WELCOME TO CS-1180!!

Acknowledgements: Slides created based off material provided by Dr. Travis Doom

JAVA

- Object-oriented
- Popular choice in industry
- Portable
  - Execution without recompiling
- Many libraries, IDEs, etc.
  - VSCode, IntelliJ, NetBeans, Eclipse, etc.

VOCABULARY

- Semantics
  - Meaning
- Syntax
  - Grammar
- Style
  - Conventions for readability
- Compiler/IDE
  - Spellchecker

HELLOWORLD EXAMPLE

- Semantics
  - Classes
    - Containers for a portion of a design
  - Method (/function/subroutine) header
    - Containers for a specific task
    - main method
  - Method body
    - Statements implementing an algorithm

HELLOWORLD EXAMPLE

- Syntax
  - One class per java file
  - Name matches file name
  - One main method
  - Keywords
    - public, class, static, void, main, etc.
  - Identifiers
    - Your names for classes, methods, variables, etc
  - Literals
    - Unchanging constant value
    - Ex: `5, 'a', 7.5`

HELLOWORLD EXAMPLE

- Style
  - White space usage
  - Descriptive names
  - Capital letters for class names; lower case for methods
  - Curly braces
  - Statements
    - Generally, end with a semicolon
    - Whitespace is ignored

COMMENTS

- Incredibly important
- Three ways:
  - `// one line comment`
  - ```
    /*
            Block comment
      */
    ```
  - ```
    /** @author Clarissa
              Java doc comments have specific syntax to generate separate
    documentation
      */
    ```

VARIABLES

- Named location to hold information
- Data type to describe the type of information
- Ex: `dataType variableName`
- Primitive data types
  - byte, short, **int**, long, float, **double**, **boolean**, **char**
- Assignments
  - LHS: must be a variable
  - RHS: can be a literal, variable, method, equation, anything that results in a value
  - Ex:
```
value = 5;
letter = 'a';
```

OPERATORS

- Used to manipulate variables
- Java has 5 arithmetic operators

Operator	Meaning	Type	Example
+	Addition	Binary	total = cost + tax;
-	Subtraction	Binary	cost = total – tax;
*	Multiplication	Binary	tax = cost * rate;
/	Division	Binary	salePrice = original / 2;
%	Modulus	Binary	remainder = value % 5;

Operator	Associativity	Example	Result
- (unary negation)	Right to left	x = -4 + 3;	-1
* / %	Left to right	x = -4 + 4 % 3 * 13 + 2;	11
+ -	Left to right	x = 6 + 3 – 4 + 6 * 3;	23

JAVA.LANG.MATH

- Provides mathematic operations beyond the standard operators
  - Rounding
    - floor(x), ceil(x), round(x)
  - Trig
    - sin(x), cos(x), tan(x), atan(x), asin(x), acos(x), log(x), exp(x)
  - Other Math operations
    - pow(x, y), sqrt(x), min(x, y), max(x, y)
  - Useful constants
    - PI, E

MORE ON FORMAT

- `System.out.printf(format, v1, v2, ...vn);`
- `String s = String.format(format, v1, ...);`
- For each variable:
  - %[flags][width][.precision][type]
  - Ex: `System.out.printf("Pi starts with %.1f", 3.14159);`
  - Output: `Pi starts with 3.1`
  - Ex: `System.out.printf("%+0,20.5f", 123456789.987654321);`
  - Output: `+00123,456,789.98765`

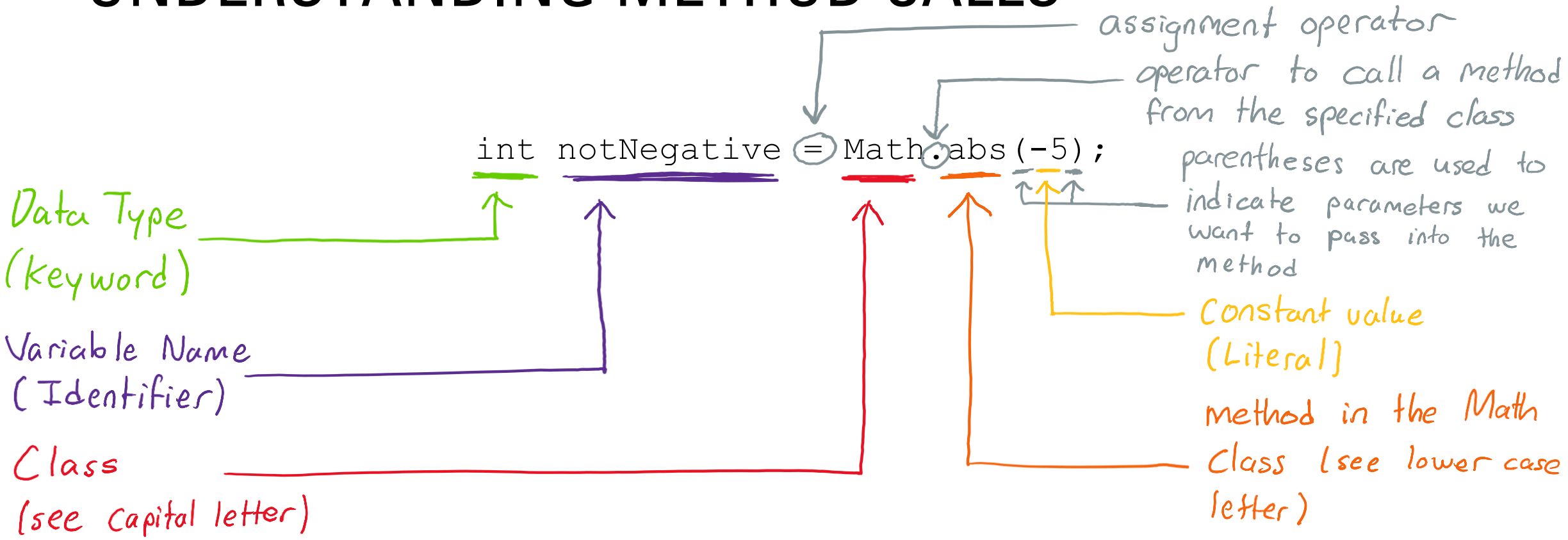
Flags	Meaning
-	Left justified
+	prefix with +/-
0	pad with zeros
,	separate by thousands
(	negatives in parens

Type	Meaning
%d	integer (digits)
%f	floating point
%e	exponential /scientific notation (floatingpoint)
%b	boolean
%c	character
%s	string

ESCAPE SEQUENCES

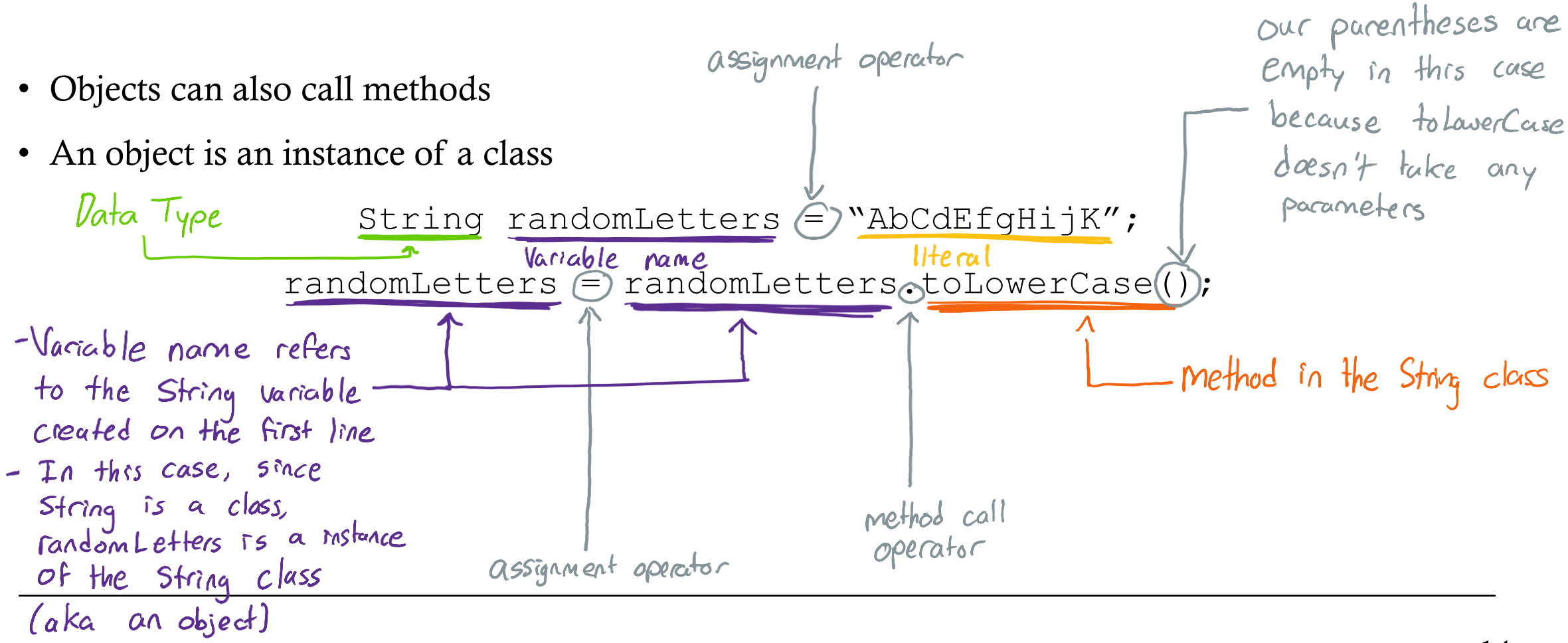
\n	newline	Advances the cursor to the next line for subsequent printing
\t	tab	Causes the cursor to skip over to the next tab stop
\b	backspace	Causes the cursor to back up, or move left, one position
\r	carriage return	Causes the cursor to go to the beginning of the current line, not the next line
\\	backslash	Causes a backslash to be printed
\’	single quote	Causes a single quotation mark to be printed
\”	double quote	Causes a double quotation mark to be printed

UNDERSTANDING METHOD CALLS



UNDERSTANDING METHOD CALLS

- Objects can also call methods
- An object is an instance of a class



NOTE ON IDENTIFYING METHODS

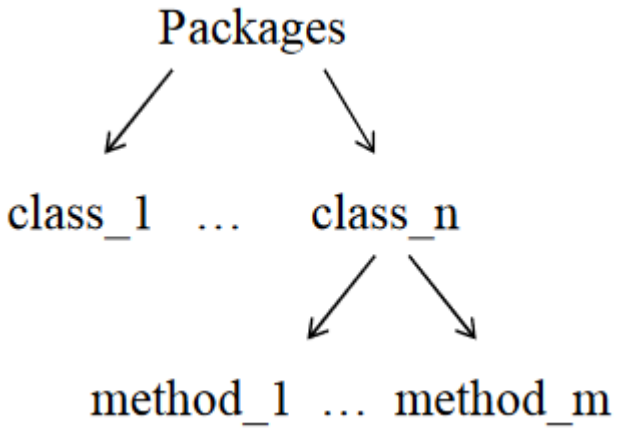
- Generally, start with a lowercase letter
- Often appear to the right of a period (when called from a class or object)
- Always have parentheses () after them

NOTE ON CALLING METHODS

- Methods can be called by a **class** or an **object**
- Classes start with a capital letter
  - EX: `double answer = Math.pow(4, 2);`
  - EX: `String rounded = String.format("My value is: %.2f", 4.38494);`
- Objects are variables that were created with a class as the data type
  - EX:
```
String greeting = "hello!";
String yell = greeting.toUpperCase();
```

JAVA LIBRARY METHODS

- java.lang (built-in)
  - System class performs system-level tasks
  - Ex: `System.out.println("Hello World");`
  - Math class includes useful math operations
  - Ex: `Math.abs(-5);`
  - String class features useful string operations
  - Ex: `String.valueOf(-5);`
- java.util (must import)
  - Ex: `import java.util.Scanner;`

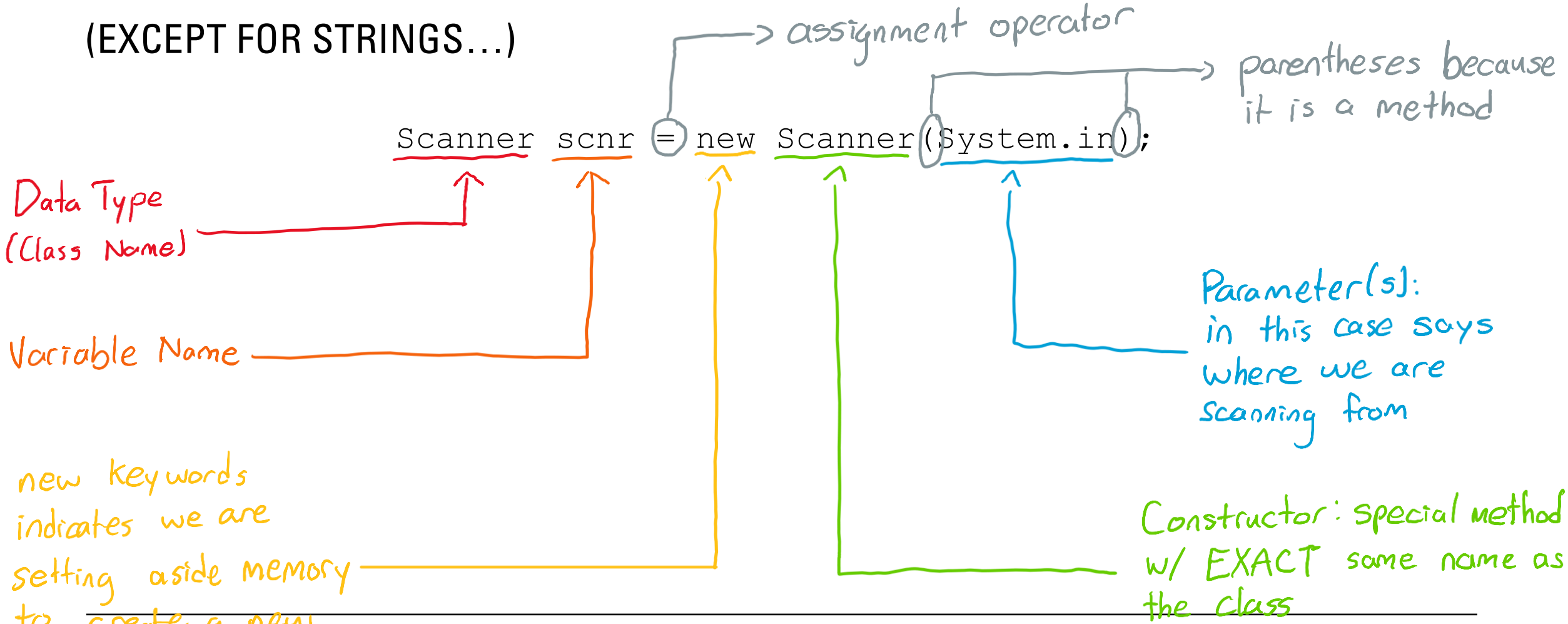


CONSOLE I/O

- Output
  - `System.out.println("Hello!");`
  - `System.out.print("Hi");`
  - `System.out.printf("Pi starts with %.1f", 3.14159);`
- Input
  - `Scanner input = new Scanner(System.in);`
  - `String answer = input.nextLine();`
  - `int answer = input.nextInt();`

HOW TO CREATE AN OBJECT

(EXCEPT FOR STRINGS…)



MORE ON SCANNER

- Must import: `import java.util.Scanner;`
- Creation: `Scanner input = new Scanner(System.in);`
- Useful methods:
  - `next()` returns the next token
  - `nextInt()` returns the next input as an integer
  - `nextDouble()` returns the next input as a double
  - `nextLine()` returns the next line as a String