Java static keyword

- The **static keyword** in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

- The static can be:
- variable (also known as class variable)
- method (also known as class method)
- block
- nested class

Java static variable

If you declare any variable as static, it is known static variable.

- The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.

Advantage of static variable

- It makes your program **memory efficient** (i.e it saves memory).

Example of static variable

```
class Student8{
   int rollno;
   String name;
   static String college ="ITS";

   Student8(int r,String n){
   rollno = r;
   name = n;
   }
  void display ()
{System.out.println(rollno+" "+name+" "+college)
;}
```

```
public static void main(String args[])
{
 Student8 s1 = new Student8(111,"Karan");
 Student8 s2 = new Student8(222,"Aryan");

 s1.display();
 s2.display();
 }
}
```

Program of counter without static variable

- **class** Counter{
- **int** count=0;//will get memory when instance i s created
-
- Counter(){
- count++;
- System.out.println(count);
- }

```
public static void main(String args[])
{

Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();

 }
}
```

Program of counter by static variable

- **class** Counter2{
- **static int** count=0;//will get memory only onc e and retain its value
-
- Counter2(){
- count++;
- System.out.println(count);
- }

```
public static void main(String args[])
{

Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();

 }
}
```

Java static method

If you apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- static method can access static data member and can change the value of it.

```
class Calculate
{
  static int cube(int x)
{
  return x*x*x;
  }

```

```
public static void main(String args[])
{
  int result=Calculate.cube(5);
  System.out.println(result);
  }
}
```

Restrictions for static method

- There are two main restrictions for the static method. They are:The static method can not use non static data member or call non-static method directly.
- this and super cannot be used in static context.

```
class A
{
 int a=40;//non static

 public static void main(String args[])
{
  System.out.println(a);
 }
}
```

why java main method is static?

- because object is not required to call static method if it were non-static method, jvm create object first then call main() method that will lead the problem of extra memory allocation.

Java static block

- Is used to initialize the static data member.
- It is executed before main method at the time of classloading.

```
class A2
{
  static
{
System.out.println("static block is invoked");
}
  public static void main(String args[])
{
   System.out.println("Hello main");
  }
}
```

Can we execute a program without main() method?

- ?????

- Yes, one of the way is static block but in previous version of JDK not in JDK 1.7

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

  System.exit(0);
  }
}
```

this keyword in java

- There can be a lot of usage of **java this keyword**. In java, this is a **reference variable** that refers to the current object.

Usage of java this keyword

- this can be used to refer current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.
- this can be passed as an argument in the method call.
- this can be passed as argument in the constructor call.
- this can be used to return the current class instance from the method.

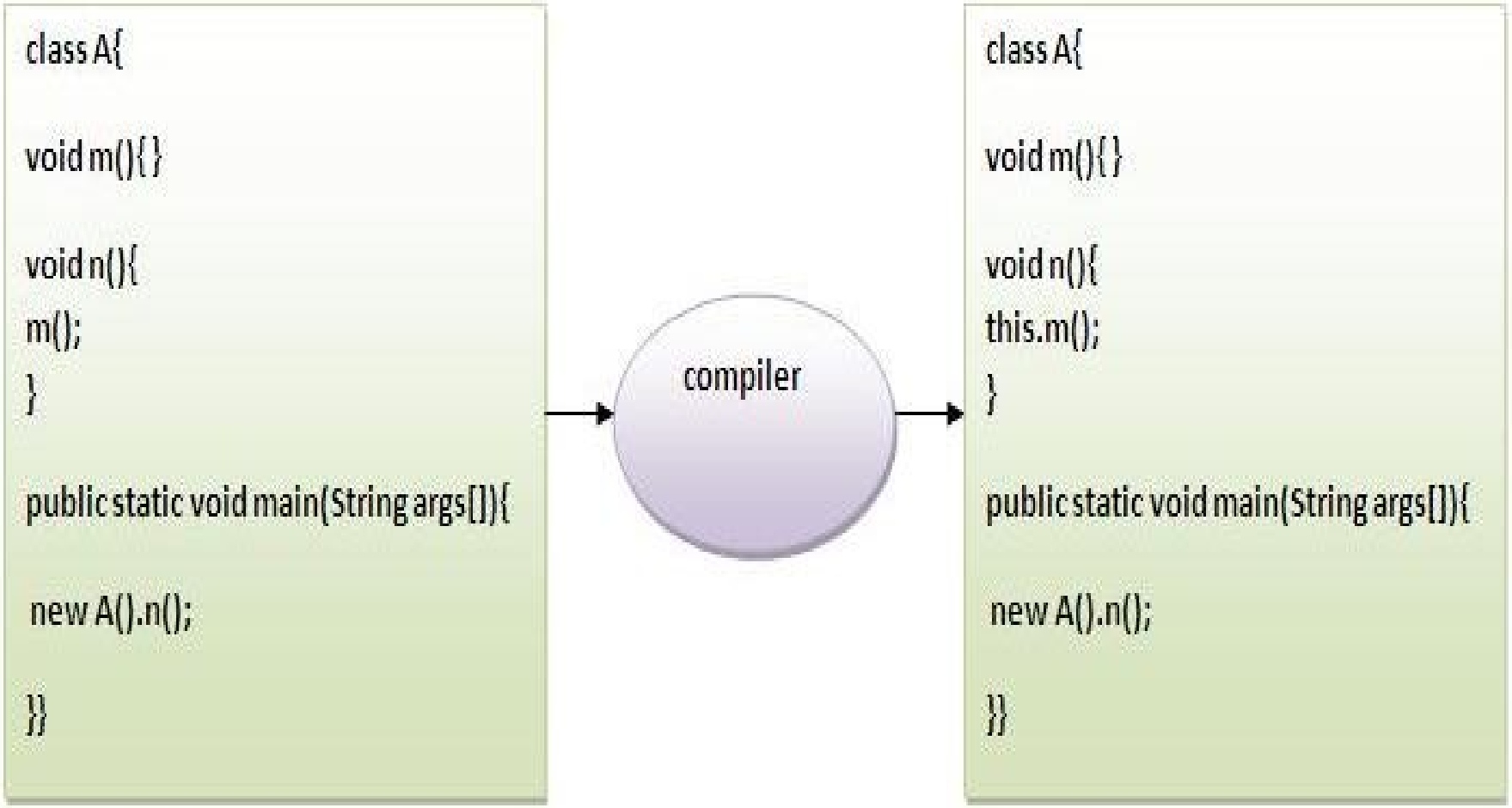
this: to refer current class instance variable

- The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
this.rollno=rollno;
this.name=name;
this.fee=fee;
}
void display()
{System.out.println(rollno+" "+name+" "+fee);}
}
```

```
class TestThis2
{
public static void main(String args[])
{
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}
}
```

this: to invoke current class method



```
class A
{
void m(){System.out.println("hello m");}
void n()
{
System.out.println("hello n");
//m();//same as this.m()
this.m();
}
}
```

```
class TestThis4
{
public static void main(String args[])
{
A a=new A();
a.n();
}
}
```

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.

Calling default constructor from parameterized constructor:

```
class A{
A(){System.out.println("hello a");}
A(int x){
this();
System.out.println(x);
}
}
class TestThis5{
public static void main(String args[]){
A a=new A(10);
}}
```

Calling parameterized constructor from default constructor:

```
class A{
A(){
this(5);
System.out.println("hello a");
}
A(int x){
System.out.println(x);
}
}
class TestThis6{
public static void main(String args[]){
A a=new A();
}}
```