JAVA NOTE

Class

1. Class is a `type` as well as a `module`
2. Class contains fields and methods

Data Type

1. Comment Using:

- `/* */` or `//`

2. Declaration:

- Array:

```
int [] arr; //declare
arr = new int [3] // allocate size
arr[0] = 5, arr[1] = 8, arr[2] = 7 //fill
int [] arr2 = { 5, 8, 7 } // all in one
```

Encapsulation

1. Access Level

Modifier	Class	Package	Subclass	World
Public	Y	Y	Y	Y
Protected	Y	Y	Y	N
No Modifier	Y	Y	N	N
Private	Y	N	N	N

2. Getters:

- It can be used to make a field read-only

3. Robustness:

- Object can guarantee consistency properties (invariants)

4. Constructor

- a function constructs a class object
- `new class_name ()`
- it can be made explicit and adapted to suit class

Clean Code

1. Comment:

- Go at the top of a `class` or just before a `method`
- Explain what `it is for or how to use it`, not how it works

2. Width:

- Ident 2 or 4 space (with soft tabs)
- Width Limit of 80 characters

3. Others:

- Make methods `so short`, `readable` and `obvious`

Unit Testing

1. approach: one-line assertions in the class itself

Multiple Classes

1. It's good for the `main class` to be tiny.

Equality

1. Object is a structure accessed via a ponter Ex.1 :

```
Counter c1 = new Counter ();
 Counter c2 = c1;
 if (c1 == c2 )..... // True
 c1.tick(); // c2 also changes
```

```
Counter c1 = new Counter ();
 Counter c2 = new Counter ();
 if (c1 == c2 )..... // false
```

Ex.2 :

```
String s = "";
 if (s == " ")... // False! (probably!)
 if (s.equals (""))...// True
```

Primitive Types V.S Reference Types

- Primitive
    - byte, short, int, long, float, double, boolean, char
- Reference

    - `Dog myDog = new Dog()`

Garbage Collection (Heap)

Pass-by-Copy

Instance and Local

Generality

- List xs = new List();
- `@SuppressWarnings("unchecked")`

Inheritance

- Object class
- Direct Inheritance
- Parent/base class
    - Overloading Java can sort out by context and handle method with the same of arguments
- subclass
    - super()
    - override()
    - ex

```
public class Doctor {
        ...
}
public class Family Doctor extend Doctor{
        ...
}
```