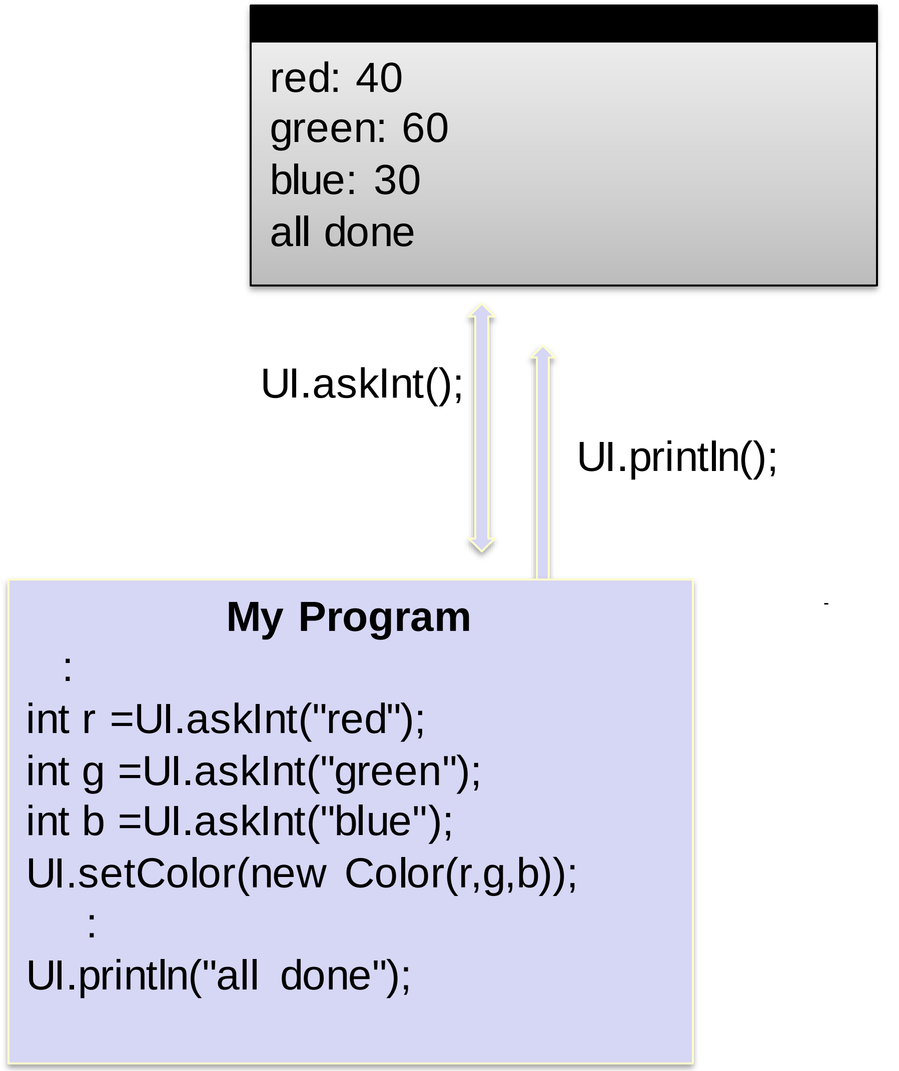
Files

- The UI text pane window is transient:
  - Typing large amounts of input into the text pane is a pain!
  - It would be nice to be able to save the output of the program easily.
- Large amounts of text belong in files
- How can your program read from a file and write to a file?
- Writing to files is like writing to the UI text pane!
  - Use print, println, printf methods
  - But, need a PrintStream object instead of UI
- Reading from files is a bit different
  - Doesn't use "ask…" methods
  - Lots of different ways of reading from files
  - We will just use a very simple one that reads a list of lines from a text file
  - We will use Scanner objects to break up the lines into separate values.

Text with the text pane



```
red: 40
green: 60
blue: 30
all done
```

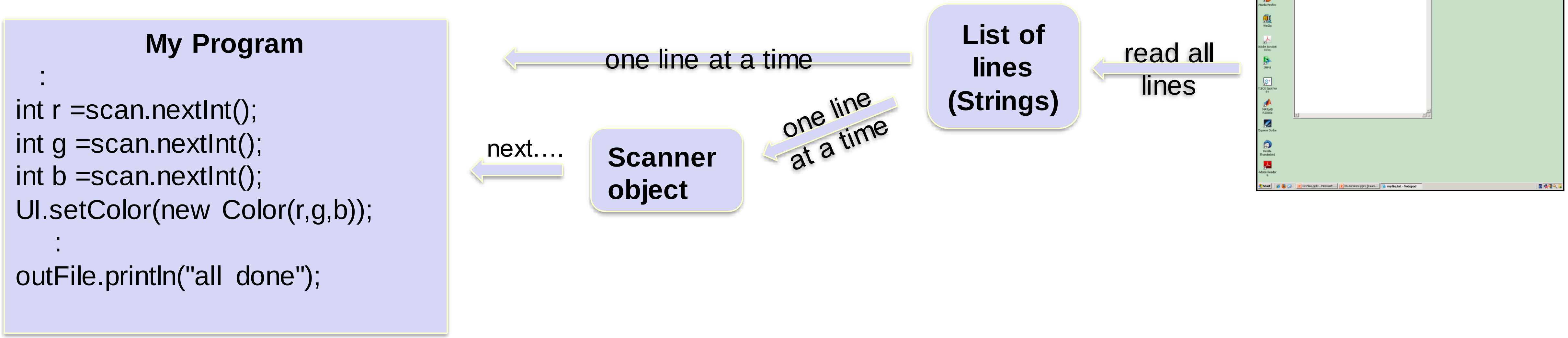
```
   :
int r =UI.askInt("red");
int g =UI.askInt("green");
int b =UI.askInt("blue");
UI.setColor(new  Color(r,g,b));
     :
UI.println("all  done");
```

Text with Files

Reading data from a file:

- Read the file into a list of lines (Strings).
- Either
  - Use the lines directly, or
  - Use a Scanner object to get the values out of the lines

A real file: “mydata.txt”



```
   :
int r =scan.nextInt();
int g =scan.nextInt();
int b =scan.nextInt();
UI.setColor(new Color(r,g,b));
     :
outFile.println("all done");
```

```
40 apples
60 oranges
30 pineapples
50 bananas
```

Reading lines from a file:

```
/** Read lines from a file and print them to UI text pane. */
public void displayFile(String fileName){
    try {
        List<String> allLines = Files.readAllLines(Path.of(fileName));
        for (String line : allLines){
            UI.println(line);
        }
    } catch (IOException e) { UI.println("File failure: " + e); }
}
```

Missing bits to handle exceptions !!

what to do

what to do if it goes wrong

- Files.readAllLines(Path.of(….)) reads every line of the file into a List of Strings.
- Almost right, but compiler complains!!!
- Dealing with files may “raise exceptions”
- Need a **try** { ………………… } **catch** (…){ … }

Reading lines from a file, ask user for file:

```
/** Read lines from a file and print them to UI text pane. */
public void displayFile(){
    String fileName = UIFileChooser.open("File to open ");
    try {
        List<String> allLines = Files.readAllLines(Path.of(fileName));
        for (String line : allLines){
            UI.println(line);
        }
    } catch (IOException e) { UI.println("File failure: " + e); }
}
```

UIFileChooser.open("title")	lets user select an existing file to read from
UIFileChooser.save("title")	lets user select a new or existing file to write.

Text with Files

Reading data from a file:

- Read the file into a list of lines (Strings).
- Use a Scanner object to get the values out of the lines

Writing data to a file:

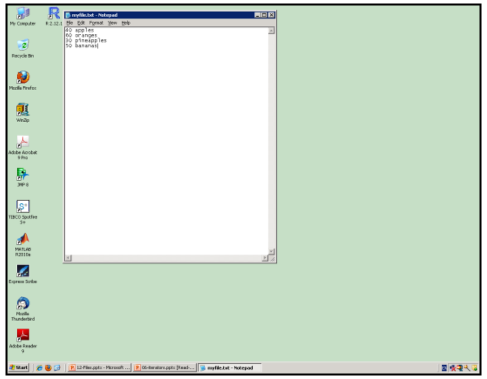
- Use a PrintStream object.

My Program

```
   :
int r =scan.nextInt();
int g =scan.nextInt();
int b =scan.nextInt();
UI.setColor(new Color(r,g,b));
     :
outFile.println("all done");
```

$$\begin{tikzcd} \text{My Program} \arrow[r, "\text{print...}"'] & \textbf{PrintStream object} \arrow[r] & \text{A real file: “output.txt”} \end{tikzcd}$$

A real file: “output.txt”



```
40 apples
60 oranges
30 pineapples
50 bananas
```

Writing to a file:

```
/** Read lines from a user and print them to a file. */
public void makeFile(String filename){
    ArrayList<String> lines = UI.askStrings("Type in file contents:");
    try {
        PrintStream outfile = new PrintStream(filename);
        for (String line : lines) {
            outfile.println(line);
        }
        outfile.close();
    } catch (IOException e) { UI.println("File failure: " + e); }
}
```

- PrintStreams work just like printing to UI
- Close the file when finished.
- Need a try { … } catch (…){….} around printing to files also.

Writing to a file, using UIFileChooser

```
/** Read lines from a user and print them to a file. */
public void copyFile(){
    ArrayList<String> lines = UI.askStrings("Type in file contents:");
    String filename = UIFileChooser.save("Filename to save to");
    try {
        PrintStream outfile = new PrintStream(filename);
        for (String line : lines) {
            outfile.println(line);
        }
        outfile.close();
    } catch (IOException e) { UI.println("File failure: " + e); }
}
```

- UIFileChooser.save("….prompt….") lets the user choose a (possibly new) file.

UIFileChooser

- So far, we've specified which file to open and read or write with a String.

- How can we allow the user to *choose* a file?
  - UIFileChooser class (part of ecs100 library, like UI)

Method	**What it does**	**Returns**
open()	Opens dialog box; User can select an **existing** file to open. Returns name of file or null if user cancelled.	String
open(String title)	Same as open(), but with specified title;	String
save()	Opens dialog box; User can select file (possibly new) to save to. Returns name of file, or null if the user cancelled.	String
save(String title)	Same as save(), but with specified title.	String

Copying a file

```
/** Read lines from one file and print them to another file. */
public void copyFile(){
    String fromFile = UIFileChooser.open("File to copy");
    String toFile = UIFileChooser.save("Filename to save to");
    try {
        List<String> lines = Files.readAllLines(Path.of(fromFile));
        PrintStream outfile = new PrintStream(toFile);
        for (String line : lines) {
            outfile.println(line);
        }
        outfile.close();
    } catch (IOException e) { UI.println("File failure: " + e); }
}
```

- UIFileChooser.open("….prompt….")   lets the user choose an existing file.

Doing more with data in a file:

```
/** Find all lines in a file containing a search word. */
public void findWordInFile(){
    String fileName = UIFileChooser.open("Choose file to search");
    String word = UI.askString("Word to search for");
    try {
        List<String> allLines = Files.readAllLines(Path.of(fileName));
        int lineNumber = 1;
        for (String line : allLines){
            if (line.contains(word)){
                UI.printf("Found %s on line %d: %s\n", word, lineNumber, line);
            }
            lineNumber++;
        }
    } catch (IOException e) { UI.println("File failure: " + e); }
}
```

Doing more with data in a file:

What if the file has numbers? How do we get numbers out of the Strings?

- If a String consists of one number only:
- can use Double.parseDouble(line) or Integer.parseInt(line)

```
/** Returns total of numbers in a file (one number per line). */
public double totalFile(String fname){
    try {
        double total = 0;
        for (String line : Files.readAllLines(Path.of(fname))){
            total += Double.parseDouble(line);    // fails if not a number!!!
        }
    } catch (Exception e) { UI.println("File failure: " + e); }
    return total;
}
```

Doing more with data in a file:

What if each line of the file has multiple values?

How do we get individual values out of the Strings?

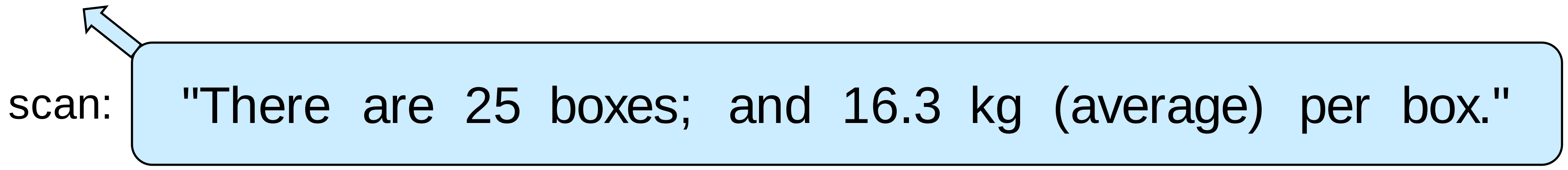
fruit.txt

```
4447 quince  11.45
4430 pineapple 6.82
4041 red-plum 5.99
4416 D'Anjou-pear  5.44
4011 Banana     2.99
```

Use a Scanner

Scanners

- Scanner: a class in Java that allows a program to read values out of a String (or any other source of characters…)
- Gives the values one at a time



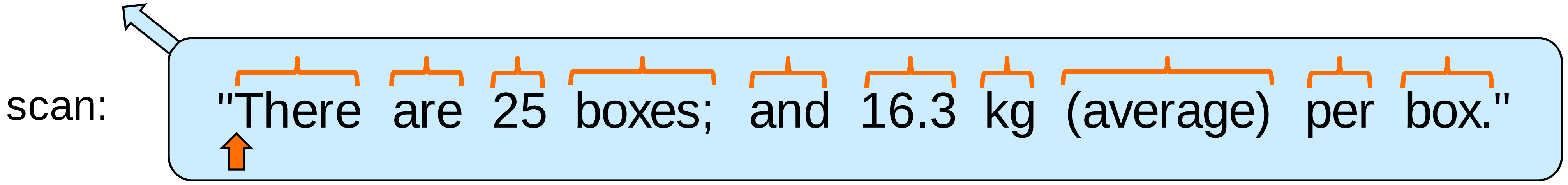
To get a Scanner:

- Create a new Scanner object, passing it the source:

```
Scanner scan = new Scanner("There are 25 boxes; and 16.3 kg (average) per box.");
Scanner sc = new Scanner(UI.askString("Enter some text"));
String line = ….
Scanner lineSc = new Scanner(line);
```

Scanners

- A Scanner breaks up the source string into a sequence of tokens, separated by spaces or tabs.



- Token: a word, a number, or … any sequence of non-space characters.
- A Scanner provides the tokens, one at a time, using the .next…() methods:

scan.next()	⇒	next token as a string
scan.nextInt()	⇒	next token <u>as an int</u> (error if next token is not an integer)
scan.nextDouble()	⇒	next token <u>as a double</u> (error if next token is not a number)

- Each call to .next…() moves the "cursor" to the end of the token.

Reading Tokens from a Scanner

- If you know how many tokens in the Scanner, you can just pull them out:

```
Scanner scan = new Scanner ("4447 quince 11.45");
String PLU = scan.next();
String product = scan.next();
String price = scan.next();

Scanner scan = new Scanner ("This string has (exactly) 10 tokens:  a-b-c-d & 9.0  #10");
for (int i = 0; i <10; i++){
    String tok = scan.next();
    UI.printf("Token %d : %s\n", i, tok);
}
```

- Tokens are Strings (whether they look like words, numbers, other…)
- Can only take them out in order

Reading from a Scanner

- If you know the number of tokens and their types, can extract them.
  - Eg, if the string has an integer, a word, and a double:

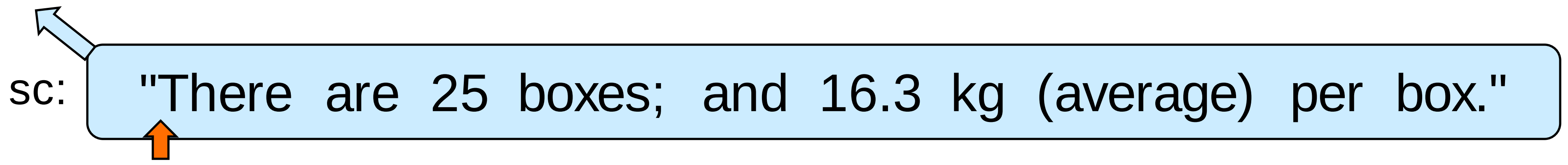
```
Scanner scan = new Scanner ("4447 quince 11.45");
int PLU = scan.nextInt();
String product = scan.next();
double price = scan.nextDouble();

Scanner scan = new Scanner ("4430 pineapple 6.82");
double PLU = scan.nextDouble();
double ~~product = scan~~.nextDouble();
int price = ~~scan.next~~Int();
```

- Safe to read a number as a String, or an integer as a double.
- Not safe to read a non-number as a number, or a double as an int

Reading from a Scanner

- If the number of tokens in a scanner is unknown,
  How can you tell when to stop?

```
Scanner sc = new Scanner (UI.askString("Enter a line of text"));
```



- Scanner lets you ask if there is another token using the .hasNext() method:

  sc.hasNext() ⇒ true or false: is there another token in the scanner?

- Can use a while loop with a Scanner:

```
while (sc.hasNext()){
    String word = sc.next();
    ….
}
```