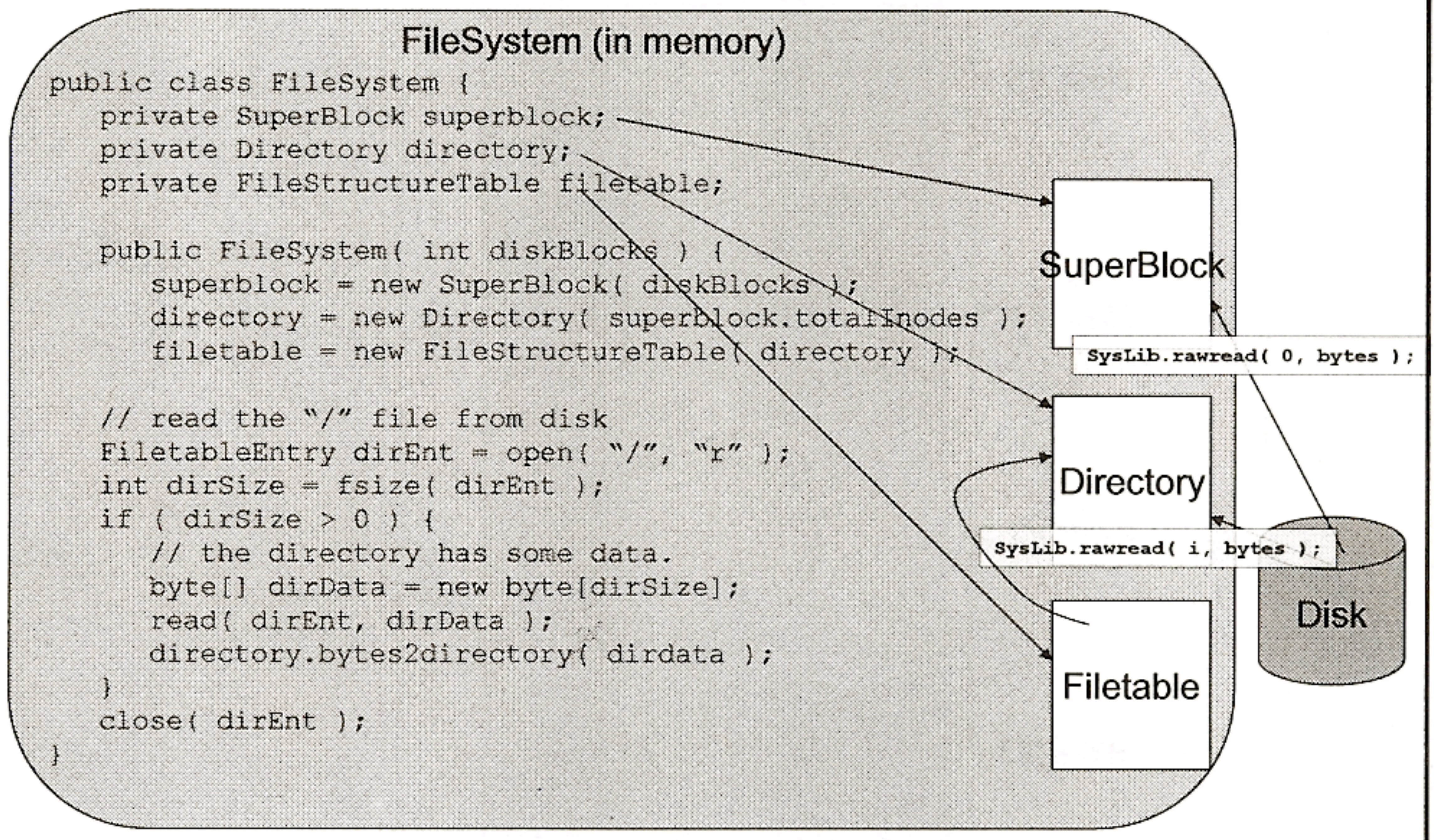
CSS430 Final Project

File System

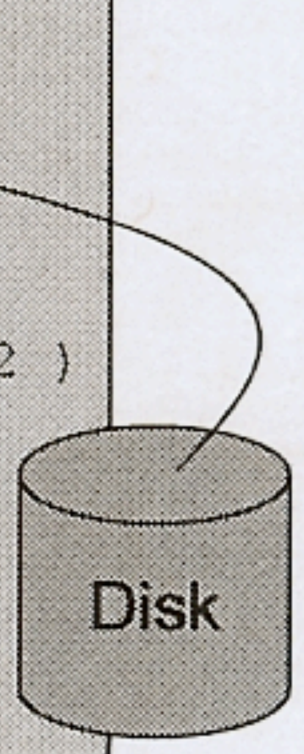
File System Constructor



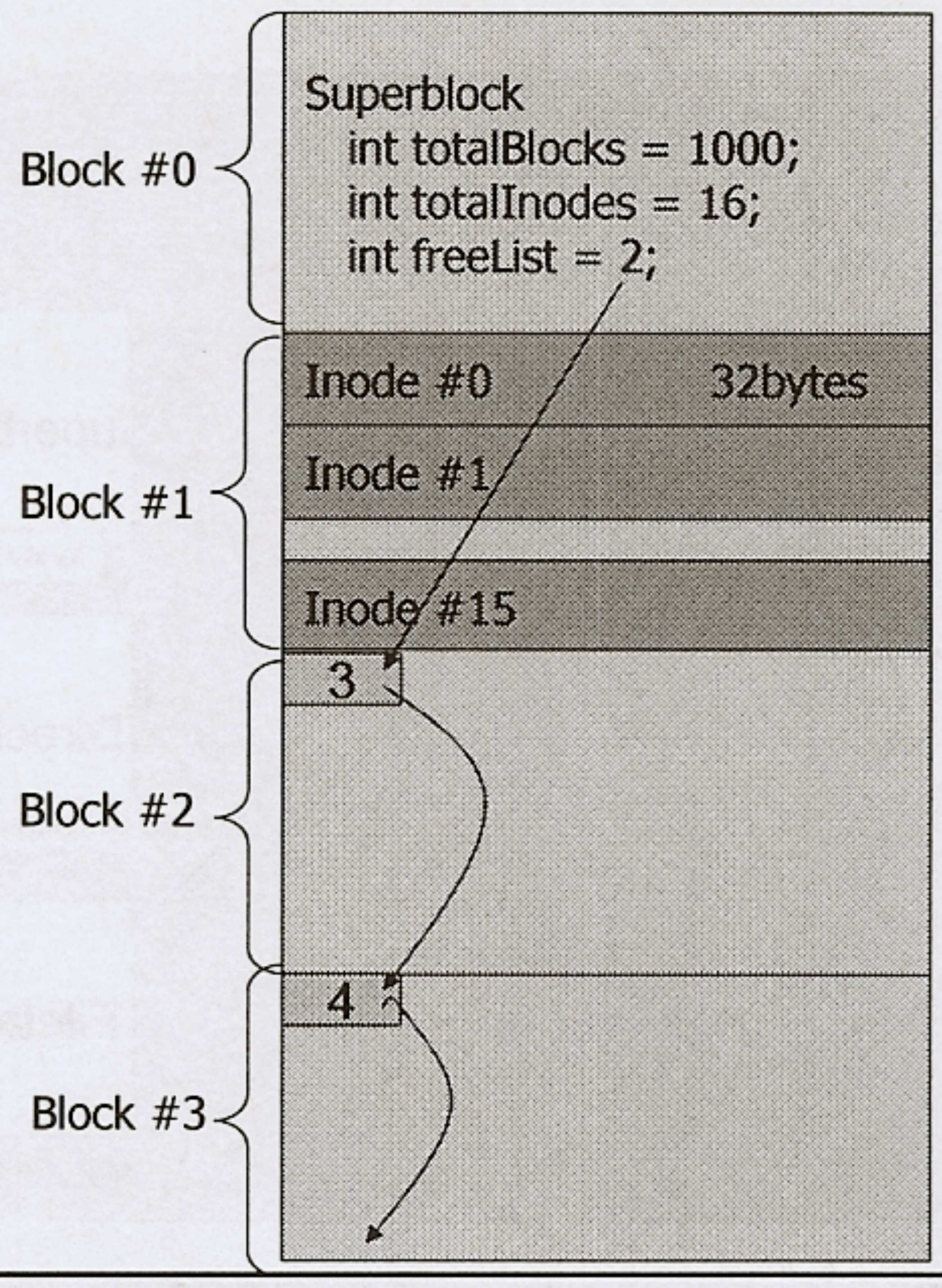
```
public class FileSystem {
   private SuperBlock superblock;
   private Directory directory;
   private FileStructureTable filetable;

   public FileSystem( int diskBlocks ) {
      superblock = new SuperBlock( diskBlocks );
      directory = new Directory( superblock.totalInodes );
      filetable = new FileStructureTable( directory );

   // read the "/" file from disk
   FiletableEntry dirEnt = open( "/", "r" );
   int dirSize = fsize( dirEnt );
   if ( dirSize > 0 ) {
      // the directory has some data.
      byte[] dirData = new byte[dirSize];
      read( dirEnt, dirData );
      directory.bytes2directory( dirdata );
   }
   close( dirEnt );
}
```

SuperBlock Constructor

```
Public class SuperBlock {
   private final int defaultInodeBlocks = 64;
   public int totalBlocks;
   public int totalInodes;
   public int freeList;

   public SuperBlock( int diskSize ) {
      // read the superblock from disk
      byte[] superBlock = new byte[Disk.blockSize];
      SysLib.rawread( 0, superBlock );
      totalBlocks = SysLib.bytes2int( superBlock, 0 );
      totalInodes = SysLib.bytes2int( superBlock, 4 );
      freeList = SysLib.bytes2int( superBlock, 8 );

      if ( totalBlocks == diskSize && totalInodes > 0 && freeList >= 2 )
         // disk contents are valid
         return;
      else {
         // need to format disk
         totalBlocks = diskSize;
         format( defaultInodeBlocks );
      }
   }
}
```



Format(16)



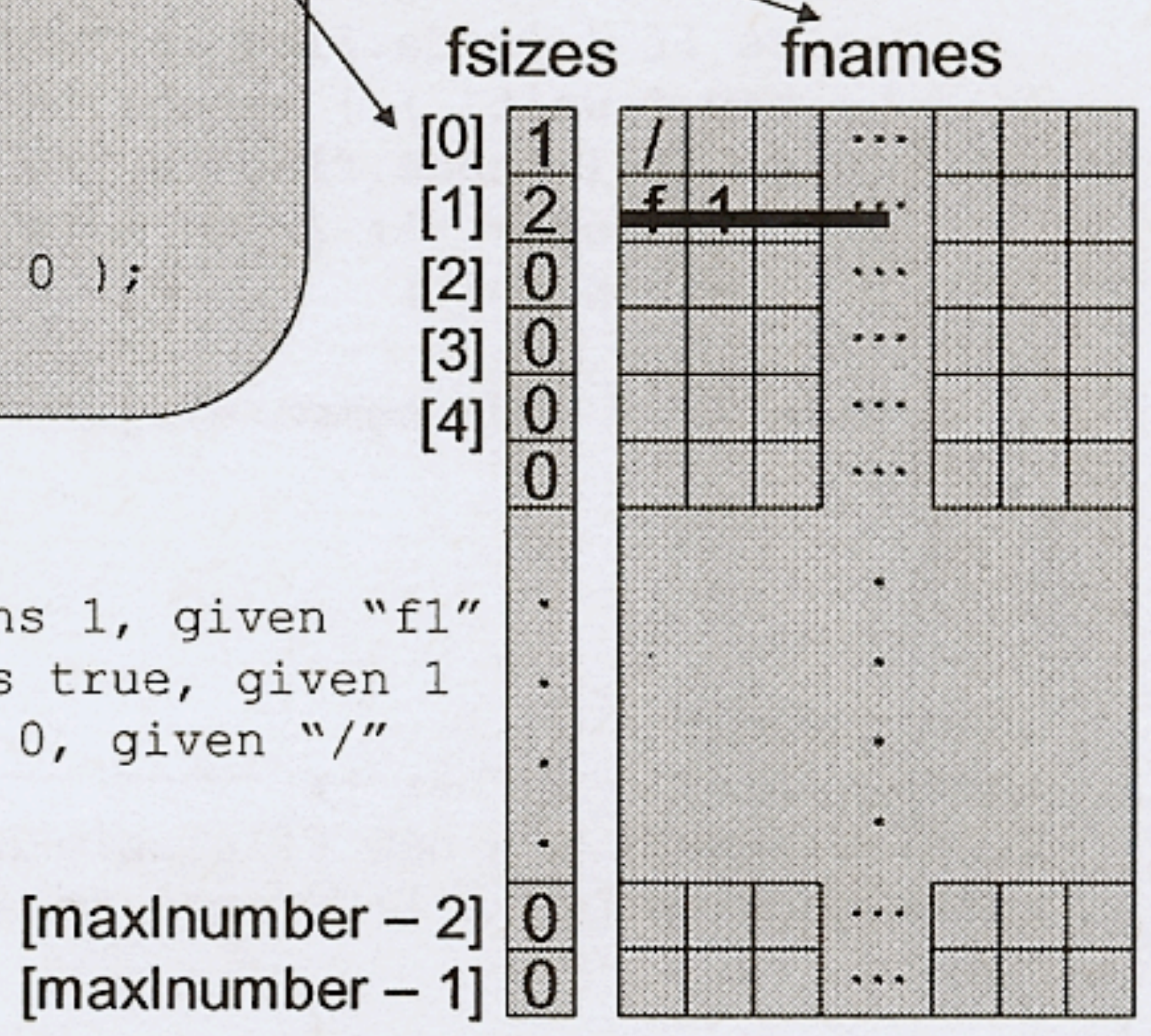
Other SuperBlock Methods

- sync( )
  - Write back totalBlocks, inodeBlocks, and freeList to disk.
- getFreeBlock( )
  - Dequeue the top block from the free list.
- returnBlock(int blockNumber)
  - Enqueue a given block to the end of the free list.

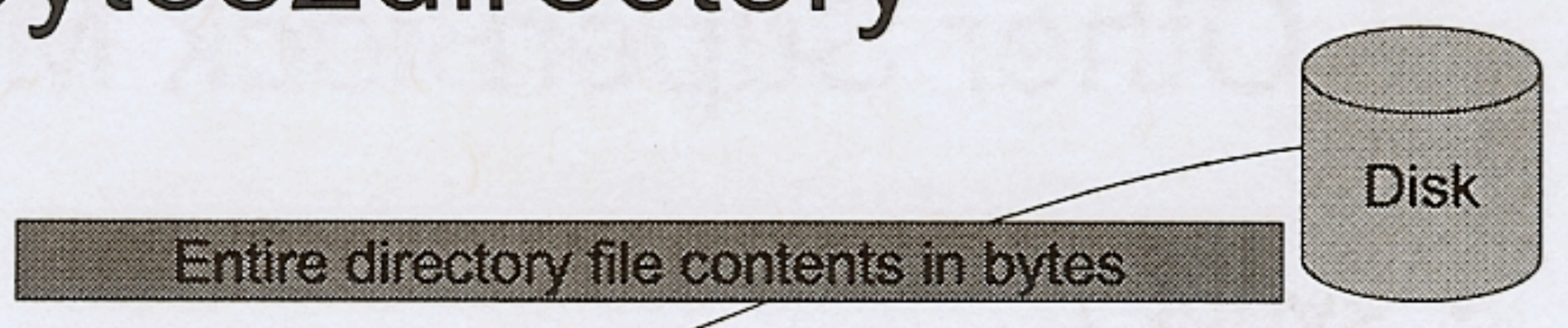
Directory

```
public class Directory {
   private static in maxChars = 30;
   private int fsizes[];      // the file name's length
   private char fnames[][];  // file names

   public Directory( int maxInumber ) {
      fsizes = new int[maxInumber];
      for ( int I = 0; I < maxInumber; i++ )
         fsizes[i] = 0;

      String root = "/";
      fsizes[0] = root.length( );
      root.getChars( 0, fiszes[0], fnames[0], 0 );
}
```

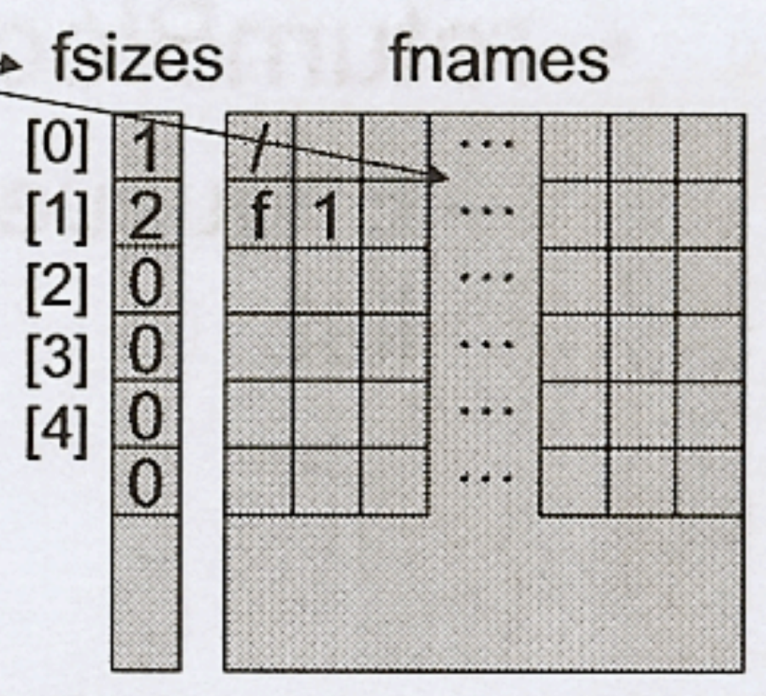
```
public short ialloc( String filename ) returns 1, given "f1"
public boolean ifree( short iNumber ) returns true, given 1
public short namei( String filename )returns 0, given "/"
```



bytes2directory

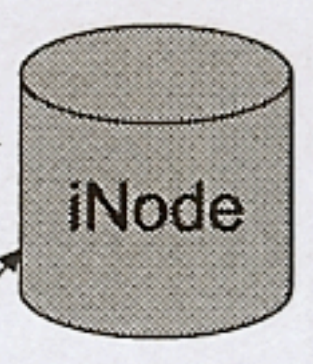


```
public void bytes2directory( byte data[] ) {
   int offset = 0;
   for ( int I = 0; I < fsizes.length; i++ offset += 4 )
      fsizes[i] = SysLib.bytes2int( data, offset );

   for ( int I = 0; I < fnames.length; i++, offset += maxChars * 2 ) {
      String fname = new String( data, offset, maxChars * 2 );
      fname.getChars( 0, fsizes[i], fnames[i], 0 );
   }
}
```



File Structure Table

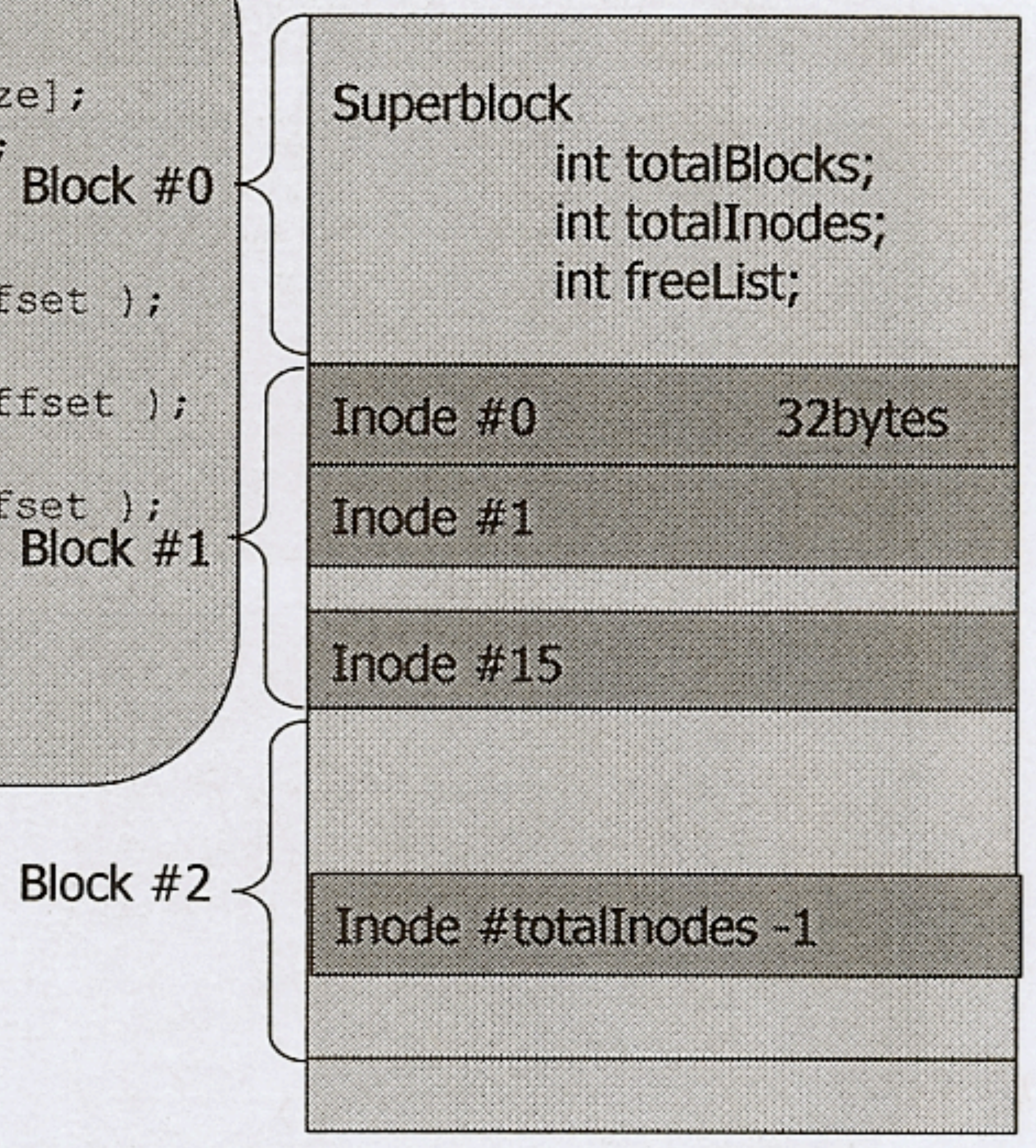
```
public synchronized FiletableEntry falloc( String fname, String mode ) {

   short iNumber = -1;
   Inode inode = null;

   while ( true ) {
      iNumber = ( fnames.equals( "/" ) ? 0 : dir.namei( fname );
      if ( iNumber >= 0 ) {
      inode = new Inode( iNumber );
      if ( mode.compareTo( "r" ) ) {
         if ( inode.flag is "read" ) break; // no need to wait
         else if ( inode.flag is "write" ) { // wait for a write to exit
           try { wait( ) } catch( InterruptedException e ) {
         else if ( inode.flag is "to be deleted" ) {
            iNumber = -1; // no more open
            return null;
         }
      } else if ( mode.compareTo( "w" ) ) {
         . . .
      }
   }

   inode.count++;
   inode.toDisk( iNumber );
   FileTableEntry e = new FileTableEntry( inode, iNumber, mode );
   table.addElement( e ); // create a table entry and register it.
   return e;
}
```



Save iNode to disk every update

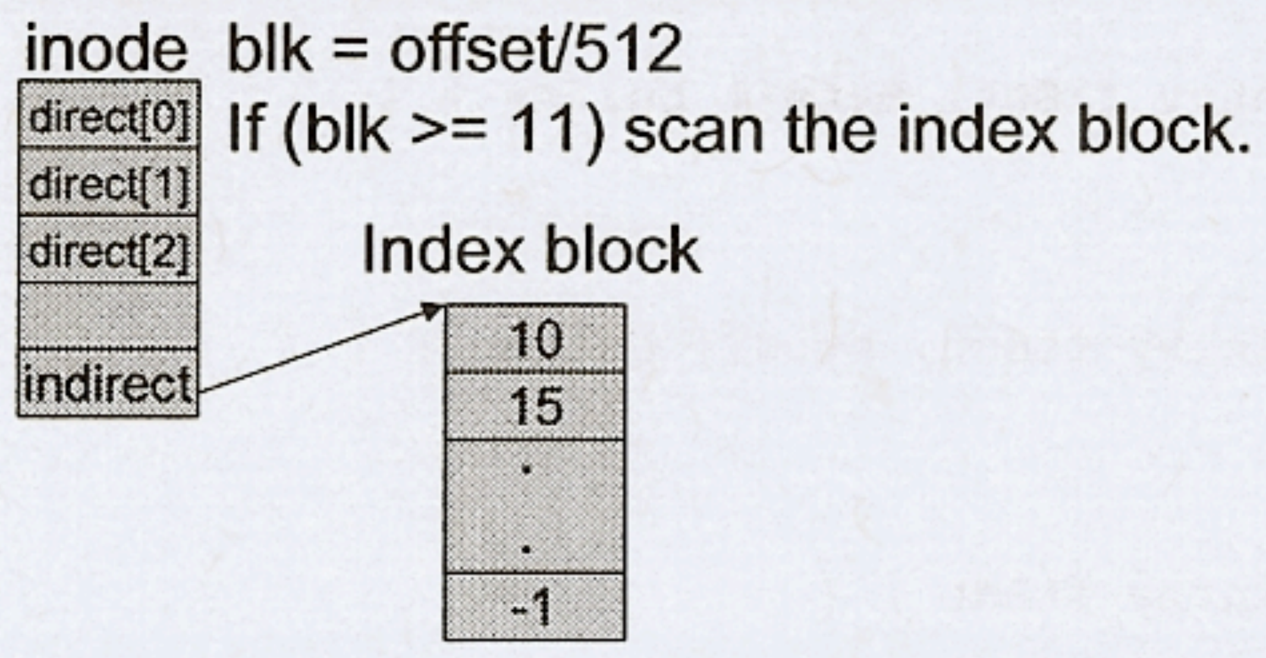
Inode Constructor

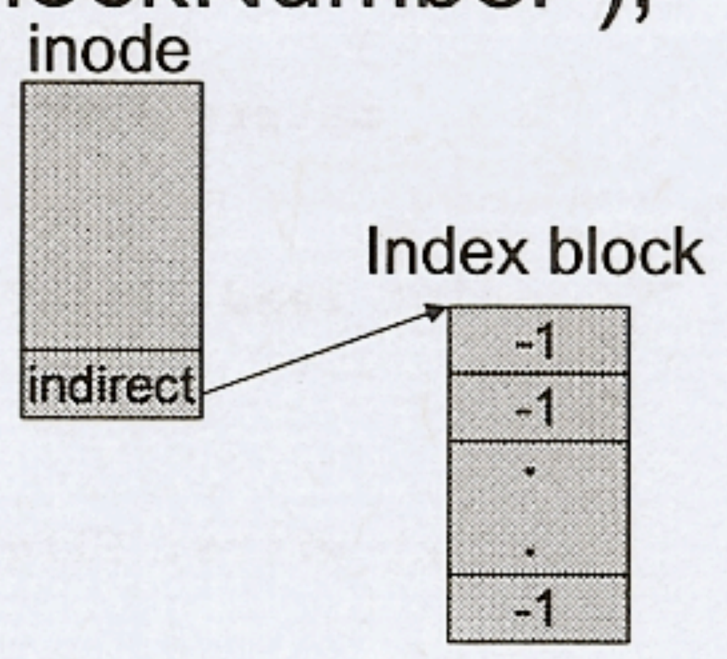
```
public class Inode {
   Inode( short iNumber ) {
      int blockNumber = 1 + iNumber / 16;
      byte[] data = new byte[Disk.blockSize];
      SysLib.rawread( blockNumber, data );
      int offset = ( iNumber % 16 ) * 32;

      length = SysLib.bytes2int( data, offset );
      offset += 4;
      count = SysLib.bytes2short( data, offset );
      offset += 2;
      flag = SysLib.bytes2short( data, offset );
      offset += 2;

      ...;
   }
}
```

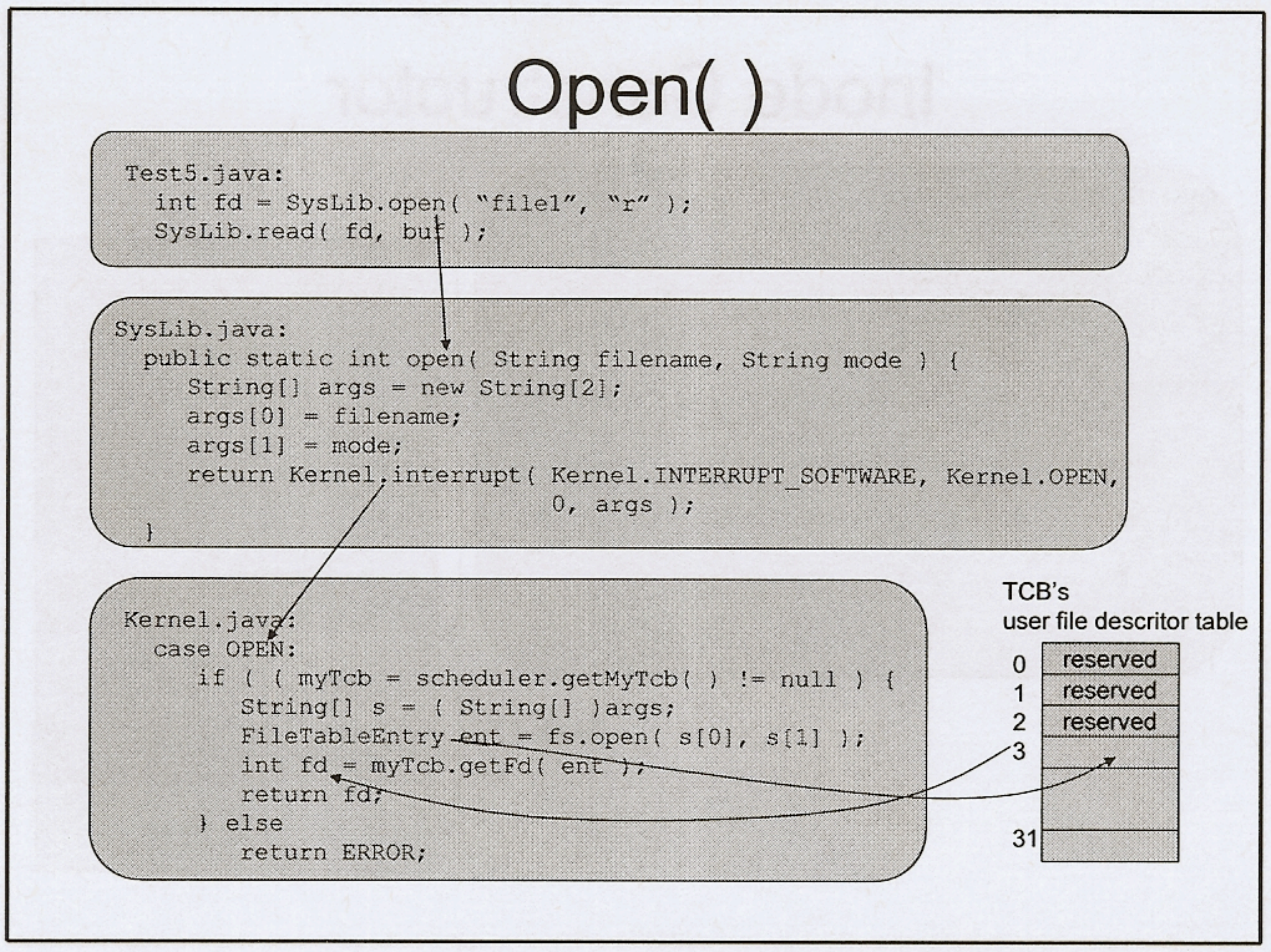


Other Inode Functions

- void toDisk(short iNumber);
- short getIndexBlockNumber( );
- Boolean setIndexBlock(short indexBlockNumber);
- Short findTargetBlock(int offset);
- Etc.







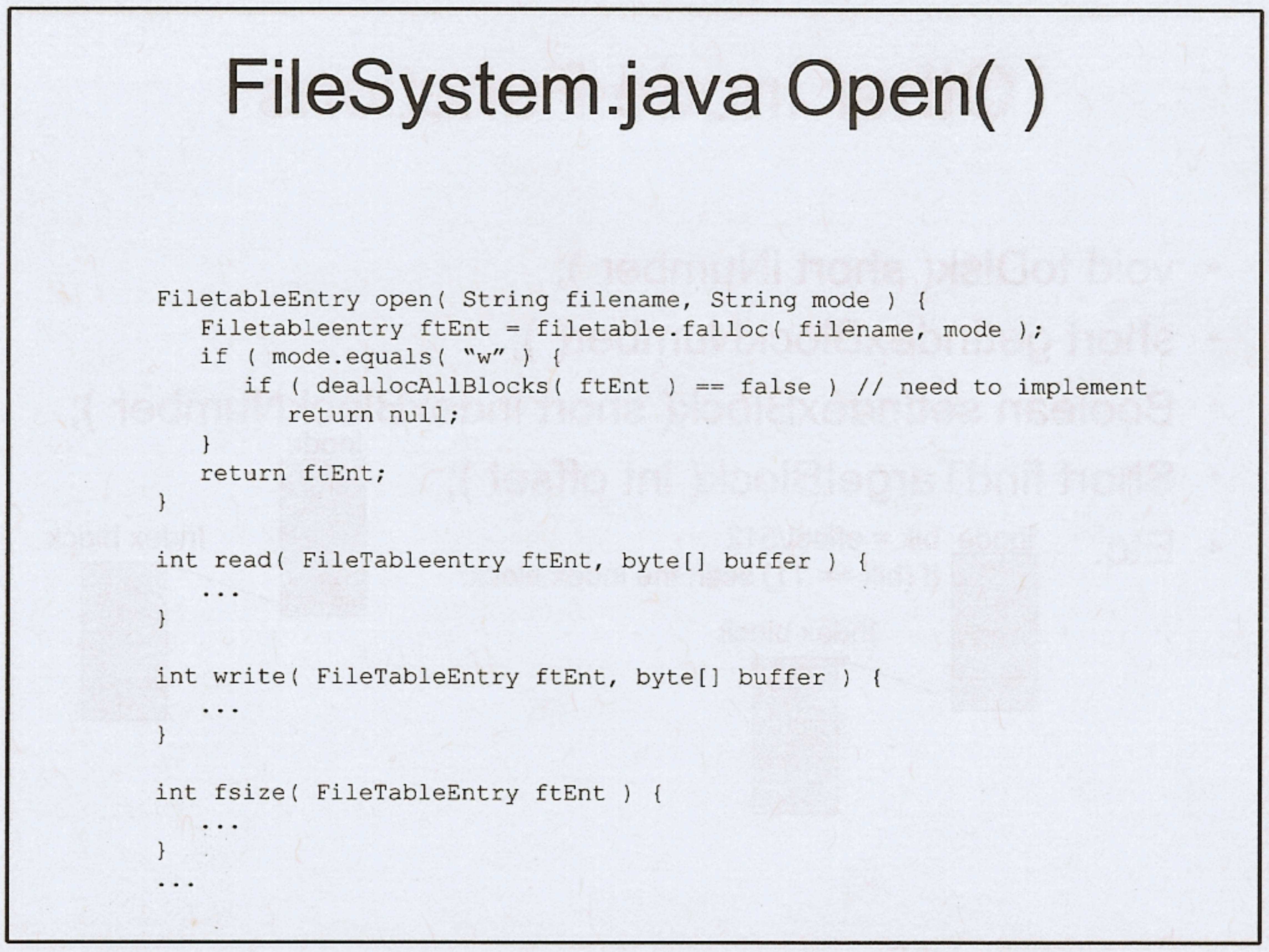
Open( )

```
Test5.java:
  int fd = SysLib.open( "file1", "r" );
  SysLib.read( fd, buf );
```

```
SysLib.java:
  public static int open( String filename, String mode ) {
     String[] args = new String[2];
     args[0] = filename;
     args[1] = mode;
     return Kernel.interrupt( Kernel.INTERRUPT_SOFTWARE, Kernel.OPEN,
                              0, args );
  }
```

```
Kernel.java:
  case OPEN:
     if ( ( myTcb = scheduler.getMyTcb( ) != null ) {
        String[] s = ( String[] )args;
        FileTableEntry ent = fs.open( s[0], s[1] );
        int fd = myTcb.getFd( ent );
        return fd;
     } else
        return ERROR;
```

TCB's
user file descritor table
0
reserved
1
reserved
2
reserved
3
31



FileSystem.java Open( )

```
FiletableEntry open( String filename, String mode ) {
   Filetableentry ftEnt = filetable.falloc( filename, mode );
   if ( mode.equals( "w" ) ) {
      if ( deallocAllBlocks( ftEnt ) == false ) // need to implement
         return null;
   }
   return ftEnt;
}

int read( FileTableentry ftEnt, byte[] buffer ) {
   ...
}

int write( FileTableEntry ftEnt, byte[] buffer ) {
   ...
}

int fsize( FileTableEntry ftEnt ) {
   ...
}
...
```