Data Types in Java

Primitive data types:

- Boolean – 1 bit (0/1, true/false)
- Numeric
    - Char – 16 bits (a char is a number, as the decimal has an ASCII character equivalent)
    - Integral
        - Integer
            - Byte – 8 bits (1 byte)
            - Short – 16 bits (2 bytes)
            - Int – 32 bits (4 bytes)
            - Long – 64 bits (8 bytes)
        - Floating-point
            - Float – 32 bits
            - Double – 64 bits

Non primitive data types include:

- Arrays
- Strings
- Classes

Declare

When a variable is first created, it is declared.

Initialise

A variable is initialised if it is given a value, on or after creation.

Instantiate

Instantiating refers to when an object instance is created, via the constructor.

Values vs References

A value is what the variable contains. A reference is the location of the variable as a whole. It is important to know the difference when writing functions..

```c
// Takes: value
// Action: changes the variable v to 2
int changeByValue(int v) {
    v = 2;
}

// Takes: pointer
// Action: dereferences the pointer to alter the original variable (a)
int changeByReference(int* v) {
    *v = 2;
}

int main()
{
    int a = 3;
    //changeByValue(a); // prints 2
    changeByReference(&a); // prints 3

    printf("%i", a);

    return 0;
}
```

Passing by value will only allow a local value to be accessible to the function via parameters. Passing by reference will give the function access to the original object in memory.

C - Pointers

A pointer is a variable which stores the memory address of another variable. You declare a pointer by using the asterisk * before the variable name. You can use the ampersand & to access the address of the variable which follows.

```c
int a = 3;
int *b = &a;
// b now stores the address of a ( & = address-of )

// Access the value of the pointer by dereferencing it first
printf("%i", *b); // prints 3

// to change a via b, dereference the b pointer
*b = 2;

printf("%i", a); // prints 2
```

If a pointer is uninitialised, it will contain a random memory location, and the dereferenced value will be random. This will sometimes crash the program with a

Segmentation fault. It's bad practise to leave a pointer uninitialised. Always give it a value, or specify NULL.

Mutable vs Immutable

A data type is considered immutable if their value cannot be changed after creation. Immutable objects are thread-safe, meaning that multiple threads can try to access the contents of an object, as they wont change after creation.

If an object is immutable, you should create a reference to this object instead of copying the object. This conserves memory, as references are always cheaper than another duplicate object in memory.