static Block

_____

=> As we know that the execution of program is started from main().

=>If main() class contains static block, Then static Block will run before main() method.

=>static block will implicitly invoked before main() by JVM.

=>if a class contains more than 1 static block, then all static block will run before main() from top to bottom.

=>a static block can call only static variable and a variable which is declared inside static block, implicitly declared as static.

=>Basically static block is used to initialize static variable so in other Langaues it is also called static constructor.

Exp

____

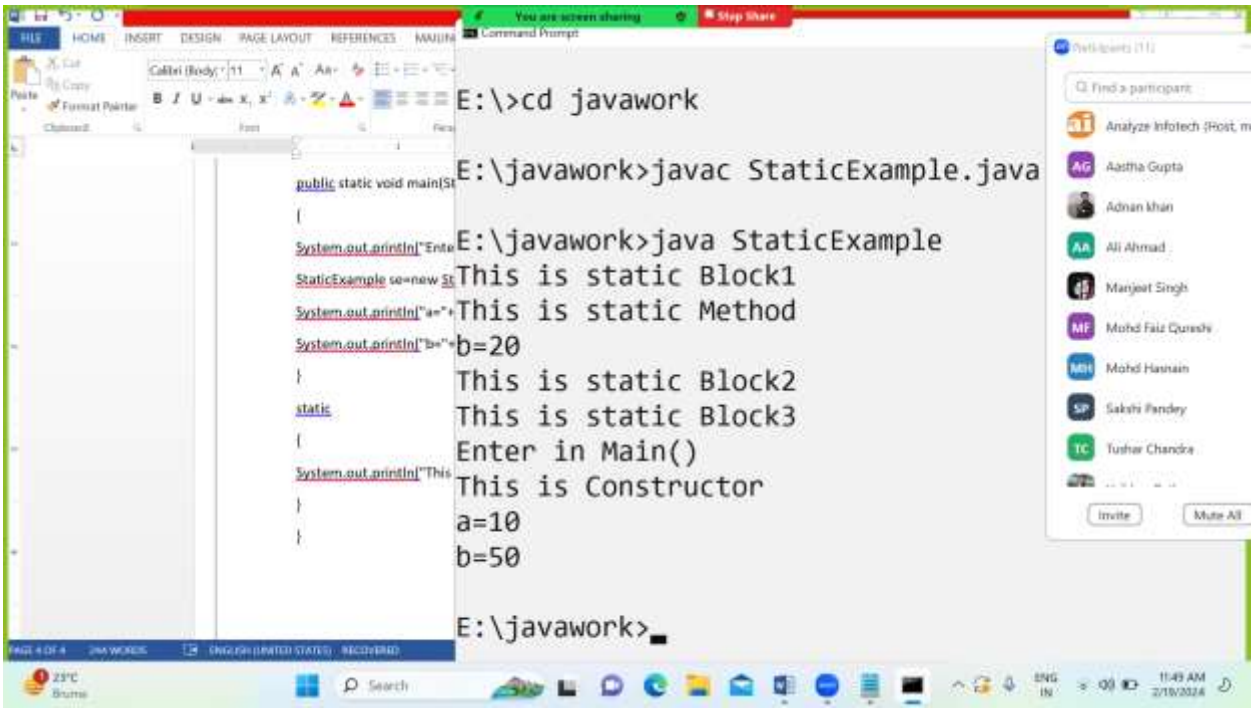
```
class data
{
int a;
static int b;
data()
{
a=10;
b=20;
}
/*static
{
b=20;
}*/
public static void main(String ar[])
{
System.out.println("b="+data.b);//0
```

```
data d=new data();
System.out.println("a="+d.a);//10
System.out.println("b="+d.b); //20
}
}
```

```
class data
{
int a;
static int b;
data()
{
a=10;
}
static
{
b=20;
}
public static void main(String ar[])
{
System.out.println("b="+data.b);//20
data d=new data();
System.out.println("a="+d.a);//10
System.out.println("b="+d.b); //20
}
}
```

```
//Program to Demonstration of static Block
class StaticExample
{
int a;
static int b;
StaticExample()
{
a=10;
b=50;
System.out.println("This is Constructor");
}
static
{
b=20;
//a=30; Error because a is non static
System.out.println("This is static Block1");
}
static void staticMethod()
{
System.out.println("This is static Method");
System.out.println("b="+b);
}
static
{
staticMethod();
System.out.println("This is static Block2");
```

```
}
public static void main(String arg[])
{
System.out.println("Enter in Main()");
StaticExample se=new StaticExample();
System.out.println("a="+se.a);
System.out.println("b="+se.b);
}
static
{
System.out.println("This is static Block3");
}
}
```



Ques : Can we execute a program without main()

ANS : upto JDK1.6 using static block and without main(), we can execute a program, but JDK 1.7 and upper version does not allow to execute a program without main().

```
class Test
{
static
{
int a=10,b=20,c;
c=a+b;
System.out.println("Addition="+c);
System.exit(0);
}
}
```

static class

_____

=> An outer class can not be static

=> An Inner class can be static, Then it is called Nested class

=> An inner class required to activate explicitly and a static class is implicitly activated when the object of its outer class is created

_____

Question : what is diff. between inner and nested class.

```
class ABC
{
static class UVW //Nested Class
{
}
class XYZ //Inner Class
{
}
}
```

_____

WAP to demonstrate the difference between nested and inner class.

_____

final Modifier

_____

1. final Variable
2. final Method
3. final class

1. final Variable

_____

=> a final type variable must be initialize at the time of declaration and once it is initialized , it can not be change.

Example

```
//final int a; Compile Time Error
final int a=20;
//a=50; Error
```

Note : final variable is used to make a variable as constant.

final Method

_____

=> if we have a base class method as final , Then it can not be override.

class Value

```
{
void show()
{
}
final void msg()
{
}
}
class Data extends Value
{
void show()
{
}
/*void msg() Error because final Method can not be Override
{
}*/
}
```

final class

_____

=> final modifier prevents a class to be inherit ie a final class can not pe inherit.

```
final class CompanyABC
{
void display()
{
}
}
/*class CompanyXYZ extends CompanyABC
```

{ Compile Time Error

}*/

Note: final and abstract modifiers have just opposite property so it can not use for same component

---

Interface

=>An Interface is a pure abstract class that also can not have own Objects.

=> The Basic Difference between Abstract class and Interface that an abstract class can have both(declaration and difinition) of Method, but an interface can have only declaration of Methods.

Properties of Interface

---

1. An interface does not contains constructor neither implicit nor explicit.

2. An interface variable implicitly assign 3 modifiers => public, static, final

3. An interface Method implicitly assign 2 modifiers => public and abstract.

---

4. An Interface is declared using interface keyword and it should be implemented by a class using implements keyword.

5. A class that implements an interface, it must override all the methods of interface.

6. To override a method of interface , we must use public modifier.

---

7. A class can extends a class and implements an interface at same time

Example

```
interface Company
{
}

class Department
{
}

class Employee extends Department implements Company
{
}

/*class Employee implements Company extends Department
{
}*/
```

_____

8. A class can implements 2 or more interfaces at same time, it is also called implementation of Multiple Inheritance using Interface.

Example

```
interface Company
{
}
interface Department
{
}
interface EPF
{
}
class Employee implements Company,Department,EPF
{
}
```

Note: Here class Employee must override all the Methods of interface Company,Department,EPF

---

9. An Interface can extends 2 or more interface at same Time.

```
interface UGC
{
}
interface University
{
}
interface College extends UGC,University
```

{

}

```
import java.util.*;
interface college
{
//int a;Error
String c_name="Lucknow University";//public static final are automatically Added
int c_code=1020;//public static final are autometically Added
String city="Lucknow";
void printStudentDetail();
void msg();
/*void show()
{          Error
System.out.println("This is interface college");
}*/
}
class student implements college
{
String s_name,s_course,s_sem;
void inputStudentDetail(){
//c_name="Integral University"; ERROR
Scanner scan=new Scanner(System.in);
System.out.println("Enter Student Name, course, semester");
s_name=scan.next();
s_course=scan.next();
s_sem=scan.next();
}
public void printStudentDetail()
```

```
{                                                    //override
//c_code=2040;Error
System.out.println("College Name\t"+c_name);
System.out.println("College Code\t"+c_code);
System.out.println("student name\t"+s_name);
System.out.println("course      \t"+s_course);
System.out.println("Semester    \t"+s_sem);
}
public void msg()
{//override
}
}
public  class interface1
{
public static void main(String ar[])
{
System.out.println("City     "+college.city);
//college obj=new college(); Error
student stu=new student();
stu.inputStudentDetail();
stu.printStudentDetail();
college col=new student();
//col.inputStudentDetail(); Error
col.printStudentDetail();
}
}
```

```
//multiple inheritance using Interface
import java.io.*;
interface college
```

```
{
String c_name="Integral";
int c_code=1020;
String city="Lucknow";
void printStudentDetail();
void msg();
}
class Department
{
String dname;
int dcode;
void getDept(String s1, int s2)
{
dname=s1;
dcode=s2;
}
}
class student   extends Department implements college
{
String s_name,s_course,s_sem;
void inputStudentDetail()throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter Student Name, course, semester");
s_name=br.readLine();
s_course=br.readLine();
s_sem=br.readLine();
}
 public void printStudentDetail()//overriding
```

```
{
System.out.println("College Name\t"+c_name);
System.out.println("College Code\t"+c_code);
System.out.println("Department Name\t"+dname);
System.out.println("Department Code\t"+dcode);
System.out.println("student name\t"+s_name);
System.out.println("course      \t"+s_course);
System.out.println("Semester    \t"+s_sem);
}
public void msg()
{//override
System.out.println("This is Lucknow");
}
}
public class interface2
{
public static void main(String ar[])throws IOException
{
System.out.println(college.c_name);
student stu=new student();
stu.inputStudentDetail();
stu.getDept("Compter Science",1002);
stu.printStudentDetail();
}
}
```

```
//Program to Demonstration of  an interface can extends another interface
interface University
{
```

```
String uname="AKTU";
String city="Lucknow";
void putuniversity();
}
interface UGC
{
String ugc="University Grant Commision";
void show();
}
interface College extends University,UGC // Multiple Inheritance
{
String cname="IET";
String city="Lucknow";
void putcollege();
}
class Student implements College
{

String sname,scourse;
static void display()
{
System.out.println("Hello");
}
void getstudent(String str1,String str2)
{
sname=str1;
scourse=str2;
}
void putstudent()
```

```
{
Student.display();
System.out.println("Student Name\t"+sname);
System.out.println("Course\t"+scourse);
}
public void putuniversity()
{//overriding
System.out.println("University Name\t"+uname);
System.out.println("City\t"+city);
}
public void putcollege()
{
System.out.println("College Name\t"+cname);
System.out.println("City\t"+city);
}
public void show()
{
}
}
class interface3
{
public static void main(String ar[])
{
Student stu=new Student();
stu.getstudent("James","MCA");
stu.putuniversity();
stu.putcollege();
stu.putstudent();
}
```

```
}
```

```
//Program to Demonstration of  a class can implements more than 1 Inerfaces(Multiple Inharitance
interface Company
{
String cname="ANALYZE";
String city="Lucknow";
void putcompany();
}
interface Department
{
String dname="SD";
String dcode="ai1020";
void putdept();
}
class Employee implements Company,Department
{
String ename,epost;
void getemp(String str1,String str2)
{
ename=str1;
epost=str2;
}
public void putdept()
{
System.out.println("Department Name\t"+dname);
System.out.println("DepartmentCode\t"+dcode);
}
public void putcompany()
```

```
{//overriding
System.out.println("Company Name\t"+cname);
System.out.println("City\t"+city);
}
 void putemp()
{
System.out.println("Employee Name\t"+ename);
System.out.println("Post\t"+epost);
}
}
class interface4
{
public static void main(String ar[])
{
Employee emp=new Employee();
emp.getemp("James","SE");
emp.putemp();
emp.putcompany();
emp.putdept();
}
}
```