Java Naming Convention

1. - Variable Naming: Variables should start with a lowercase letter and subsequent words should start with an uppercase letter (e.g. **firstName**).
2. Class Naming: Classes should start with an uppercase letter and subsequent words should start with an uppercase letter as well (e.g. **Person**).
3. Method Naming: Methods should start with a lowercase letter and subsequent words should start with an uppercase letter (e.g. **getFirstName**).
4. Constant Naming: Constants should be named in all uppercase letters using underscore to separate words (e.g. **MAX\_HEIGHT**).
5. Package Naming: Package names should be in all lowercase letters and should not contain any underscores or dashes (e.g. **com.example.util**).

Compile Error vs Runtime Error

- every statement in java must end with a semicolon

- when you cannot generate a test.class, it means compile error.

- examples of compile error: not adding a semicolon at the end of statements

- examples of runtime error: attempting to divide by 0, trying to access an array out of bounds

8 primitive data types in java

- integral: byte, short, int, long

- floating: float, double

- boolean

- char

Statically typed vs Dynamically typed

- Java is a statically typed language

- after you assign x = 1 you cannot reassign x = “abc” after.

Boolean Data Type

For boolean data types, in java it is not represented by 1 and 0 like other languages. It is represented by true and false.

Reference Data Type vs Primitive Data Type

- string variable stores the address of the string (reference data type)

- int variable stores a value (primitive data type)

- any instantible class is a reference data type

\*\*\* Important to know the difference and example data types of primitive and reference data types

Content comparison

- For content comparison, the correct way to do it is to use the .equals() method.

- the == checks if the 2 variables are referring to the same object or not. But .equals() compares the contents of the 2 variables.

- if you use == for 2 strings which are reference data types that store addresses, the == operator is comparing the memory address of the 2 objects. hence even if the strings are both assigned the same value, the operation will return false.

- .equals() method is used to compare the values of two objects

How to prompt for input (integer)

Scanner sc = new Scanner(System.in);

System.out.print(“Enter n:”);

Int n = sc.nextInt();

\*\*\*\* You cannot re initialise the same variable

Loops

- If you want something to run 1 or more times, use a do while loop.

- If you want something to run 0 or more times, use a while loop.

- If you want something to run n times, use a for loop

++x vs x++

If you see ++ behind, it will do whatever other operations there are first before the increment of x

If you see ++ in front, x will be incremented first followed by the rest of the operation.

Common Methods

.length() – gives you the length of string

.length() and .length are 2 DIFFERENT THINGS. .length is a property and gives the length of the array. .length() method is used to get the length of a string.

.charAt(<index>) – gives you the char at the index of the string

s1.equals.(s2) – compares s1 and s2 and returns a true if they are equal and false if not. the equivalent of strcmp() in C

.split(<regex>, limit) to split up a string based on the regex.

IMPORTANT!!!!

- When you are comparing primitive data types, use ==

- For reference data types, when you want to check for content equality, make use of .equals

- if you want to use .split and put the regex as 1 or more whitespace use .split([\\s+](file://s+))

- you cannot pass in an int for a char parameter in a method

- when a string is assigned the value null and you try and print it out, null will be printed

Primitive data type variables initialisation:

In Java, when no value is explicitly assigned to a primitive data type, it is automatically initialized to a default value based on its type. The default values for Java's primitive data types are:

* byte: 0
* short: 0
* int: 0
* long: 0L
* float: 0.0f
* double: 0.0d
* char: '\u0000' (the null character)
* boolean: false