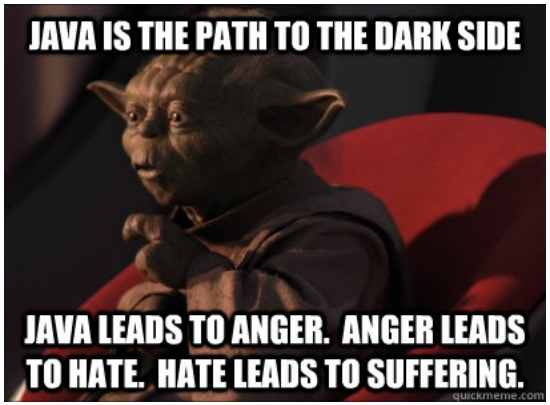
Object orientated programming

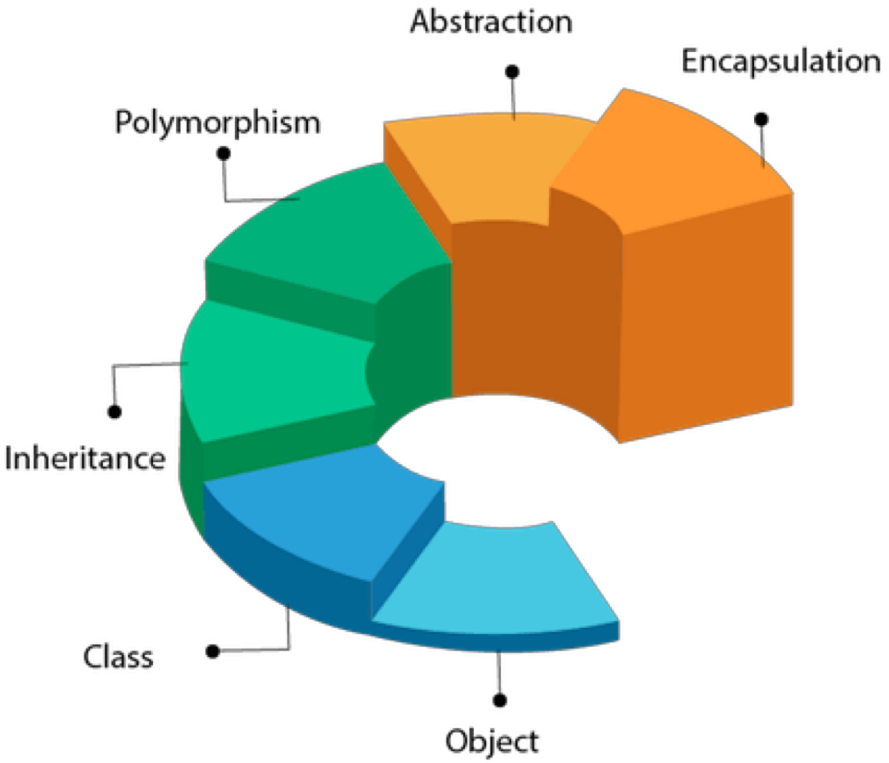
Constructor, static, this



…and suffering leads to skill ☺

Java OOPs Concepts

- Object-Oriented Programming is a paradigm that provides many concepts such as inheritance, data binding, polymorphism, etc.
  - **Simula** is considered the first object-oriented programming language
  - **Smalltalk** is considered the first truly object-oriented programming language
- Popular object-oriented languages are Java, C#, PHP, Python, C++, etc.
- The main aim of object-oriented programming is to implement real-world entities for example object, classes, abstraction, inheritance, polymorphism, etc.



Java OOPs Concepts

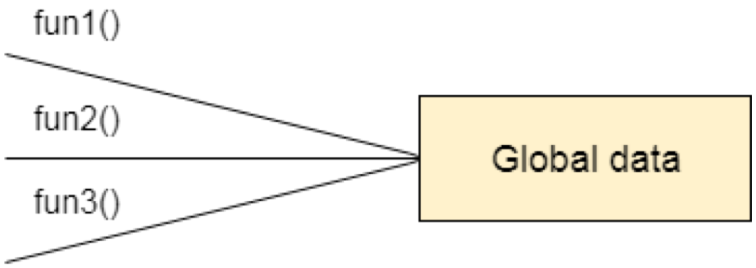
- Class: A collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class does not consume any space on the memory.
- Object: Is an instance of a class. Takes space on the memory and can communicate with other objects knowing the details of each others data or code.
- Inheritance: When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
- Polymorphism: Simply said the same task is performed in different ways. In Java, we use method overloading and method overriding to achieve polymorphism. E.g., animals speak different

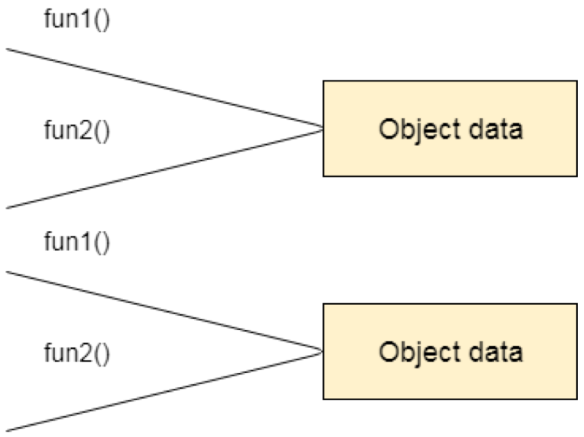
Java OOPs Concepts

- Abstraction: Hiding internal details and showing functionality is known as abstraction. For example for a phone call, we do not know the internal processing (what is actually happening and we do not care).
  - In Java, we use abstract class and interface to achieve abstraction.
- Binding (or wrapping) code and data together into a single unit are known as encapsulation. Can be imagined as a capsule wrapping different medicine.
  - A java class is the example of encapsulation.

Advantage of OOPs over procedure-oriented programming language

- OOP makes development and maintenance easier whereas in a procedure-oriented programming language it is not easy to manage if code grows as project size increases.
- OOP provides data hiding whereas in a procedure-oriented programming language global data can be accessed from anywhere.
- OOP provides the ability to simulate a real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.





Constructors

- A constructor is a block of code similar to a method. It is called when an instance of the object is created, and memory is allocated for the object.
- Basically a special type of method which is used to initialize the object.

When is a constructor called?

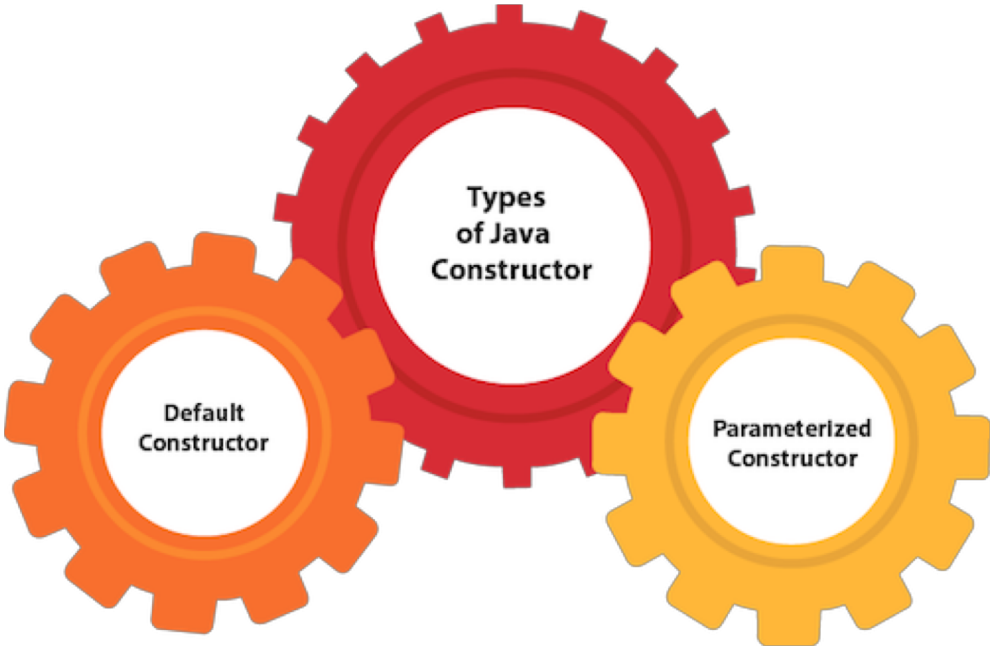
- Every time an object is created using the new() keyword, at least one constructor is called.
  - It calls a default constructor if not specified.
- **Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class (Default constructor).**

Rules for creating a constructor

- There are three rules defined for the constructor:
  - A constructor name must be the same as its class name
  - A constructor must have no explicit return type
  - A constructor cannot be abstract, static, final or synchronized
- **Note: We can use access modifiers while declaring a constructor. It controls the object creation.**
  - In other words, we can have private, protected, public or default constructors in Java.

Types of constructors

- There are two types of constructors in Java
- 1) Default constructor (no-arguments constructor)
- 2) Parameterized constructor



Java Default Constructor

- A constructor is called "Default Constructor" when it does not have any parameters.

```
<class_name>(){}
```

- In the following example, we are creating the no-arguments constructor for a bike class. It is called at the time of object construction/creation (bike1).
- **Rule: If there is no constructor in a class, the compiler automatically creates a default constructor.**

Java default constructor example

- Bike1

```
//Java Program to create and call a default constructor
class Bike1{
//creating a default constructor
Bike1(){System.out.println("Bike is created");}
//main method
public static void main(String args[]){
//calling a default constructor
Bike1 b=new Bike1();
}
}
```

What is the purpose of the default constructor?

- The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.
  - Remember we talked about this before!

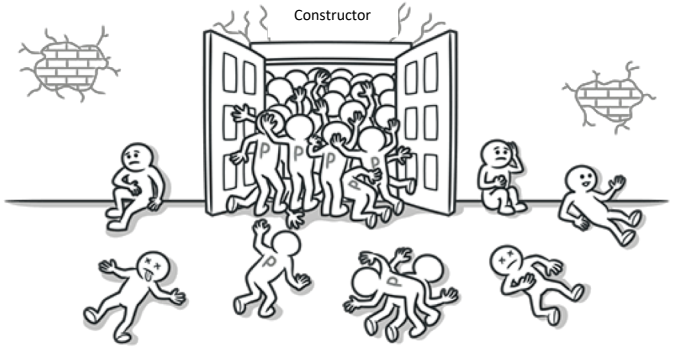
Example of default constructor that displays the default values

- In the example the compiler provides a default constructor. The default constructor provides 0 and null values (Student1).

```
//Let us see another example of default constructor
//which displays the default values
class Student3{
int id;
String name;
//method to display the value of id and name
void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
//creating objects
Student3 s1=new Student3();
Student3 s2=new Student3();
//displaying values of the object
s1.display();
s2.display();
}
}
```

Java parameterized constructor

- Is a constructor which has a specific number of parameters.
- The parameterized constructor is used to provide different values to the distinct objects. However, nothing prevents from providing the same values (will be a new object anyway).
- The next example, we create the constructor for the Student class that has two parameters. There is no limitation for the number or parameters (Student2)!



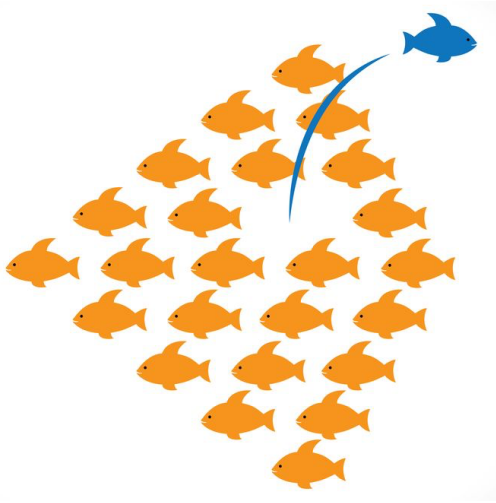
Example of a parameterized constructor

- Student2

```
//Java Program to demonstrate the use of parameterized constructor
class Student4{
    int id;
    String name;
    //creating a parameterized constructor
    Student4(int i,String n){
    id = i;
    name = n;
    }
    //method to display the values
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    //creating objects and passing values
    Student4 s1 = new Student4(111,"Karan");
    Student4 s2 = new Student4(222,"Aryan");
    //calling method to display the values of object
    s1.display();
    s2.display();
   }
}
```

Constructor overloading

- In Java, a constructor is just like a method but without a return type. It can also be overloaded like methods.
- Constructor overloading in Java is a technique of having more than one constructor with different parameter lists.
- They are arranged in a way that each constructor performs a different task.
- They are differentiated by the compiler by the number of parameters in the list and their types.



Example of constructor overloading

- Student3

```
//Java program to overload constructors in java
class Student5{
   int id;
   String name;
   int age;
   //creating two arg constructor
   Student5(int i,String n){
   id = i;
   name = n;
   }
   //creating three arg constructor
   Student5(int i,String n,int a){
   id = i;
   name = n;
   age=a;
   }
   void display(){System.out.println(id+" "+name+" "+age);}

   public static void main(String args[]){
   Student5 s1 = new Student5(111,"Karan");
   Student5 s2 = new Student5(222,"Aryan",25);
   s1.display();
   s2.display();
  }
}
```

Difference between constructors and methods in Java

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as class name.

Copy constructor

- In C++ we can copy the values from one object to another with copy constructor.
- There is no copy constructor in java.
- Although there are many other ways to copy the values of one object into another in java.
  - With a constructor
  - By assigning the values of one object into another
  - By the clone() method of the Object class
- In the next example we are going to copy the values of one object into another using a constructor.

Copy constructor example

- Student4

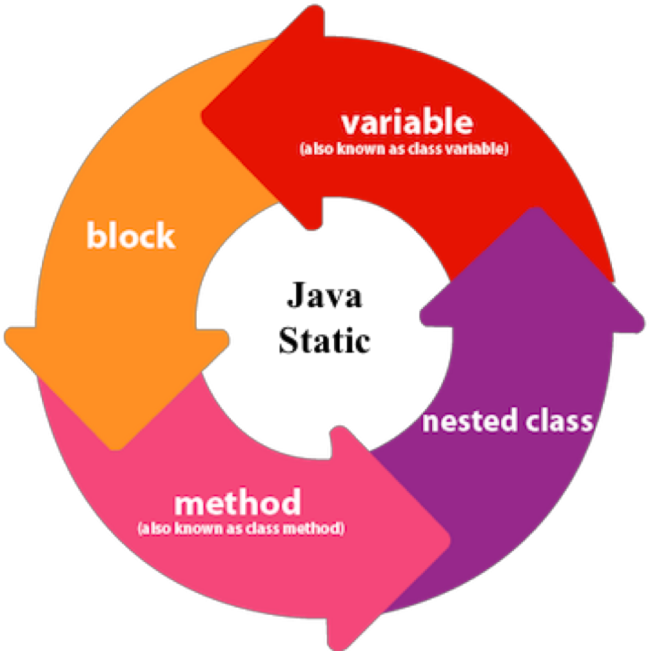
```
//Java program to initialize the values from one object to another
class Student6{
    int id;
    String name;
    //constructor to initialize integer and string
    Student6(int i,String n){
    id = i;
    name = n;
    }
    //constructor to initialize another object
    Student6(Student6 s){
    id = s.id;
    name =s.name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student6 s1 = new Student6(111,"Karan");
    Student6 s2 = new Student6(s1);
    s1.display();
    s2.display();
   }
}
```

Copying values without constructor

- We can also copy the values of one object into another by assigning the objects values to another object. In this case, there is no need to create the constructor (Student5).
- Q) Does constructor return any value?
  - Yes, it is the current class instance (cannot use return type yet it returns a value).
- Q) Can constructor perform other tasks instead of initialization?
  - Yes, like object creation, starting a thread, calling a method, etc. You can perform any operation in the constructor as you perform it in the method.

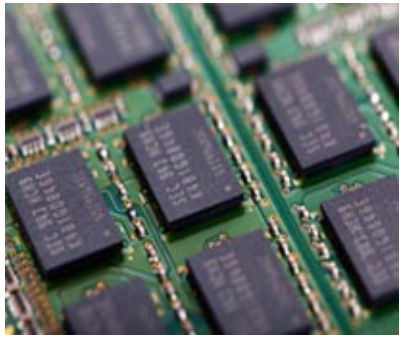
Java static keyword

- The **static keyword** in Java is mainly used for memory management. We can apply java static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class and not to an instance of the class.
- Static can be:
  - A variable (also known as a class variable)
  - A method (also known as a class method)
  - A block
  - A nested class



Java static variable

- If you declare any variable as static, it is known as a static variable.
- The static variable can be used to refer to the common property of all objects (which is not unique for each object).
  - For example, the company name of employees, college name of students, etc.
- The static variable gets memory only once in the class area at the time of class loading.
- Advantages of static variable
  - It makes your program **memory efficient** (i.e., it saves memory).



Understanding the problem without static variable

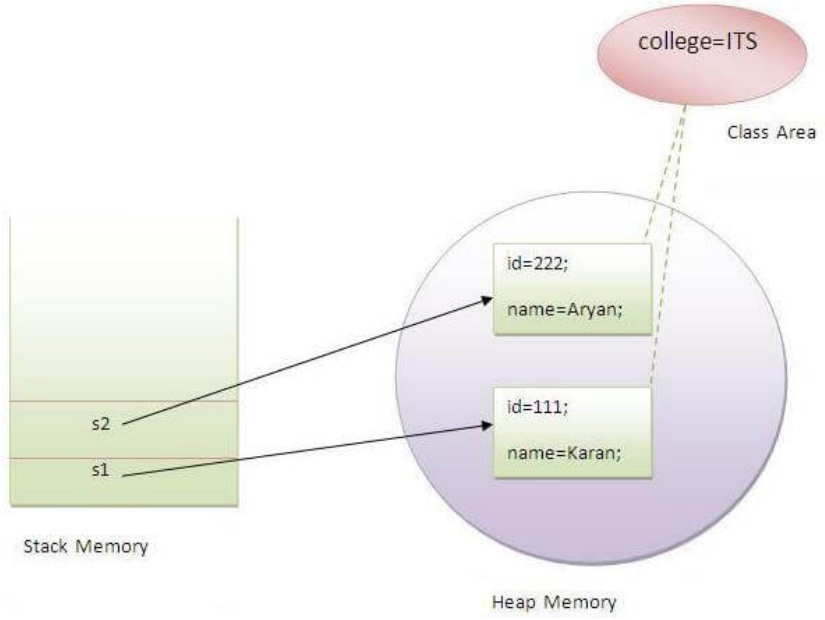
- We can copy the values of one object into another by assigning the objects values to another object. In this case, there is no need to create a constructor.
- Suppose there are 500 students in a college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name. Instance data members are necessary for these two.
  - But "college" refers to the common property of all objects. If we make it static, this field will get memory assigned only once.
- Java static property is shared to all objects.

```
class Student{
    int rollno;
    String name;
    String college="ITS";
}
```

Example of static variable

```
//Java Program to demonstrate the use of static variable
class Student{
   int rollno;//instance variable
   String name;
   static String college ="ITS";//static variable
   //constructor
   Student(int r, String n){
   rollno = r;
   name = n;
   }
   //method to display the values
   void display (){System.out.println(rollno+" "+name+" "+college);}
}
//Test class to show the values of objects
public class TestStaticVariable1{
 public static void main(String args[]){
 Student s1 = new Student(111,"Karan");
 Student s2 = new Student(222,"Aryan");
 //we can change the college of all objects by the single line of code
 //Student.college="BBDIT";
 s1.display();
 s2.display();
 }
}
```

- TestStaticVariable1



Program of a counter without static variable

- In this example, we have created an instance variable named count which is incremented in the constructor. Since the instance variable gets the memory at the time of object creation, each object will have a copy of the instance variable.
- If it is incremented, it will not reflect other objects. Therefore, each object would have the value 1 in the count variable (Counter).

Program of a counter using static variable

- As mentioned before, a static variable will get the memory assigned only once.
- If any object changes the value of the static variable, it will retain its value.
- Example Counter2

Static method

- The static keyword applied with any method is known as static method.
- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access a static data member and can change the value of it.
- Two examples: students and a static method that performs a normal calculation (TestStaticMehtod and Calculate)

Restrictions for the static method

- There are two main restrictions for the static method.
  - The static method cannot use non static data members or call non-static methods directly (Example A).
  - this and super cannot be used in static context.

- Why is the Java main method static?
- Because an object is not required to call a static method. If it would be a non-static method, JVM creates an object first then call the main() method which would lead to the problem of extra memory allocation.

Static block

- Is used to initialize the static data members.
- It is executed before the main method at the time of class loading (Example A2).

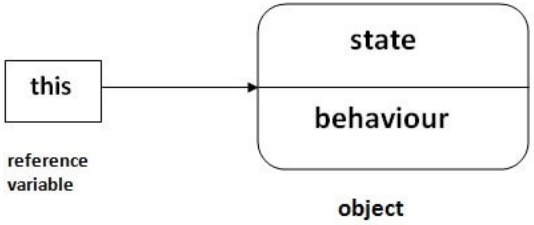
- Can we execute a program without main() method?
- No, one of the ways was the static block, but it was only possible until JDK 1.6. Since JDK 1.7, it is not possible to execute a java class without the main method (Example A3).

```
class A3{
  static{
  System.out.println("static block is invoked");
  System.exit(0);
  }
}
```

Will not work!

this keyword in java

- The this keyword in jave can be used in several different ways. In java, **this** is a **reference variable** that refers to the current object.



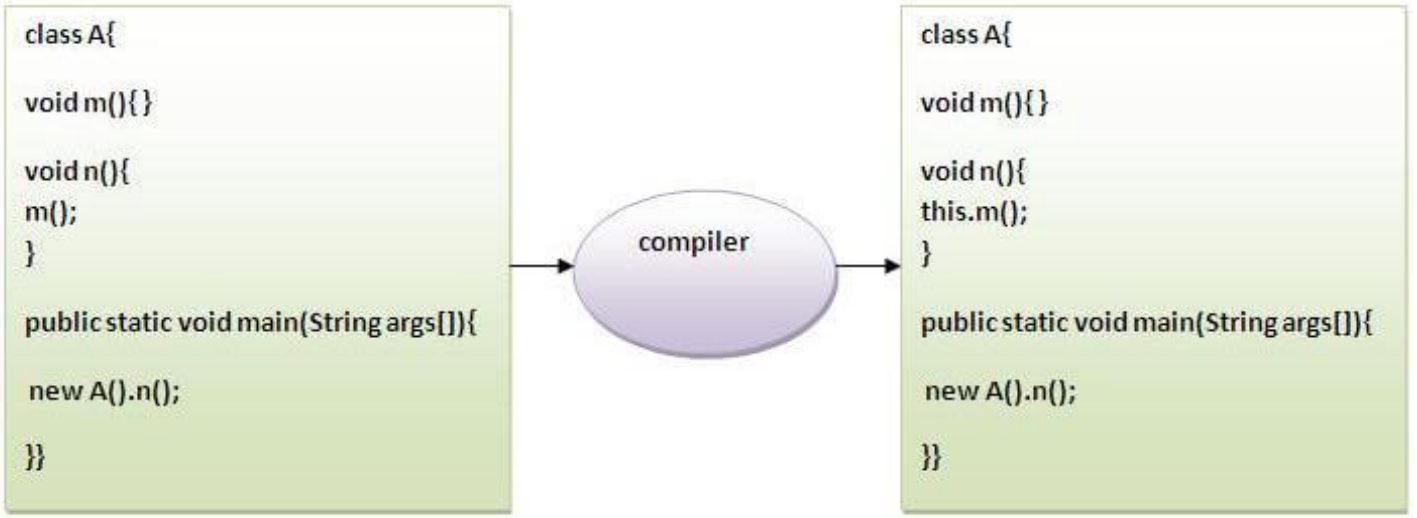
1) this: to refer to a current class instance variable

- The this keyword can be used to refer to a current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.
- Understanding the problem without this keyword
- Let's understand the problem if we don't use this keyword by the example (TestThis1)
- In the example, parameters (formal arguments) and instance variables are the same. We are not using the this keyword to distinguish local variables and instance variables.
- Solution of the above problem by the this keyword (TestThis2)
- If the local variables (formal arguments) and instance variables are different, there is no need to use the this keyword like in the next example (TestThis3, program where this keyword is not required)

It is better approach to use meaningful names for variables. So we use same name for instance variables and parameters in real time, and always use this keyword.

2) this: to invoke a current class method

- A method of the current class can be invoked by using the this keyword.
- If you do not use the this keyword, the compiler automatically adds this keyword while invoking the method. Example **TestThis4** displays this behavior.



3) this: to invoke current class constructor

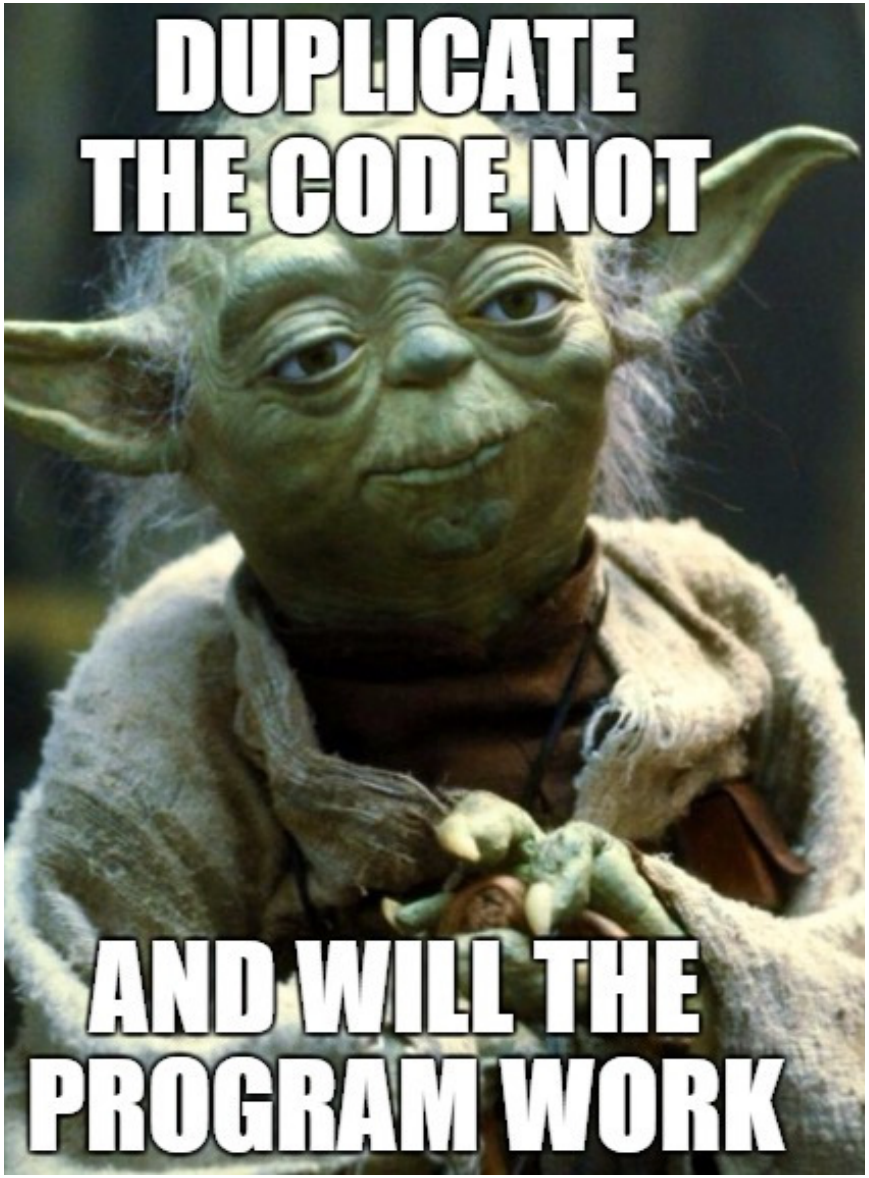
- The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.
- Example 1: **Calling the default constructor from parameterized constructor (TestThis5)**
- Example 2: **Calling a parameterized constructor from default constructor (TestThis6)**
- Example of real world usage of this() constructor call
- The this() constructor call should be used to reuse the constructor from the constructor. It maintains the chain between the constructors i.e. it is used for constructor chaining (TestThis7).
- **Rule: Call to this() must be the first statement in constructor (TestThis8)**

4) this: to pass as an argument in the method

- The this keyword can also be passed as an argument in the method. It is mainly used in the event handling (S2).
- Application of this that can be passed as an argument:
- In event handling (or) in a situation where we have to provide a reference of a class to another one. It is used to reuse one object in many methods.

5) this: to pass as argument in the constructor call

- We can pass the this keyword in the constructor also. It is useful if we have to use one object in multiple classes. Let's see the example (A4)



6) this keyword can be used to return the current class instance

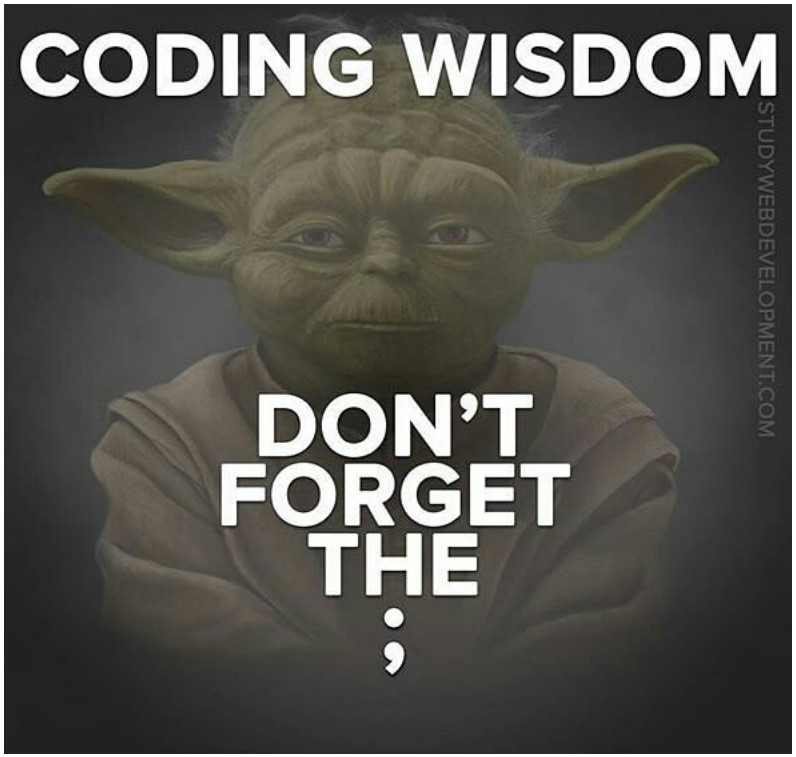
- We can return the this keyword as an statement from the method. In such case, the return type of the method must be the class type (non-primitive).
- Syntax of this that can be returned as a statement

```
return_type method_name(){
return this;
}
```

Example of this keyword that you return as a statement from the method (Test1)

Proving this keyword

- Let's prove that the this keyword refers to the current class instance variable. In the next example program, we are printing the reference variable and this, output of both variables should be the same (A5).



Where are we now?

- Java basics
- Classes and objects
- Constructors
- Static key word
- This key word
- Inheritance
- Aggregation
- Method overloading and overriding in context with OOP
- Polymorphism
- Abstract classes, Interfaces
- Encapsulation
- Exception handling
- …Object class, cloning, java math…
- Mandatory assignment: When and what
  - Groups?
  - How much other things are going on?