Files and I/O

General idea

- Several *classes* (such as Scanner) exist in package *java.io*
- Make object of such a class by **`new Scanner(...)`**
- On the dots fill in an object that does part of the task
- Example
  - `System.in` is object that handles input from keyboard
  - `scanner = new Scanner(System.in)` creates object that processes keyboard input on higher level (separates them into *tokens* (words), etc.)

Scanner (java.util)

- **Scanner** processes input from **arbitrary source**, chops them into *tokens* and offers tokens via:
  - `next()`: consume next token from input. Default: whitespace separated *words*
  - `nextInt()`: consume next token and interpret as integer (if possible)
  - `nextDouble()`, `nextBoolean()`,...: ditto
  - `nextLine()`: reads the rest of the current line and goes to the next one
  - `hasNext()`: returns true if there are tokens on input, false otherwise
    - note: usually true on keyboard input (`next()` will wait to see if there is more)
  - other token separating behaviour can be specified (with `useDelimiter`)
  - hasNextInt(), etc.: true if next token is in int (etc.) format

File (java.io)

- When program terminates, all data is lost
- Files store data more permanently
- an object of class `File` records information about a file on disk
- `new File("funnyfile.txt")` creates a File object that can be connected to the disk file *funnyfile.txt*
  - path names are possible: `new File("C:\\My Documents\\funny.doc")`
- it doesn't create or connect to a file by itself

Connecting

- a File object can be connected to a Scanner object:

- ```
  File file = new File(“funnyfile.txt”);
  Scanner scanner = new Scanner( file );
  ```

- then reading all lines into array **a** (which should be big enough):

```
int i = 0;
while ( scanner.hasNext() ) {
    a[i] = scanner.nextLine();
    i++;
}
```
```

PrintWriter (java.io)

- has objects that can perform `print` and `println` like `System.out`

- `File file = new File("funnyfile.txt");`
  `PrintWriter pw = new PrintWriter(file);`
  creates the file *funnyfile.txt* and an object to write into it.

  - if *funnyfile.txt* exists already, it is overwritten (contents is removed)

- Suppose **`a`** is an array of Strings, then writing the contents of **`a`** as lines into the file *funnyfile.txt*:

```
for (int i=0; i<a.length; i++) {
    pw.println(a[i]);
}
```

- Fixing its contents and making the file available for, e.g., input:
  `pw.close();`

Example

```
// copies a text file line by line
import java.util.*;
import java.io.*;

public class FileCopy {
    Scanner scanner;
    File source;
    File target;
    PrintWriter pw;
    String filenameSource = "rhubarb.txt";
    String filenameTarget = "rhubarb_copy.txt";

    void copy() {
        try {
            String line = null;
            source = new File( filenameSource ) ;
            scanner = new Scanner( source );
            target = new File( filenameTarget );
            pw = new PrintWriter( target );

            while (scanner.hasNext()) {
                line = scanner.nextLine();
                pw.println( line );
            }

            pw.close();
        } catch(FileNotFoundException e) {
            System.out.println("Could not open file due to");
            System.out.println(e);
        }
    }

    public static void main(String[] args) {
        new FileCopy().copy();
    }
}
```