**Table of contents**

**pg 2 coding terms**

**System** is a built-in Java class that contains tools for our program.

**Out** is short for output.

**Int** can store positive numbers, negative numbers and zero but can’t store fractions or decimals

**Doubles** can hold decimals as well as very large and small numbers

**Booleans** are used for true and questions

**Char** can hold any character like a letter, space, or punctuation mark. It must be surrounded by single quotes ‘ ‘

Example char grade = ‘A’;

**String** a string in java is an object that holds a sequence of characters contained within a pair of double quoted

**Print** outputs all text on the same line

**Println**, short for print line outputs text in a new line

**System.out.println**  outputs everything on the same line

Example System.out.println(“hello world”);

The result is that the program prints Hello World.

**A** **class** is a blueprint or template for an object in Java

**Methods** are reusable pieces of code in classes. The difference between a method and a function is that methods are always related to a class or an object.

Syntax inside of curly place is part of the class

Example public class HelloYou

String[]args is a placeholder for information we want to pass into our program

Every Java program must have a method called **main().** A method is a sequence of tasks for the computer to execute. This main() method holds all of the instructions for our program.

For single-line comments use //

Example // calculate customer satisfaction rating

For multi-line comments use /\* /\*

Example/\*

We chose to store information across multiple databases to  
Minimize the possibility of data loss. We'll need to be careful  
To make sure it does not go out of sync!

\*/

Javadoc comments are represented by /\*\* and \*/

Example

/\*\*

\* The following class accomplishes the following task...

\*/

**Semicolons** are used to mark the end of a statement

**Curly braces** denote the scope of our [classes](#class) and [Methods](#Methods). There are no semicolons at the end of a curly brace.

**ls** is short for list and this command lists all the available files

**javac** Is used to compile code in java

java programs have at least one class and one main() method

whitespace is for humans to read code easily

the main() method runs the tasks of the program

each class represents one real-world idea

example of **main** method public static void main (String[] args){

**camel case** where every world after the first is capitalized Example tipCaluclator

literal is when code is used to represent a fixed value

in java when you don’t put quotation marks around a word

Example string cat = ”Buffy”;

System.out.println(“cat”); prints cat

System.out.println(cat); prints Buffy

**Escape sequences**

**\” escape sequence** allow us to add quotation marks “ to a String value. Example System.out.println("\"Hello World\"");  
// Prints: "Hello World"

**\\ escape sequence** allows us to place backslashes In our String text EXAMPLE System.out.println("This is the backslash symbol: \\");  
// Prints: This is the backslash symbol: \

**/n escape sequence** in a String the complier will output a new line of text. Example System.out.println("Hello\nGoodbye");  
/\*  
Prints:  
Hello  
Goodbye  
\*/

**++** equals an increment of 1

**--** equals a decrease of 1

\* equals multipcation

/ equals division operator

% gives us the remainder after two numbers are divided

Example int cookiesBaked = 10;  
int cookiesLeftover = 10 % 3;  
//cookiesLeftover holds 1

compound assignment operators perform an arithmetic operation on a variable and then reassign its value

+ equals addition can be to concatenate string.

Example String username = "PrinceNelson";n

The code will print Your username is: PrinceNelson

+= for addition

Example int numCookies = 17;

numCookies +=8; changes

changes numCookies to 25

for subtraction -=

for multiplication \*=

for division /=

for modulo %=

PEMDAS 1.parenthesis, 2. exponents, 3. modulo/multipcation/division 4. Addition/subtraction

= one equal sign to we assign values

== two equal signs tells us if two variables are equal

!= To check if two variables are not equal

>= is greater than or equal to

<= is less than or equal to

.equals() is better to use than == to compare objects

Example System.out.println(person1**.equals(person2)**);

**Final** is used when declaring a variable to make it so the value cant be changed

Example to use it in when declaring a foot because it will always equal 12 inches

**Constructors** are like normal methods within the class but are used to initialize the object of the class

Example of a contrsuctor

 public Car(String carColor, int milesPerGallon) {  
    color = carColor;  
    mpg = milesPerGallon;  
  }

reference data type. This means that the value of our variable is a reference to an instance’s memory address. During its declaration, the class name is used as the variable’s type.

We use **new** to indicate that we are creating an instance.

You can use **null** to initialize a object without assigning it a value

**NullPointerException**  is an [exception](https://www.codecademy.com/resources/docs/general/error) that occurs when attempting to use a null value in place of where an object is required.

**instance**[**variables**](https://www.codecademy.com/resources/docs/java/variables)are described as a “has-a” relationship with the object. For example, a Car “has-a” color

**Fields** are a type of state each instance will possess. One instance may have "red" as its color, another "blue"

***A*formal parameter***specifies the type and name of data that can be passed into a method*

**An actual parameter,** or argument, refers to the value being passed during a method call.

in Java, because of **constructor overloading**, a class can have multiple [constructors](https://www.codecademy.com/resources/docs/java/constructors) as long as they have different parameter values.

Call by value is the process of calling a method with an argument value.

**procedural abstraction**: knowing what a method does, but not how it accomplishes it.

We mark the domain of this task using curly braces: {, and }. Everything inside the curly braces is part of the task. This domain is called the **scope of a method.**

**Calling a method***:* Methods are invoked with a . and ()

**Parameters** : Inputs to the method and their types are declared in parentheses in the method signature

Changing Instance Fields : **Methods** can be used to change the value of an instance field

**Scope** : [Variables](https://www.codecademy.com/resources/docs/java/variables) only exist within the domain that they are created in

**Return** : The type of the variables that will be [output](https://www.codecademy.com/resources/docs/java/output) are declared in the method declaration

|| equals or