OBJECT ORIENTED PROGRAMMING

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

STATIC

- When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.
- You can declare both methods and variables to be static.
- Example of a static member is main( ). main( ) is declared as static because it must be called before any objects exist.
- Instance variables declared as static are, essentially, global variables.
- When objects of its class are declared, no copy of a static variable is made. Instead, all instances of the class share the same static variable.

STATIC

- The **static keyword** in Java is used for memory management mainly.
- We can apply static keyword with variables, methods, blocks and nested classes.
- The static keyword belongs to the class than an instance of the class.
- The static can be:
    1. Variable (also known as a class variable)
    2. Method (also known as a class method)
    3. Block
    4. Nested class

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, 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: makes your program memory efficient .

```
class Student{
    int rollno;
    String name;
    String uni=“IQRA";
}
```

```
class Student{
   int rollno;
   String name;
   static String uni ="IQRA“;
}
```

EXAMPLE

Output:

```
111 Ali abc
222 Tahir abc
```

```
class Student{
int rollno;
String name;
static String uni =“IQRA";

//constructor
Student(int r, String n){
            rollno = r;
            name = n;    }

//method to display the values
void display (){
          System.out.println(rollno+" "+name+" "+uni); } }

public class TestStaticVariable1{
            public static void main(String args[]){
                        Student s1 = new Student(111,“Ali");
                        Student s2 = new Student(222,“Tahir");
                        Student.uni=“abc";
                        s1.display();
                        s2.display(); } }
```

EXAMPLE

Output:

```
1
2
3
```

```
class Counter2{
static int count=0;                //will get memory only once and retain its value
Counter2(){
          count++; //incrementing the value of static variable
          System.out.println(count); }
public static void main(String args[]){
     //creating objects
     Counter2 c1=new Counter2();
     Counter2 c2=new Counter2();
     Counter2 c3=new Counter2(); } }
```

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 static data member and can change the value of it.
- They can only call other static methods.
- They must only access static data.
- They cannot refer to this or super in any way.
- The static method cannot use non static data member or call non-static method directly.
- Outside of the class in which they are defined, static methods and variables can be used independently of any object. To do so, you need only specify the name of their class followed by the dot operator.

  classname.method( )

EXAMPLE

```
class Student{
int rollno;
String name;
static String uni = “IQRA";

//static method to change the value of static variable
static void change(){
uni = “abc"; }

//constructor to initialize the variable
Student(int r, String n){
rollno = r;
name = n;
}

//method to display values
void display(){
System.out.println(rollno+" "+name+" "+uni); } }
```

```
public class TestStaticMethod{

  public static void main(String args[]){
  Student.change();

  Student s1 = new Student(111,“ali");
  Student s2 = new Student(222,“tahir");
  Student s3 = new Student(333,“maria");

  //calling display method
  s1.display();
  s2.display();
  s3.display(); } }
```

Output:

```
111 ali abc
222 tahir abc
333 maria abc
```

EXAMPLE

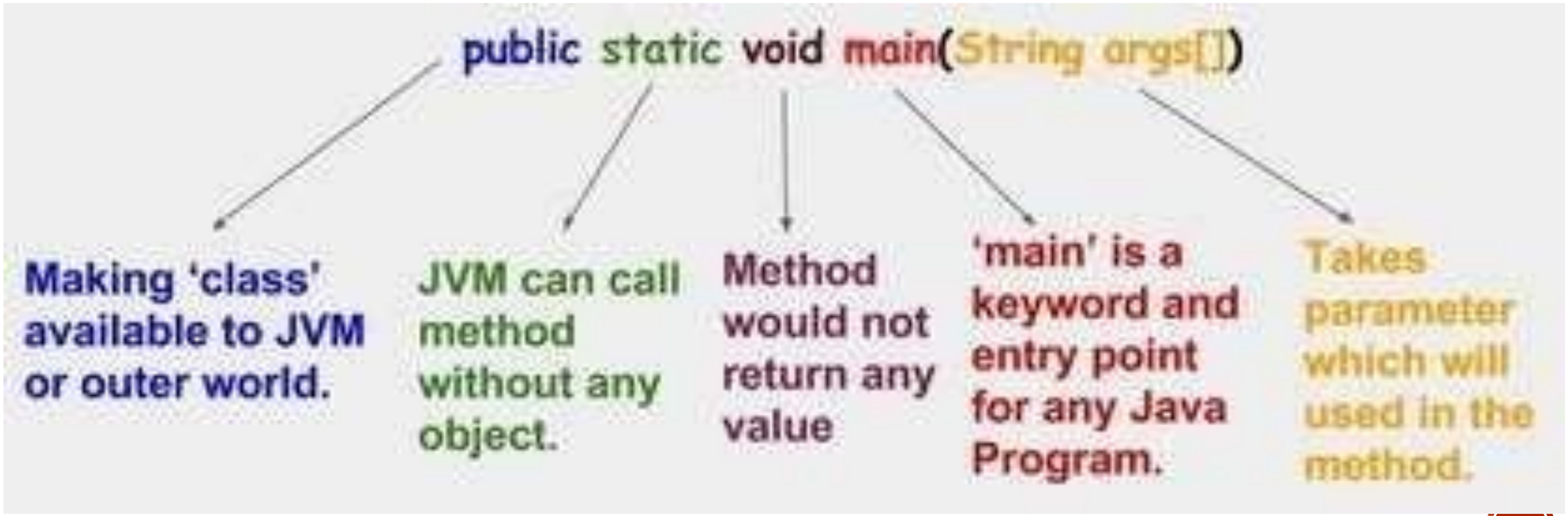
```
class StaticDemo {
        static int a = 42;
        static int b = 99;

        static void callme() {
                System.out.println("a = "  + a); } }

class StaticByName {

        public static void main(String args[]) {

                StaticDemo.callme();

                System.out.println("b = "  + StaticDemo.b); } }
```

Output:

```
a = 42
b = 99
```

WHY IS THE JAVA MAIN METHOD STATIC?

- It is because the object is not required to call a static method.
- If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation.


public static void main(String args[])
Making 'class' available to JVM or outer world.
JVM can call method without any object.
Method would not return any value
'main' is a keyword and entry point for any Java Program.
Takes parameter which will used in the method.

STATIC BLOCK

Output:

```
static block is invoked
Hello main
```

- Is used to initialize the static data member.
- It is executed before the 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"); } }
```

EXAMPLE

```
// Demonstrate static variables, methods, and blocks.
class UseStatic {
		static int a = 3;
		static int b;
		static void meth(int x) {
		System.out.println("x = " + x);
		System.out.println("a = " + a);
		System.out.println("b = " + b);
}
		static {
				System.out.println("Static block initialized.");
				b = a * 4; }

		public static void main(String args[]) {
				meth(42); } }
```

Output:

```
Static block initialized.
x = 42
a = 3
b = 12
```

CAN WE EXECUTE A PROGRAM WITHOUT MAIN() METHOD?

- No,
- one of the ways was the static block, but it was possible till JDK 1.6.
- Since JDK 1.7, it is not possible to execute a Java class without the main method.

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