# Constructors function

- Used to create an instance and initialise the instance variable
- Name is same as class name
  - easy to identify in class, suppose class have 100 methods, we can easily tell which one is constructor via its name
- Does not have any return type
  - Implicitly java will add return type of class
  - to differentiate constructor from normal methods
- Cannot be static, final, or abstract
  - constructor's are not inherited, child class does not have access to constructor
  - static method can only access static variable, if we make constructor static, constructor will not be able to initialise instance variables
    - won't be able to use super()
- 'new' keyword tells java, we need to call constructor
- Though any method can also have same name as class name, this is not good practice
- Cannot be defined in an interface, in interface we can not create an object

**Types of Constructor**

1. Default - when no constructor is defined
   - set default values for all instance variables
2. No argument constructor
   - same as default, but manually defined by programmer
3. Parameterised Constructor
   - constructor which takes parameters
4. Whenever we manually define any constructor, default constructor will not be added
5. Constructor Overload
   - Multiple constructor with same name and different parameters
6. Constructor cannot be overridden
   - Because it can not be inherited
7. Private Constructor
   - No one is allowed to call constructor or create an object outside class

**Constructor Chaining**

- this : calling constructor inside another constructor
- super: from parent to child, constructors are called
  - super() is hidden, if we do not manually invoke it
  - however, if we have parameterised parent constructor, super need to be invoked