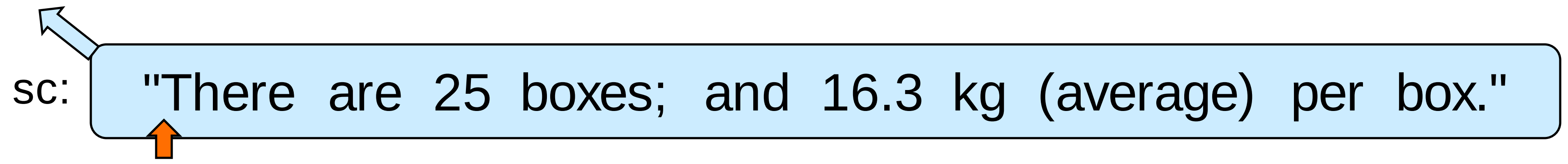
Reading from a Scanner

- 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();
    ….
}
```

Reading from a Scanner

- If the types of the tokens in a Scanner can vary,
  How can you tell what type they are?

- Scanner lets you "peek" at the next token using the .hasNext…() methods:

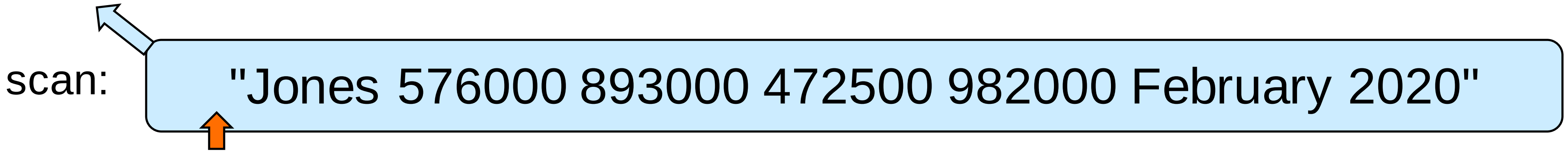
scan.hasNextInt()	⇒ true or false: is there another token AND is it an integer?
scan.hasNextDouble()	⇒ true or false: is there another token AND is it a number?

```
Scanner sc = new Scanner (UI.askString("Enter some tokens"));
int total = 0;
while (sc.hasNext()){
    if (sc.hasNextInt()){   // if the next token is an integer, read it and add to total
        int num = sc.nextInt();
        total = total + num;
    }
    else {                  // if next token is not an integer, read it and throw it away
        sc.next();
    }
}
```

"There are 25 boxes; and 16.3 kg (average) per box."

Reading from a Scanner

- More unknown values:



```
Scanner scan = new Scanner (line);
String salesperson = scan.next();
double total = 0;
while (scan.hasNextDouble()){
        total = total + scan.nextDouble();
}
String month = scan.next();
int year = scan.nextInt();
```

Scanner "next" methods

Method	What it does	Returns
next()	Read and return next token	String

nextInt()
nextDouble()
	Read the next token.
	Return it as a number, if it is a number.
	Throws an exception if it is not a number.
		int
		double

nextBoolean()	Read the next token. Return true if it is "true"; return false if it is "false". Throws an exception if it is anything else.	boolean
hasNext()	Returns true if there is another token	boolean

hasNextInt()
hasNextDouble()
hasNextBoolean()
	Returns true if there is another token AND
	the next token is an int / double / Boolean
		boolean

nextLine()	Read characters up to the next end-of-line and return them as a string. Reads and throws away the end-of-line character. If the first character is an end-of-line, then it returns an empty string ("").	String

Files and Scanners

If a file has lines, each with several values in it:

- Wrap each line from the file in a Scanner, and
- Read the values from the Scanner.

```
List<String> allLines = Files.readAllLines(Path.of("image.pxm"));
for (String line : lines){
    Scanner scan = new Scanner (line);
    UI.setColor(new Color (scan.nextInt(), scan.nextInt(), scan.nextInt()));
    UI.fillRect(x, y, 2,2);
    x = x+2;
    if (x > RIGHT) {
        x = LEFT;
        y = y+2;
    }
}
```

image.pxm

```
25 53 201
240 2 150
100 250 0
240 220 220
….
```

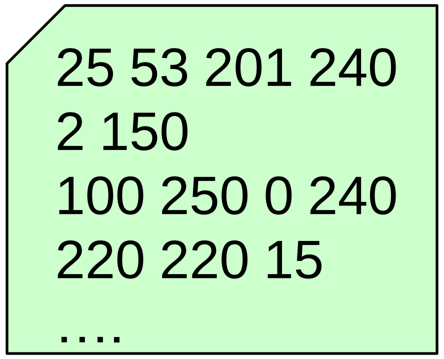
Files and Scanners

If a file has lines, each with varying number of values in it:

- Wrap each line from the file in a Scanner, and
- Read the values from the Scanner.

```
List<String> allLines = Files.readAllLines(Path.of("numbers.txt"));
double max = Double.NEGATIVE_INFINITY;
for (String line : lines){
    Scanner scan = new Scanner (line);
    while (scan.hasNextDouble()){
        double num = scan.nextDouble();
        if (num > max) {
            max = num;
        }
    }
}
```

numbers.txt



```
25 53 201 240
2 150
100 250 0 240
220 220 15
….
```

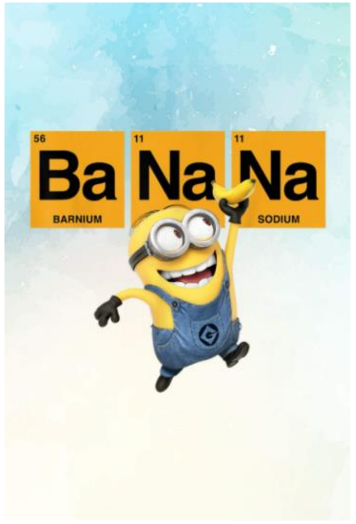
Files and Scanners

- If each line has fixed number of values of different types:

```
try {
    List<String> allLines = Files.readAllLines(Path.of("fruit.txt"));
    for (String line : lines){
        Scanner scan = new Scanner (line);
        int PLU = scan.nextInt();
        String product = scan.next();
        double price = scan.nextDouble();
        ….. // do something with the values
    }
} catch (IOException e) { UI.println("File failure: " + e); }
```

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
```



A common simple pattern

- File with one entity per line, described by multiple values:

```
List<String> lines = Files.readAllLines(Path.of(filename));

for (String line : lines){
    Scanner sc = new Scanner(line);
    String type = sc.next();
    double  cost = sc.nextDouble();
    int wheels = sc.nextInt();
    String colour = sc.next();
    String make = sc.next()
    if (wheels > 4) {
        ….
    }
    else {
        …
    }
}
```

```
bicycle 1025 2 green Giant
truck   120000 18 black Isuzu
car  26495 4 red Toyota
```

Read all the values into variables

process the values in the variables

Reading files line by line

If items have a varying number of values:
May need loop within each line:

```
/**Adds up sales of item on each line of a file */
public void addCounts(){
    List<String> lines = Files.readAllLines(Path.of("data.txt"));
    for (String line : lines) {
        Scanner sc = new Scanner(line);
        int code = sc.nextInt();
        String item = sc.next();
        int lineTot = 0;
        while (sc.hasNextInt()) {
            lineTot = lineTot + sc.nextInt();
        }
        UI.printf("%s (%d): %d\n", item, code, lineTot);
    }
}
```

```
973 biscuits  27 33 15 4 9
731 cake 3 5 2
189 fruit 54 2 83 96
446 beans 1 3 2 5 3 4 7 2 5 1
```

Processing values from a line

```
try {
    List<String> lines = Files.readAllLines(Path.of(filename));
    for ( String line : lines  ){
        Scanner sc = new Scanner(line);
        double left = sc.nextDouble();
        double top = sc.nextDouble();
        double wd = sc.nextDouble();
        double ht = sc.nextDouble();
        String shape = sc.next();
        int r = sc.nextInt();
        int g = sc.nextInt();
        int b = sc.nextInt();

        UI.setColor( new Color (r, g, b) );
        if (shape.equals("Oval") ){  UI.fillOval(left, top, wd, ht);  }
        else {                       UI.fillRect(left, top, wd, ht); }
    }
} catch (IOException e) { UI.println("File failure: " + e); }
```

Diagram.txt

```
50.0 20.0 10.3 7.8 Oval 25 53 201
75.0 100.2 16.9 12.0 Rect 240 2 150
304.0 28.7 25.0 51.5 Oval 100 250 0
…
```

extracting all the values on the line

Do something with all the values