OOP

→ Class

- A class is a template for an object and
  An objects is an instance of a class
- A class creates a new data type that can be used to create objects

- When ~~Creating~~ declaring an object of a class we are creating an instance of that class
- A class is a Logical construct
- An object has physical reality
  (ie an object occupies space in memory)

objects are characterized by three different essential properties:

State - It is the value of from its datatype

The Identity of an object distinguishes one object from another.
Object Identity = the place where its value is stored in memory.

The behaviours of an object is the effect of data types operations

- The dot operator links the name of the object with the name of an instance variable

The formal Specification for Java categorizes the '.' as a separator.

- The 'new' keyword dynamically allocates (ie allocates at run time) memory for an object & returns a reference to it.

This reference is more or less, the address in memory of the object allocated by new

This reference is stored in the variable

- Thus in Java, all class objects must be dynamically allocated.

```
Box mybox; // declare reference variable

mybox = new Box(); // allocates a Box object
```

- The first line declares mybox as a reference to an object of type Box
- Currently mybox does not refer to an actual Box.
- The next line allocates an object and assigns a reference to it.
- After second line, we can use mybox as if it were a Box object.

In reality

mybox simply holds in essence, the memory address of the actual Box object.

Note -

The keys to Java safety is that we cannot manipulate references as we can actual pointers

Thus we cannot cause an object reference to point to an arbitrary memory location or manipulate it like an integer.

New:

Classname Class_var = new classname();

→ Class_var is a variable of the class type being created.

→ The Class_name is the name of the class that is being instantiated. The class name followed by parenthesis specifies the constructor for the class.

→ A constructor defines what occurs when an object ~~is created~~ of class is created.

But, why don't we need to use new for such things as integer or characters?

Java's primitive types are not implemented as objects

Rather they are implemented as "normal" variable

This is done in the interest of efficiency.

Important

'New' allocates memory for an object during run time.

```
Box b1 = new Box();
Box b2 = b1;
```

b1 & b2 both refer to the same object.
The assignment of b1 & b2 did not allocate any memory or copy any part of the original object.
It simply makes b2 refer to the same object as does b1.
Thus, any changes made to the object through b2 will affect the object to which b1 is referring, since they are the same object.

When we assign one object reference variable to another object reference variable, we are not creating a copy of the object, we are only making a copy of the reference.

```
int square(int i){
    return i*i;
}
```

A parameter is a variable defined by a method that recives a value when the method is called.
For example.
int square(int i) , i is a parameter.
An argument is a value that is passed to a method when it is invoked.

For example
~~int~~
Square(100) , passes 100 as an argument.
Inside square() , the parameter i recives the value.

Note:

Bus bus = new Bus();
LHS (reference ie bus) is looked by the compiler
&
RHS (object ie new Bus()) is looked by JVM