class A

{

public static A ref=new A();

static

{

System.out.println("in A static block");

}

A()

{

System.out.println("in A default constructor");

}

}

public class DemoApp

{

public static void main(String args[])

{

System.out.println(A.ref); System.out.println("in main");

}

}

**why constructor is called first here?**

**our code**

class A {

public static A ref = new A();

static {

System.out.println("in A static block");

}

A() {

System.out.println("in A default constructor");

}

}

public class DemoApp {

public static void main(String args[]) {

System.out.println(A.ref);

System.out.println("in main");

}

}

**Sequence of events when we run DemoApp**

1. **DemoApp class is loaded first**
   * JVM starts with DemoApp.main().
   * At this point, A is not yet loaded.
2. **First reference to A → triggers class loading of A**
   * Inside main(), you access A.ref.
   * JVM sees that class A must be loaded and initialized.
3. **Class A loading → static variable initialization happens in the order of declaration**
   * Static fields and static blocks are executed **in the order they appear in the class**.
   * So first:
   * public static A ref = new A();

needs to be initialized.

* + To initialize this field, new A() is called → which runs the **constructor**.
  + in A default constructor

1. **Next, static block executes**  
   After all static fields declared above it are initialized, the static block executes:
2. in A static block
3. **Back to main()**
   * At this point A.ref is already initialized and printed.
   * Finally "in main" is printed.

**Final Output**

in A default constructor

in A static block

A@some\_hashcode

in main

The **constructor executes before the static block here** because the static variable ref requires creating an object *before* the static block is reached.

Static **initialization order** is:

1. Static variables **in order of declaration**.
2. Then static blocks **in order of declaration**.

So, the constructor runs during the initialization of ref, and *only after that* the static block runs.