CMPSC 100

Computational Expression

Data Type	Size	Min value	Max Value
byte	1 byte	-128	127
short	2 bytes	-32,768	32,767
int	4 bytes	-2,147,483,648	2,147,483,647
long	8 bytes	- a lot	+ a lot
float	4 bytes	7 decimals	7 decimals
double	8 bytes	15 decimals	15 decimals
char	2 bytes	0	65,536
boolean	(not important)	0 (true)	1 (false)

“primitive” data types

Data Type	Size	Min value	Max Value
String	Various	?	?
Scanner	Various	?	?

“reference” data type

These values aren’t important anymore.

Data Type	Size	Min value	Max Value
String	Various	?	?
Scanner	Various	?	?
Random	Various	?	?

“reference” data type

Reference types: Scanner

Scanner exists outside of the Java API, so we have to import it:

Contained in a class called Scanner

```
import java.util.Scanner;
```

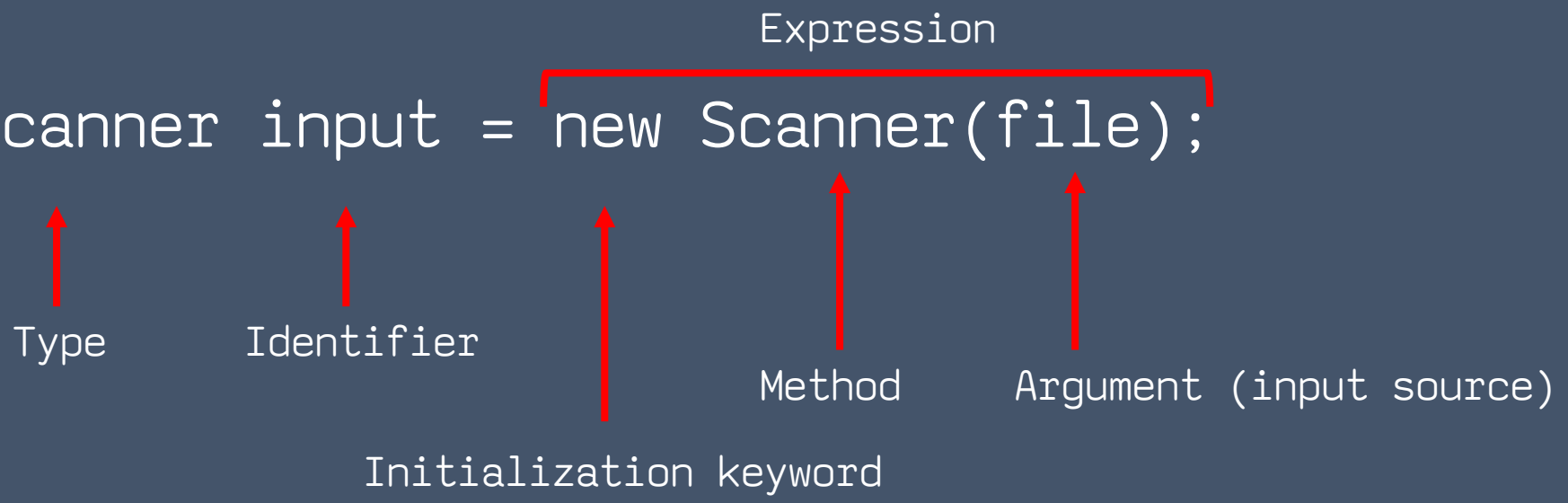
Part of the java.util “package”

Reference types: Scanner

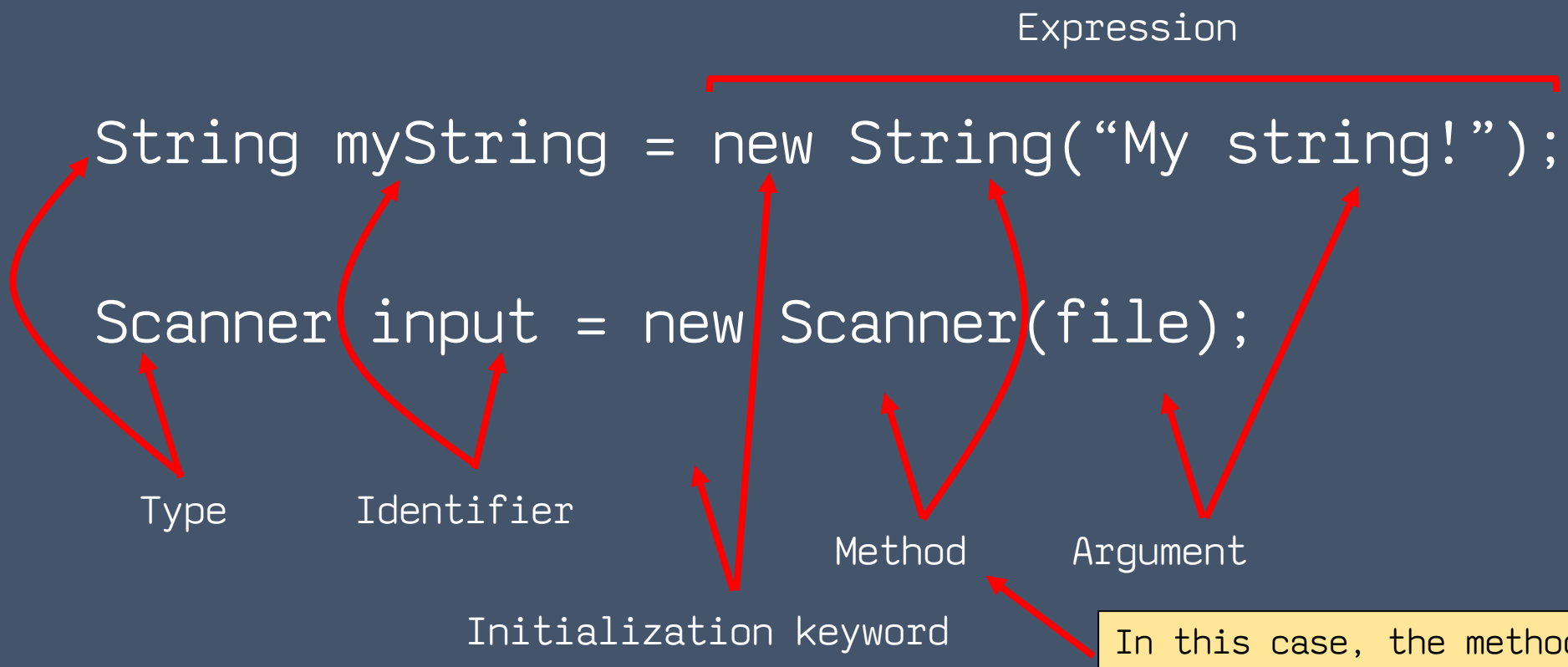
Once we've imported it, we can "summon" its powers when we invoke it:

```
File file = "inputs/cupcakes.nomnomnom";

Scanner input = new Scanner(file);
```



Reference types are objects



In this case, the method is called the "**constructor**" and every reference type has one - it specifies the *minimum* amount and type of data needed to create an **object**

Objects: Scanner

Refers to a file containing the Scanner code

```
import java.util.Scanner;
```

```
Scanner input = new Scanner(System.in);
```

Creates the actual Scanner object

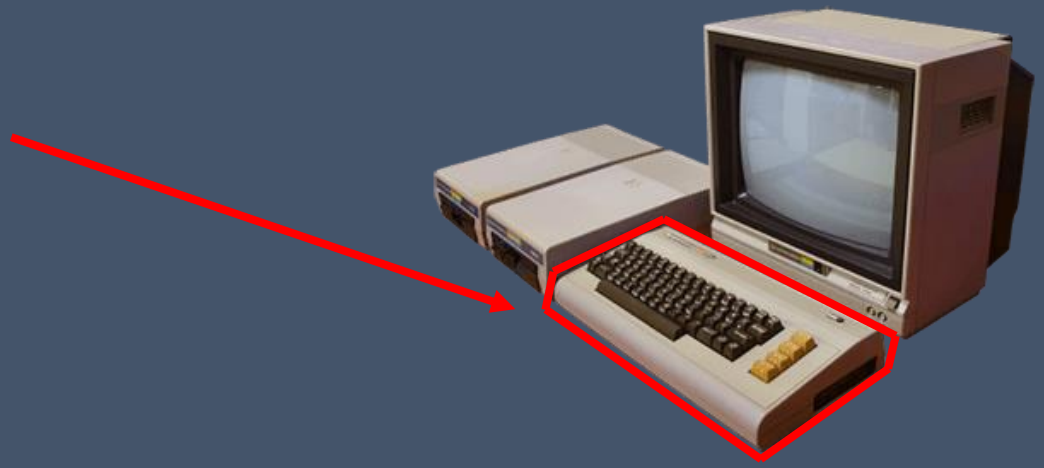
(Pics or it didn't happen)

Objects: Scanner

Creating a Scanner object requires 1 piece of data which represents an input:

- Files
- System input (STDIN)

  (System.in)
- Strings



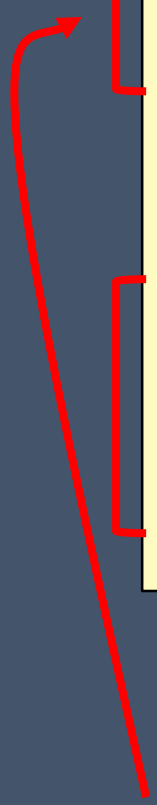
```
Scanner(InputStream source)
Scanner(File source)
Scanner(String source)
  Constructor: Creates a new object to process input

String next()
  Processes the next “input token” up to (not including) a space

String nextLine()
  Processes the next full line up to (and including) the \n (new line) character

int nextInt()
  Processes the next integer available in the input

double nextDouble()
  Processs the next double-precision floating point number in the input
```



“Input token” here refers to any value up to (and not including a space)

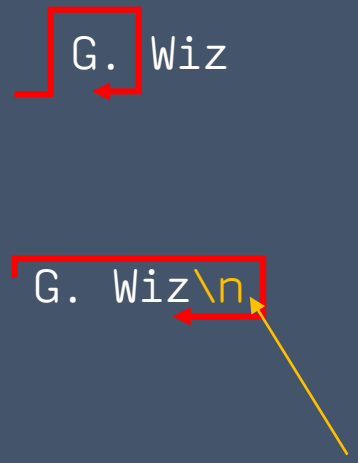
BEWARE! Any of the next____ methods not including “Line” take up to the first delimiter (space character) and do not automatically move to the next line!

```
int number   =  input.nextInt();          6          // 6

double number =  input.nextDouble();       6          // 6.0

String fName   =  input.next();            G. Wiz     "G."

String fullName = input.nextLine();        G. Wiz\n   "G. Wiz"
```



cd to your activities folder
perform a git pull download master

cd to the activity-06 folder

Objects and Methods

- Objects which are Reference Types have methods
- To use their methods (“powers”), we have to “initialize” them using the new keyword
- We call those methods using the dot operator
- Methods require space for arguments, even if that argument is empty
- Methods can return data in the form of a data type

```
String name = “G. Wiz”;
// String name = new String(“G. Wiz”);

int len = name.length();
String fName = name.substring(0,2);
String lName = name.substring(3,len);
char lastInitial = name.charAt(3);
String fullName = name.replace(“G.”,“Gator”);
```

```
File file = new File(“input/items.list”);
Scanner input = new Scanner(file);

int intNum = input.nextInt();
double dblNum = input.nextDouble();
String firstName = input.next();
String fullName = input.nextLine();
boolean isRead = input.nextBoolean();
```

Objects: Random

The Random class is an object that does exactly what it seems - it enables us to generate “pseudorandom” numbers

`Random ()`
Constructor: creates a new pseudorandom number generator.

`float nextFloat ()`
Returns a random number between 0.0 (inclusive) and 1.0 (exclusive).

`int nextInt ()`
Returns a random number that ranges over all possible int values (positive and negative).

`int nextInt (int num)`
Returns a random number in the range 0 to `num-1`.

Some methods of the `Random` class

Takes one `int` as an `argument`

Returns an `int`

Objects: Random

Like Scanner, we don't get this "for free" - we have to ask for (import) it:

Contained in a class called Random

```
import java.util.Random;
```

Part of the java.util "package"

cd to your activities folder
perform a git pull download master

cd to the activity-07 folder

Start our activity by implementing the following methods and printing their results:

- nextInt
- nextDouble