static
 => This access modifier can applied at
      a. class
      b. variable
      c. method
      d. block

What type of variable we need to make as static in oops?
 if we want all the objects of a particular class to get a common copy, then such variables should be made as
 "static",so we say static variables as class variables.

What type of methods we need to make as static in oops?
  if we want some behaviour to be executed even without its object creation, then such type of methods we need
  to mark as "static".

static control flow
++++++++++++++++++
Can we write multiple static blocks, if yes how will they get executed?
 Ans. yes possible, it gets executed in order of placing the block in the class.

eg#1.
```
class Test
{
      static
      {
            System.out.println("First static block");
      }

      static
      {
            System.out.println("Second static block");
      }

      public static void main(String[] args)
      {

      }
}
```
Output
```
First static block
Second static block
```

eg#2.
```
class Test
{
      static int i = 10;

      static
      {
            methodOne();
            System.out.println("First static block");
      }

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

```
            methodOne();
            System.out.println("Inside main method");
      }

      public static void methodOne()
      {
            System.out.println(j);
      }
      static
      {
            System.out.println("Second static block");
      }
      static int j = 20;
}

Output
0
First static block
Second static block
20
Inside main method



Read Indirect Write Only (RIWO)
++++++++++++++++++++++++++++++
Direct Read   => Within a static block, if we are reading a variable then such type
of read is called as "Direct read".
Indirect Read => If we are calling a static method,and with in the static method if
we are reading a static variable then
                  such type of read is called "Indirect Read".

eg#1.
class Test
{
      static int i = 10;
      static
      {
            System.out.println(i);
      }
      public static void main(String[] args)
      {

      }
}
Output
10

eg#2.

If a variable is in RIWO state,then we can't perform direct read operation, if we
try to do it would result in CompileTime Error.
class Test
{
      static
      {
            System.out.println(i);
      }
      public static void main(String[] args)
```

```
	{

	}
	static int i = 10;
}
Output: CompileTimeError :illegal forward reference


eg#3.
class Test
{
	static
	{
		methodOne();
	}
	public static void main(String[] args)
	{

	}
	public static void methodOne()
	{
		System.out.println(i);
	}
	static int i = 10;
}
Output
 0

Usage of static block in realtime
++++++++++++++++++++++++++++++++
static block : It is called as "One Time Execution block".
		It will be executed during the loading of .class file.

Note1:
 Every driver software internally contains static block to register the driver with
DriverManager, which helps the programmer to
 get JDBC environment in JRE, to do this we need static block.

Note2:
 Can we write any statements to the console without writing statement inside
main()?
 Answer : yes , by using static block.

Note3:
 Without using main() and static block statements, is it possible to write any
statements to the console?

eg#1.
class Test
{
	static int i = methodOne();

	public static void main(String[] args)
	{

	}

	public static int methodOne()
	{
```

```
            System.out.println("Hello i can print");
            System.exit(0);
            return 10;
      }

}
Output
Hello i can print

eg#2.
class Test
{
      static Test t = new Test();
      public Test()
      {
            System.out.println("Hello i can print");
            System.exit(0);//shutdown jvm
      }

      public static void main(String[] args)
      {

      }
}
Output
Hello i can print


Note: It is mandatory to write main() inside every class for the execution to
happen, if we don't write then class will not be loaded.
            public static void main(String[] args).

      System.exit(0) -> This line if jvm executes, then jvm will shutdown itself,
by skipping the remaining statements in the program.


instance control flow
++++++++++++++++++++
 instance block : This block gets executed at the time of creating an object,but
before the call to constructor.
              This block will be executed for every object we create,but before the
call to a constructor.

Note:
  1. Constructor    : initialize the object with the required values.
  2. instance block : Apart from initalization, if we want to perform any other
activities then we need to go for "instance block".
  3. Order of execution : first instance block then constructor.


eg#1.
class Student
{
      //instance variable
      int sid;
      String sname;
      int sage;

      //constructor: shadowing -> resolved through "this"
```

```
	Student(int sid, String sname,int sage)
	{
		System.out.println("Supplied values through constructor");
		this.sid   = sid;
		this.sname = sname;
		this.sage  = sage;
	}

	//instance block
	{
		System.out.println("Supplied values through instance block");
		sid = 100;
		sname = "dhoni";
		sage = 41;
		dispStdDetails();
	}

	//instance method
	public void dispStdDetails()
	{
		System.out.println("SID   is :: "+sid);
		System.out.println("SNAME is :: "+sname);
		System.out.println("SAGE  is :: "+sage);
	}
}

class Test
{
	public static void main(String[] args)
	{
		Student std = new Student(10,"sachin",50);
		std.dispStdDetails();
	}
}

Output
Supplied values through instance block
SID   is :: 100
SNAME is :: dhoni
SAGE  is :: 41

Supplied values through constructor
SID   is :: 10
SNAME is :: sachin
SAGE  is :: 50


eg#2.
class Test
{
	int i = 10;
	{
		methodOne();
		System.out.println("First instance block");
	}
	Test()
	{
		System.out.println("Test class constructor");
	}
```

```
	public static void main(String[] args)
	{
		Test t =new Test();
		System.out.println("Inside main method...");
	}
	public void methodOne()
	{
		System.out.println(j);
	}
	{
		System.out.println("Second instance block");
	}
	int j = 20;
}
```

Output

```
0
First instance block
Second instance block
Test class constructor
Inside main method...
```