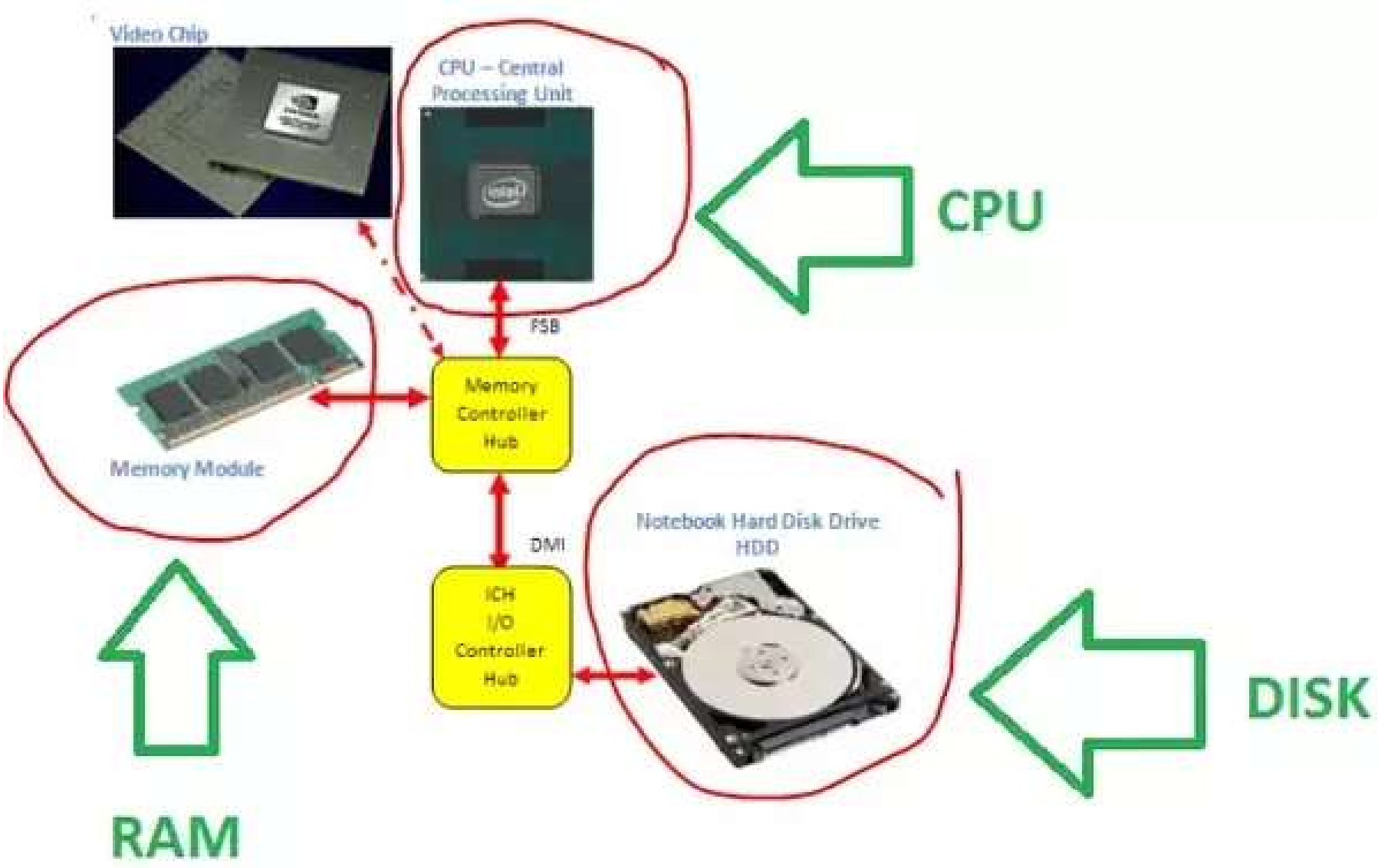
COMP 110/L Lecture 23

Mahdi Ebrahimi

Some slides adapted from Dr. Kyle Dewey

Outline

- Reading from files
- Writing to files


Video Chip
CPU – Central Processing Unit
CPU
FSB
Memory Controller Hub
Memory Module
DMI
ICH I/O Controller Hub
Notebook Hard Disk Drive HDD
DISK
RAM

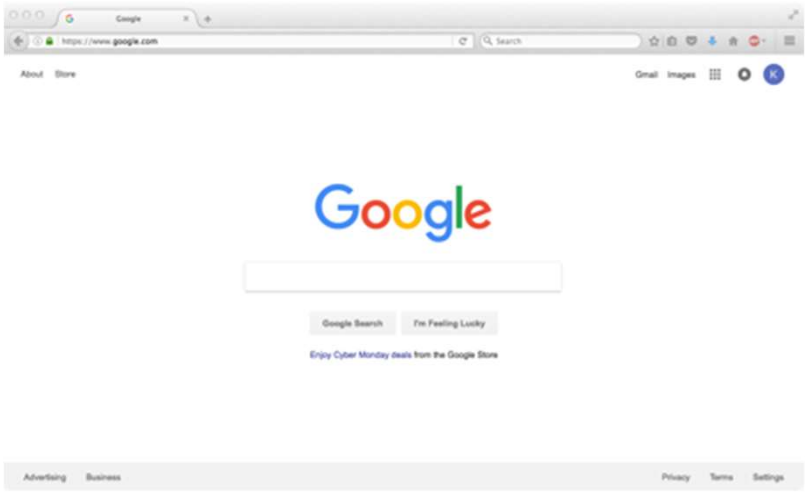
Reading From Files

Motivation

Files act like very large inputs; basis for most things.

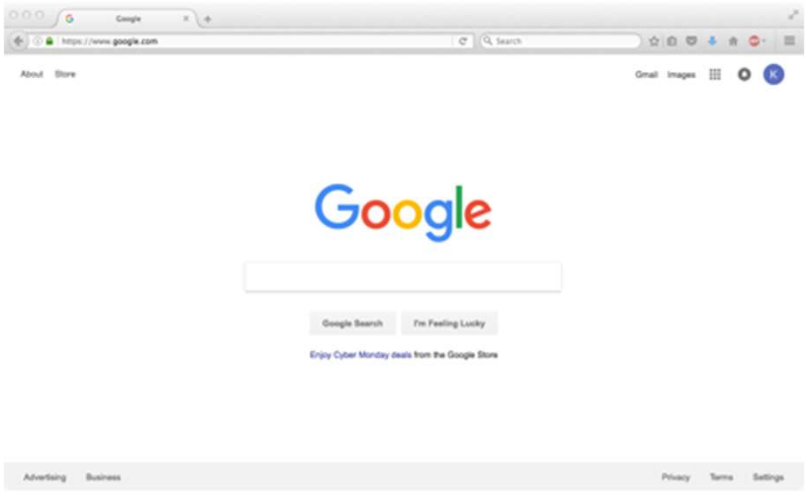
Motivation

Files act like very large inputs; basis for most things.



Motivation

Files act like very large inputs; basis for most things.




```
public class MyClass {
   ...
}
```

Reading from Files

Reading from Files

myFile.txt

myFile.txt

Contents

```
one
two
three
```

Reading from Files

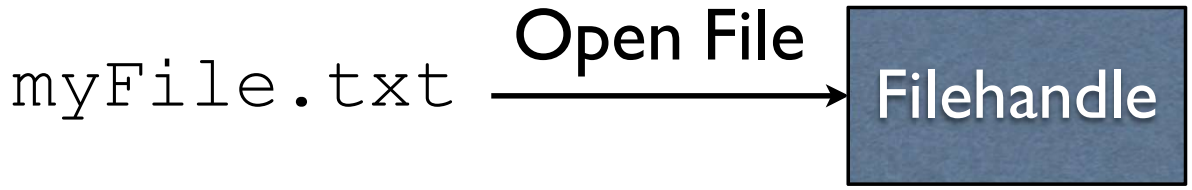
`myFile.txt` Open File 

`myFile.txt`

Contents

```
one
two
three
```

Reading from Files

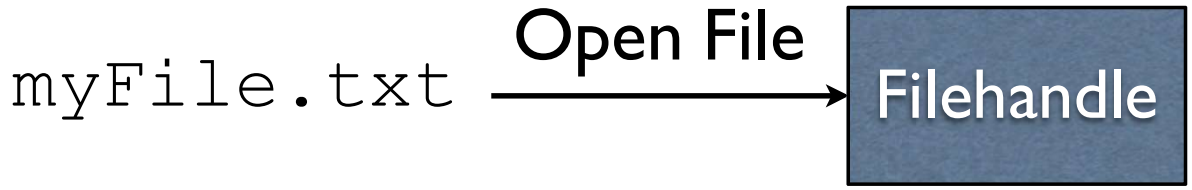


myFile.txt

Contents

```
one
two
three
```

Reading from Files

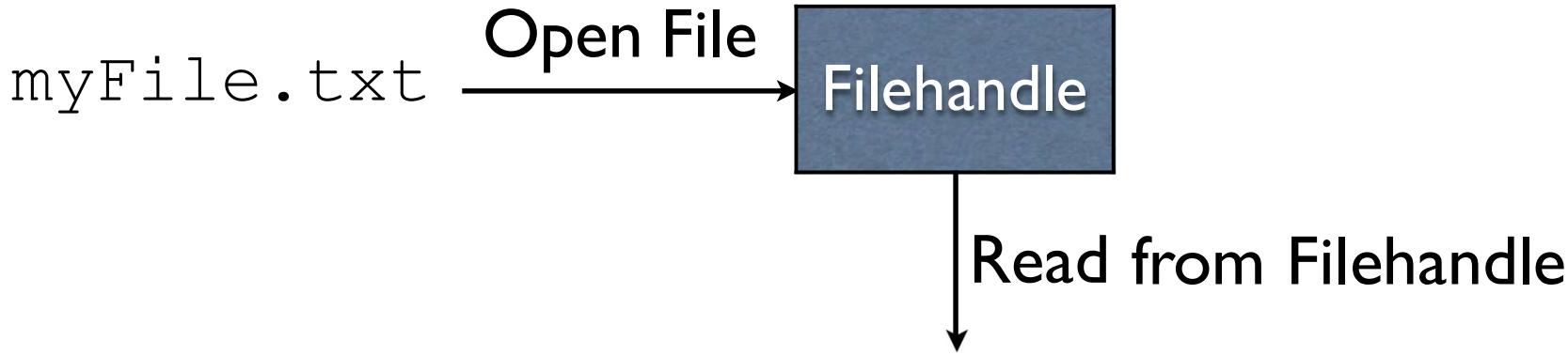


myFile.txt

Contents

```
one
two
three
```

Reading from Files

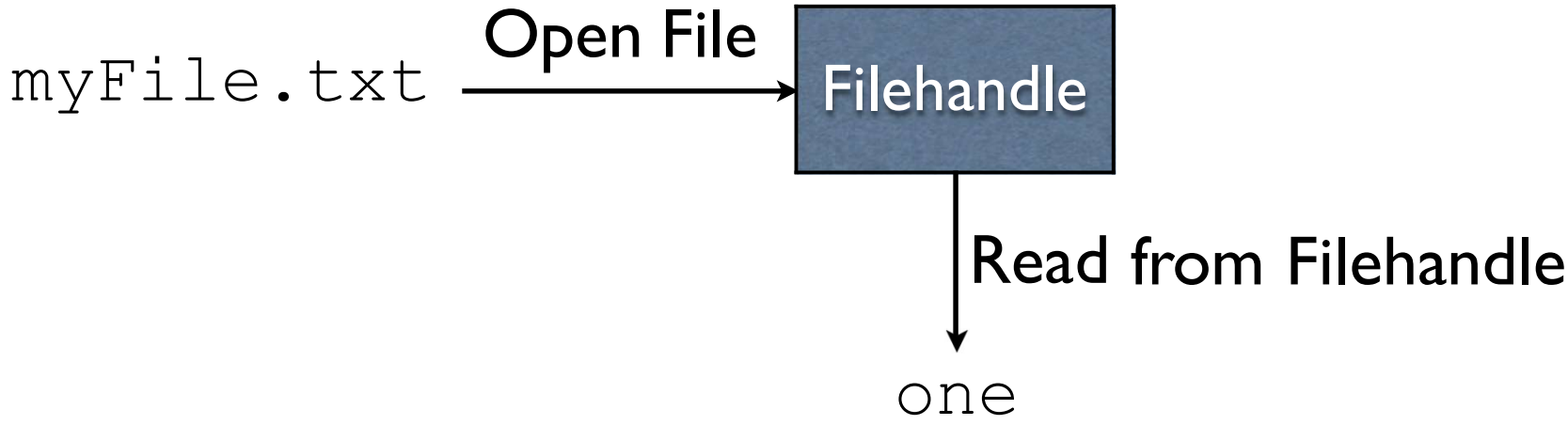


myFile.txt

Contents

```
one
two
three
```

Reading from Files

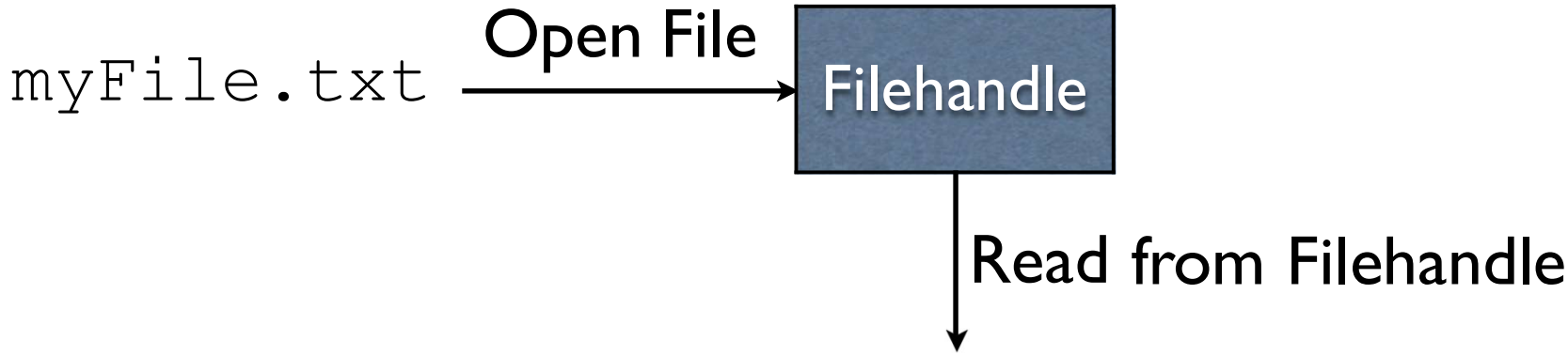


myFile.txt

Contents

```
one
two
three
```

Reading from Files

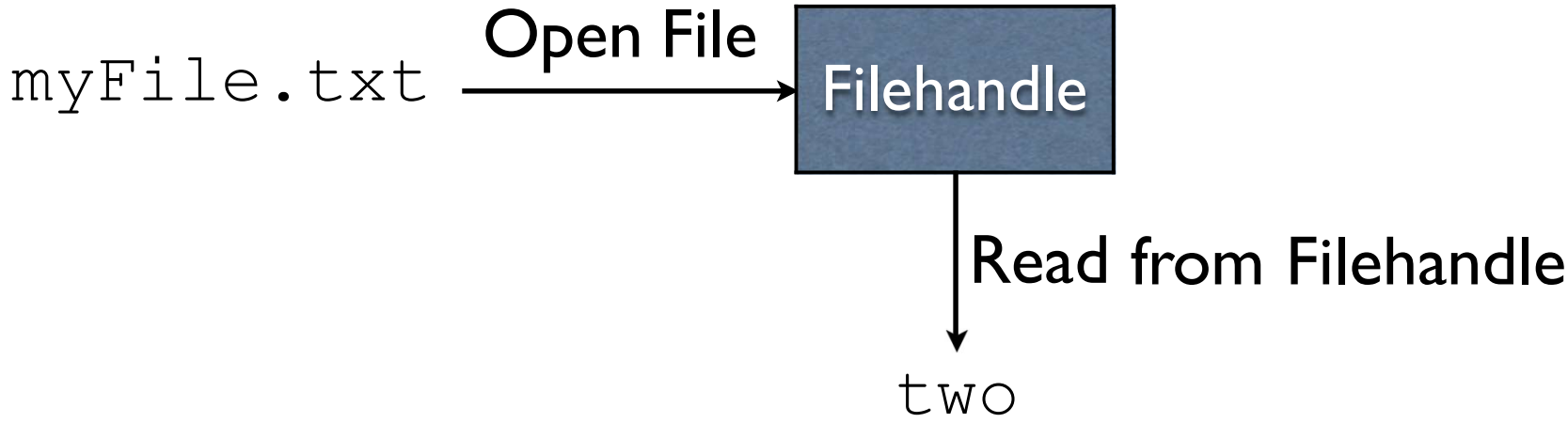


myFile.txt

Contents

```
one
two
three
```

Reading from Files

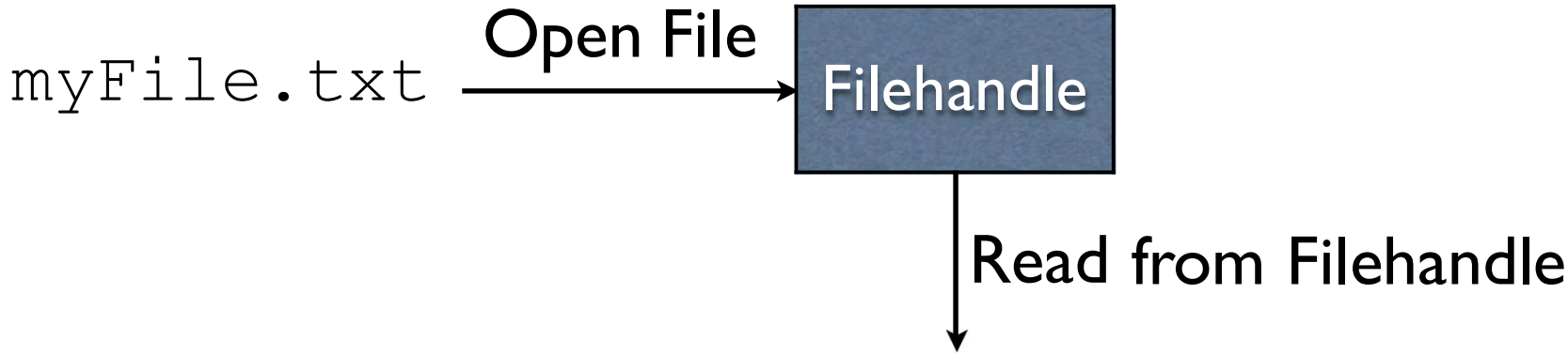


myFile.txt

Contents

```
one
two
three
```

Reading from Files

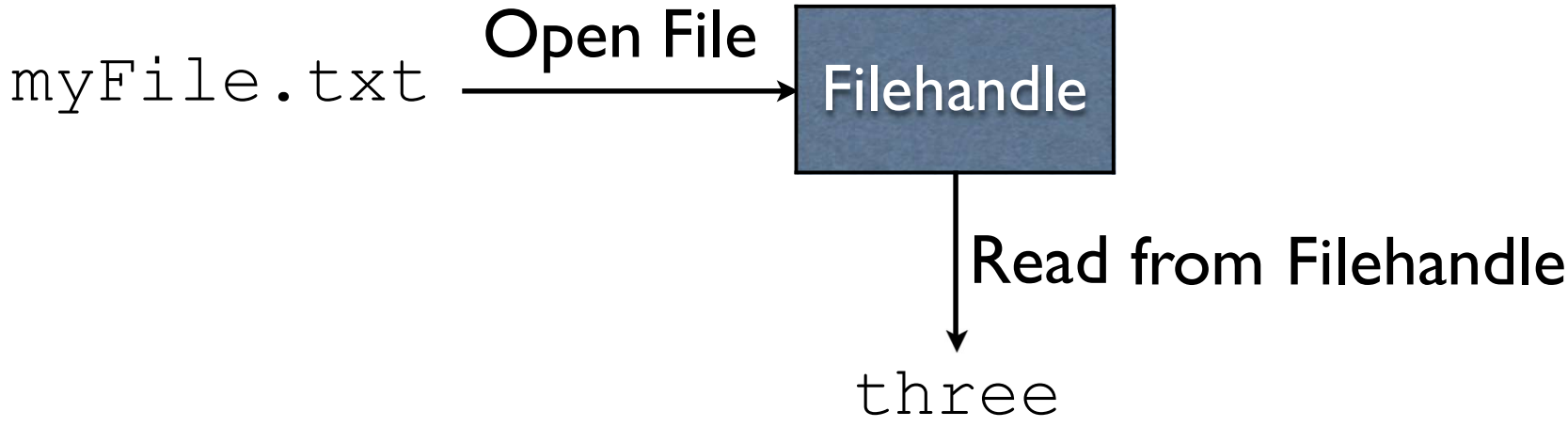


myFile.txt

Contents

```
one
two
three
```

Reading from Files

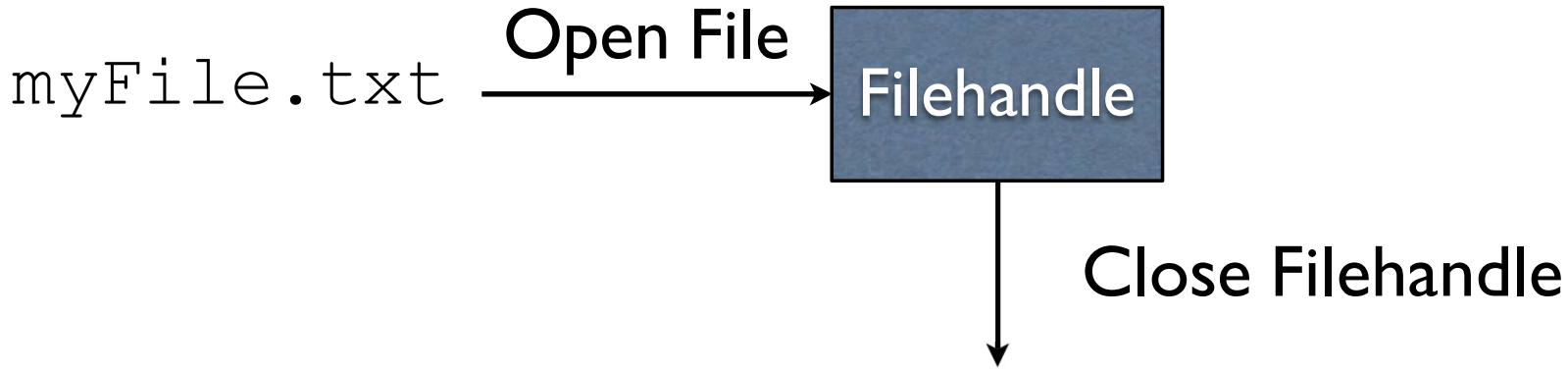


myFile.txt

Contents

```
one
two
three
```

Reading from Files

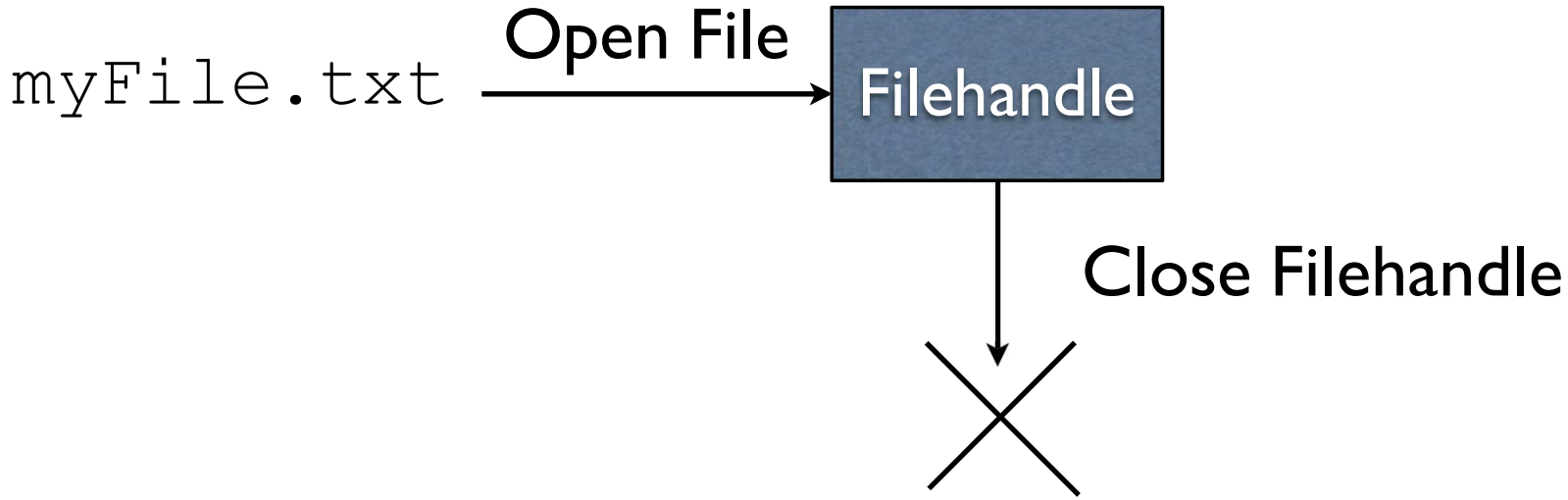


myFile.txt

Contents

```
one
two
three
```

Reading from Files



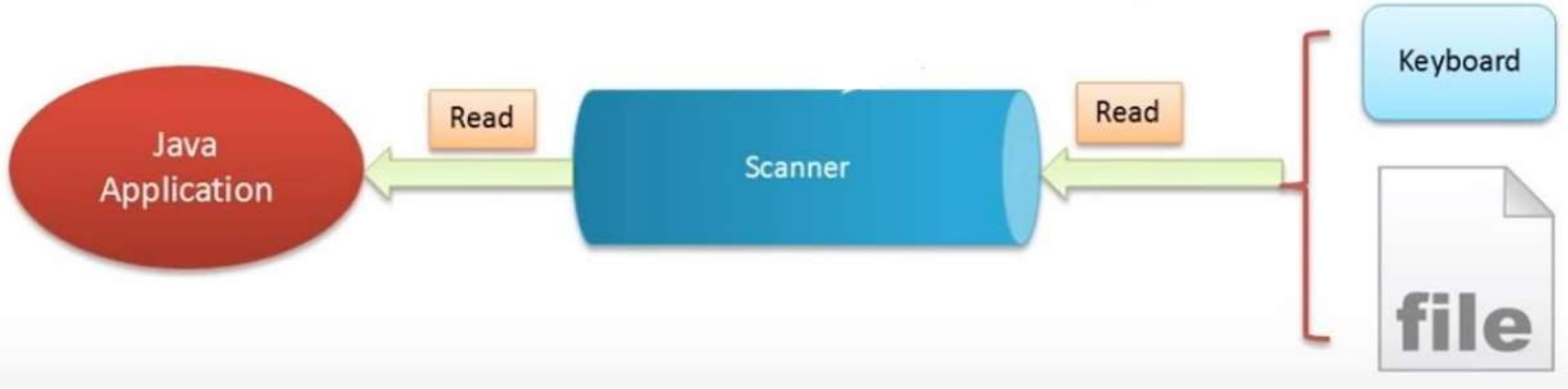
myFile.txt

Contents

```
one
two
three
```

Reading from Files with `Scanner`

Reading from Files with `Scanner`



Reading from Files with `Scanner`

Step 1: Create `File`

Reading from Files with `Scanner`

Step 1: Create `File`

```
File myFile = new File("myFile.txt");
```

Reading from Files with `Scanner`

Step 1: Create `File`

```
File myFile = new File("myFile.txt");
```

Step 2: Create `Scanner` with the `File`

Reading from Files with `Scanner`

Step 1: Create `File`

```
File myFile = new File("myFile.txt");
```

Step 2: Create `Scanner` object with the `File`

```
Scanner input = new Scanner(myFile);
```

Reading from Files with `Scanner`

Step 1: Create `File`

```
File myFile = new File("myFile.txt");
```

Step 2: Create `Scanner` object with the `File`

```
Scanner input = new Scanner(myFile);
```

Step 3: Read from `Scanner`

Reading from Files with `Scanner`

Step 1: Create `File`

```
File myFile = new File("myFile.txt");
```

Step 2: Create `Scanner` object with the `File`

```
Scanner input = new Scanner(myFile);
```

Step 3: Read from `Scanner`

```
if (input.hasNextLine()) {
  String line = input.nextLine();
  ...
```

Reading from Files with `Scanner`

Step 4: Close `Scanner`

Reading from Files with `Scanner`

Step 4: Close `Scanner`

```
input.close();
```

Example:
`ReadFirstLine.java`

Example:
`ReadWholeFile.java`

`FileNotFoundException`

`Scanner` **will throw a**
`FileNotFoundException` **if the file does not exist.**

FileNotFoundException

`Scanner` **will throw a**
`FileNotFoundException` **if the file does not exist.**

Example:
`ReadWholeFileWithTry.java`

Writing to Files

Writing to Files

Writing to Files

Step 1: Create a `File`

```
File myFile = new File("myFile.txt");
```

Writing to Files

Step 1: Create a `File`

```
File myFile = new File("myFile.txt");
```

Step 2: Create a `FileWriter`

```
FileWriter fw = new FileWriter(myFile);
```

Writing to Files

Step 1: Create a `File`

```
File myFile = new File("myFile.txt");
```

Step 2: Create a `FileWriter`

```
FileWriter fw = new FileWriter(myFile);
```

Step 3: Create a `BufferedWriter`

```
BufferedWriter bw =
  new BufferedWriter(fw);
```

Writing to Files

Step 4: Write to `BufferedWriter` as needed

```
bw.write("Hello");
bw.newLine();
bw.write("World");
bw.newLine();
```

Writing to Files

Step 4: Write to `BufferedWriter` as needed

```
bw.write("Hello");
bw.newLine();
bw.write("World");
bw.newLine();
```

Step 5: Close the `BufferedWriter`

```
bw.close();
```

Example:
`WriteStrings.java`

BufferedWriter

Observation: `PrintWriter` **seems to do everything** `BufferedWriter` **does, so why is** `BufferedWriter` **needed?**

`BufferedWriter`

Observation: `PrintWriter` seems to do everything `BufferedWriter` does, so why is `BufferedWriter` needed?

- Acts as a *buffer*
  - Layer between us saying `write` and the actual writing to the file
- Repeated short writes to files is **slow**
- Buffering idea: collect "writes" together in memory, then write to file all at once

finally

Motivation

Sometimes we want to perform an action, whether or not an exception is thrown.

Motivation

Sometimes we want to perform an action, whether or not an exception is thrown.

```
try {
  maybeThrowException();
  maybeDoThis();
} catch (SomeException e) {
  maybeDoThat();
} finally {
  alwaysDoThis();
}
maybeDoTheOtherThing();
```

Example:
`FinallyExample.java`

Common Use

- `finally` is often used to make sure a file was closed, even if an exception was thrown while manipulating the file
  - `WriteStrings.java` will **not** do this
  - See `WriteStringsFinally.java`