Constructors

If you closely look at following lines of code:

```
Scanner scn = new Scanner(System.in);
Student st1 = new Student();
```

The bold lines look like functions. If is actually a function. It is a special type of function called Constructor.

The job of the constructor is to allocate memory to the object and create it in the heap.

The default constructor creates object and gives default values to the data members

```
public Student () {

}
```

The access specifier can be either public or private or default. The constructor doesn't have any return type and its name is the same as that of the class name.

Parameterised Constructor

Apart from the default constructor, we can give custom values as arguments and those arguments can be used to initialise the values of data members.

Note: 1. Java gives you a default constructor, even if you don't write it.
2. If you write your own parameterised constructor, then Java deletes its default constructor.
3. You can have multiple custom constructors.

Static and Final

static keyword: There are instances when we have data members which are property of the whole class or of all the objects instantiated by class [common to all objects of class]. For example if we want to count the number of student objects created by student class, then we can keep a static variable called num_students and increment it in the constructor.

static functions: Functions which don't depend on objects. Functions which can be called by the class name. For example: getnumstudents(): int

final keyword: There would be some data members whose value you would not like to change. For example, we want to say that the roll number for a student is fixed and once allotted, it will never change. We can use the final keyword for the same. Once initialised, the value of the final variable can never change.

final variables can be initialised in the constructor, as constructor is executed only once.

this is a keyword used in member functions to access local variables with the same name as data members

this refer to the current object to which function is called

Components of OOP

Main 3 components are:

1. Encapsulation and Abstraction
2. Inheritance
3. Polymorphism

Inheritance and Polymorphism

Inheritance: Common data members and member functions in 2 classes. For example: there is a Vehicle class and Car class. There will be a lot of member functions and data members in the Vehicle class which can be used in Car class. So the main objective of Inheritance is code reuse.

We can define a class parent and another class as child and all the properties of parent class will come in the child class