();
  }
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public static void main( String[] args ) {

  Buffer buffer = new Buffer( 5 );

  Productor p = new Productor( 20, buffer );
  Consumidor c = new Consumidor( 20, buffer );

  p.start();
  c.start();
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public void almacenar ( Integer i ){

  if( buff.size( ) == n ) lleno.wait( );   //Conceptual

  synchronized( this ){ buff.add( i ); }
  vacío.notify( );
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public Integer retirar(){

  if( buff.size( ) == 0 ) vacío.wait( );    //Conceptual

  Integer i;
  synchronized( this ){ i = (Integer)buff.remove(0); }
  lleno.notify( );
  return i;
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public void almacenar ( Integer i ){

  while ( buff.size( ) == n ){    //Implementación
    synchronized( lleno ){
      try { lleno.wait( ); }
        catch( InterruptedException e ){}
    }
  }

  synchronized( this ){ buff.add( i ); }
  synchronized( vacío ){ vacío.notify(); }
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public void almacenar ( Integer i ){

  synchronized( lleno ){
    while ( buff.size( ) == n ){    //Corrección
      try { lleno.wait( ); }
        catch( InterruptedException e ){}
    }
  }

  synchronized( this ){ buff.add( i ); }
  synchronized( vacío ){ vacío.notify(); }
}
```

Universidad de los Andes

Sincronización - eventos

- Java – ejemplo: productor- consumidor

```
public Integer retirar (){

  synchronized( vacío ){
    while ( buff.size( ) == 0 ){    //Implementación
      try { vacío.wait( ); }
        catch( InterruptedException e ){}
    }
  }
  Integer i;
  synchronized( this ){ i = (Integer)buff.remove(0); }
  synchronized( lleno ){ lleno.notify( ); }
  return i;
}
```

Universidad de los Andes

Sincronización - encuentros

- Java: join

```
for ( i = 0; i < nThreads; i++ ) {
  t[i] = new T( );
  t[i].start( );
}
  …
for ( i = 0; i < nThreads; i++ ) {
  t[i].join( );
}
```