Noroff
School of technology
and digital media

Friday - Week 1

File IO

File IO

- Accessing a disk is ‘slow’ comparably
- Need to use an external library
- May need to handle an exception
    - try-catch

File IO

- Need the appropriate library:
- import java.io.*;
- Then we create an Input (for reading)

  `FileInputStream`
- Or an Output (for writing)

  `FileOutputStream`

Different Types of Readers/Writers

- FileInputStream / FileOutputStream
  - read/write bytes
- FileReader/FileWriter
  - read/write chars
- BufferedReader/BufferedWriter
  - read/write entire line at a time

Example

```
try (BufferedReader br = new BufferedReader(new FileReader(file))) {
    String line;
    while ((line = br.readLine()) != null) {
       // process the line.
    }
}
```

Noroff
School of technology
and digital media

Friday - Week 1

Exceptions

Try - Catch Blocks

- We cannot guarantee the state of the file / hardware:
  - Disk may fail / crash
  - File may be corrupt
  - File may be protected or missing

Solution

- Place volatile code in a try block
- Usually followed by a catch block

Example

```
try( /* optional */ ) {
// code that may cause an exception
}
catch (Exception e) {
// hande e
}
```

Catching

- We catch by matching our the exception with a conditional
- If we expect a IOException;
  then we must catch an IOException

Multiple Exception types

- We can catch the superclass exception type:
  `Exception e`
- Or we can have multiple catch blocks:

```
try { /* code */ }
catch(SpecificException e) { /* code */}
catch(VagueException e) { /* code */}
catch(Exception e) { /* code */}
```

Noroff
School of technology
and digital media

Friday - Week 1

Working with git

Covered Previously // External Notes

Noroff
School of technology
and digital media

Friday - Week 1

Tasks

Task 12 [Optional]

- Complete any previous task that you failed to complete earlier this week

  OR

- Resubmit a previous task that you want me to reevaluate based on an updated solution.