Java Syntax Cheat Sheet

Types

A type tells Java what kind of value or object to expect, such as **int**, **double**, **String**, etc.

Variables

A variable is basically a way of giving a name to a value or object, so we can easily use it later!

A variable has a **type**, a **name** and a **value**:
<type> <name> = <value>;

e.g an integer variable:
```
int myNumber = 5;
```

Or, if you had written a class called Cell:
```
Cell myCell = new Cell();
```

Note: Any variable that isn't an int, double or String will need to use the 'new' keyword shown in the example above.

Arrays

An array is basically like a container that can be used to store a collection of values of the same type. It is a little different from other objects in that you have to tell Java **how many items** can be stored inside it.

To declare an array is similar to declaring a normal variable, but you need to use square brackets ('[' and ']':
*<type>***[]** *<name>* = new *<type>***[***<number of values>***];**

e.g for an array storing 5 ints:
```
int[] myArray = new int[10];
```

To store an item in an array, you have to tell Java which slot (called an index) in the array to put it in. These indexes start at **0**, so for the myArray variable above, the indexes go from 0 to 4.

e.g to store an int in the **second** position (index **one**):
```
myArray[1] = 5;
```

Fields

A field is a type of variable that is located **inside the class** but **outside any methods.** Fields are normally declared at the top of the .class file, right after the class declaration (this is the 'public class ClassName' line).

Fields should always be declared as **private**, but otherwise follow the same syntax as normal variables. **Note: It's common to just declare the variable, but to not give it a value straight away:**

```
public class Cell {

    private int sideSize;
    private int row;
    private int col;
```

Java Syntax Cheat Sheet

For Loops

A for loop is a type of loop that lets us run the same bit of code multiple times. This is very useful when working with arrays as we can use a loop to do something to all the values held in an array.

The for loop syntax looks like this:

```
for (int num = 0; num < 10; num++) {
    // Do something here, e.g
    result = result + num;
}
```

What this means in English is while the value of num (initially 0) is less than 10, run the code inside the for loop. After each time the code is executed, 'num++' increases the value of num by 1.

Methods

Methods are a good way to reuse code, so we don't have to type it out every time the code is needed!

A method is made up of one or more **modifiers**, a **return type**, a **name**, and (optionally) **parameters. Note: Even if your method doesn't have any parameters, you still need the brackets!:**

```
public void myMethod () {
     // Method code goes here
}
```

- Modifier: This tells Java how to access the method, common modifiers are **public, private** and **static.**
- Return type: Tells us what type of value will be returned, e.g int, String. **void** means the method doesn't return any value.
- Name: The name of the method
- Parameters: This allows you to pass information into the method. Parameters have the form <type> <name>.

An example of a method that takes an integer parameter, multiplies it by 5 and returns the value is shown below:

```
public int timesByFive(int number) {
     int newNumber = number * 5;
     return newNumber;
}
```