# File Storage in Java

Slides prepared and presented
by Marios Fokaefs
(fokaefs@cs.ualberta.ca)

## The traditional way (Read)

- To read a file we need a Reader. But Reader is abstract.
- One very good subclass is BufferedReader
- BufferedReader(Reader input)
  - BufferedReader(FileReader input)
  - BufferedReader(InputStreamReader input)
- Both FileReader and InputStreamReader take a File as an argument
- Absolute vs Relative path
  - Absolute path contains all the folders from the root to the file (e.g. /home/project/files/file.txt)
  - Relative path has all the folders from the working directory to the file(e.g. files/file.txt)
- Read line by line and character by character

# Example Code (Read)

```
BufferedReaderin = new
  BufferedReader(new FileReader(new
  File(“files.txt”)));
String next = in.readLine();
while(next != null) {
next = in.readLine();
}
in.close();
```

# The traditional way (Write)

- To write a file we need a Writer. But Writer is abstract.
- One very good subclass is BufferedWriter
- BufferedWriter(Writer output)
  - BufferedWriter(FileWriter output)
  - BufferedWriter(OutputStreamWriter output)
- Both FileWriter and OutputStreamWriter take a File as an argument.
  - If you want to append data to an existing file, add **true** in the constructor of these classes
- If the file doesn't exist it will be created.

# Example Code (Write)

```
BufferedWriterout = new
  BufferedWriter(new FileWriter(new
  File("file.txt"), true));
out.write("StudentA\t
  studenta@cs.ualberta.ca");
out.newLine();
out.write("StudentB\t
  studentb@cs.ualberta.ca");
out.close();
```

# Useful Links for String manipulation

- String
  - http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html
- Pattern
  - http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html
- Integer
  - http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Integer.html
- Double
  - http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Double.html

## The “smart” way

- Store objects instead of text
- Classes that are to be stored must implement the Serializable interface
- BufferedReader and BufferedWriter are repalced by ObjectInputStream and ObjectOutputStream

# Example Code (Read)

```
public static PeopleCatalog openPeopleCatalog(File
   f) {
   PeopleCatalog pc = null;
   try {
       FileInputStream fin = new FileInputStream(f);
       ObjectInputStream ois= new
             ObjectInputStream(fin);
       pc = (PeopleCatalog)ois.readObject();
       ois.close();
   }
   catch(ClassNotFoundExceptioncnfe) {
             cnfe.printStackTrace(); }
   catch(IOExceptionioe) { ioe.printStackTrace(); }
   return pc;
}
```

# Example Code (Write)

```
public static void savePeopleCatalog
  (PeopleCatalog pc, File f) {
try{
  FileOutputStream fout= new
     FileOutputStream(f);
  ObjectOutputStream oos= new
     ObjectOutputStream(fout);
  oos.writeObject(pc);
  oos.close();
}
catch(IOExceptionioe) { ioe.printStackTrace();
  }
}
```

# Fun Time!!

- Write a java program that stores a list of students' names and email addresses in a file. Then access the file and print the list of students in the console.
- Classes
  - Student
    - Fields:Name, Email
  - StudentList
    - Fields: List of Students
    - Methods: openFile, saveFile
- Duration: 30 min.