Working with static and non-static

Static members

Non-static members

Static block

Non-static block

|  |  |
| --- | --- |
| Program | Execution |
| class Apple{  static int greenApple=3;  int redApple;  public static void main(String ar[]){    }  static void eatGreenApple(){  }  void eatRedApple(){  }  }  Apple a1 = new Apple();  new Apple(); | redApple  eatRedApple()  redApple  eatRedApple()  Apple class  greenApple=3  eatGreenApple()  main() |

Designing a class means defining a class in simple words. Class contains **variable** which holds value, **method** which perform actions (& also can contain inner classes). Together we call it as **members**.

You can declare the variables and methods in a class with the key word **static.** Then these members will become **static members.**

Variables and methods declared in class without static keyword is called as **non-static members or instance members or object members.**

Static members belong to **class** & non static/Instance members belong to **Object.**

Static members are only one per class (one copy). Whereas instance members, each object has its own copy of non-static members/instance members.

Static variable also called as class variables.

Non static variables also called as instance variables.

**Class**

**Variables**

**Methods**

**Inner classes**

**Non-Primitive**

**Primitive**

**Array Component**

**Instance**

**Local**

**Class level**

1. **Static**
2. **Non Static**

**Method Parameter**

**Constructor Parameter**

**Exception Parameter**

Example:

class VariableDemo{

static int a=10; //static variable/class variable (Fields)

int b=20; //non-static variable/instance variable

public static void main(String args[]){

int a=30; //local variable

{ //statement block

int j=20; //local variable

}

System.out.println(a);

}

System.out.println(a);

}

* Java always gives preference to local variable when a variable with same name is present in both local and in class (static).
* If local variable is not same as class variable, then you directly access the class variable.
* Any executable statement should be part of a method and in class we can only declare and initialize variable.
* Whenever we use class variable (static) or method (static) always use class name.memberName. Don’t use it directly.

**What is the difference between static members and non static members?**

Static members are variables and methods which are declared with keyword static belong to class and they are one per class whereas non-static members are variables and methods declared without keyword static where each object has its own copy and they belongs to object.

**Note:**

1. To access static members we use class name and **.** (dot) operator along with member name.
2. To access non static members (Variables/Methods) from static reference (method) we use reference variable and **.** (dot) operator along with member name.

|  |  |
| --- | --- |
| Using static variables and methods | Using non-static variables and methods |
| class Demo{  static int i=20;  public static void main(String a[])  {  Demo.test();  }  static void test(){  System.out.println(Demo.i);  }  } | class Demo{  int a=20;  public static void main(String a[])  {  Demo d1=new Demo();  d1.a=30;  d1.test();  }  void test(){  System.out.println(a);  }  } |

Assignment: WAP to print a string vertically.

**Note:**

* Static members can be accessed anywhere in the program. It can be called from static and non-static method directly or with class\_Name.member\_Name.
* Non-static members can be accessed by non-static methods directly. Non-static members from a static method we have to use reference\_Variabe**.**(dot) OR object**.**(dot) member\_Name.

**JVM**

**Object stored in heap memory**

**Memory Location**

**Storage purpose**

**Execution purpose**

|  |  |
| --- | --- |
|  |  |
| Stack | Heap |

Until we create an object we can’t access non-static members.

Example:

Class Demo{

int a = 100;

}

Class MemoryDemo{

Public static void main(String args[]){

Demo d1 = new Demo();

d1.a=200;

Demo d2 = new Demo();

d2.a=300;

}

}

|  |  |
| --- | --- |
| **d2**  **d1**  **main**  **i=200**  **i=300** |  |
| Stack | Heap |

When main method is done, java (jvm) calls garbage (program) collector before terminating, to clean the memory.

**Static block**

It is a block { code } with keyword **static** which gets executed when the class is loaded (before the main () method) and it is used to initialized static variables.

If there is a static block, java first executes the static block then the main method.

static

{

Static block

}

You can have multiple static blocks. Always it executes in top to bottom sequence order.

Example

class staticBlock{

static int i=10;

static{

i=100;

System.out.println(3);

}

public static void main(String args[]){

System.out.println(i);

}

static{

i=200;

System.out.println(1);

}

static{

O/p: 3

1

2

300

i=300;

System.out.println(2);

}

}

Class loaded

1. Static variable loaded and initialized.
2. Static method gets loaded
3. Static block gets executed (if it’s there)
4. Main method gets executed.

|  |
| --- |
| Example:  class Apple{  static int a=10; //1  static{ //3  System.out.println(a);  print();  a=1111;  }  public static void main(String args[]){ //4  System.out.println(Apple.a);  System.out.println(Orange.o);  }  static{ //3  a=3333;  }  static void print(){ //2 (loaded)  System.out.println(“Hello”);  }  }  class Orange{  static int o=20; //1  static{ //3  o=2222;  }  } |
| O/P : 10  Hello  3333  2222 |

**Note**

* When we compile the above file Apple.java we get two class files i.e. Apple.class and Orange.class.
* Sequence of the execution

1. When the class is loaded the static variables are get loaded and initialized. If no value is given it is initialized to default value.
2. Static methods are loaded in the memory.
3. Static block or blocks gets executed. If there is more than one static block, all the block gets executed sequentially from top to bottom.
4. Then main method is getting executed.
5. In main method if we are calling members of any other class like Orange.o, first Orange.class is loaded in the memory, here also same sequence happens.

Suppose if Orange class also had main() method then you need to execute that class separately like **java Orange.**

**Assignments: WAP to validate password field.**

Requirements

First character should be in caps.

Should contain at least one special character.

Should contain at least one number.

**Non-static block**

It is a block { Code } without keyword static which gets executed every time an object of class containing non-static block gets created.

{

Non-static block

}

You can have multiple non static blocks, always it executes in top to bottom sequentially order.

Example:

**class** NonStaticBlock {

**int** i,j;

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

NonStaticBlock ns1=**new** NonStaticBlock();

System.***out***.println(ns1.i);

System.***out***.println(ns1.j);

NonStaticBlock ns2=**new** NonStaticBlock();

System.***out***.println(ns2.i);

System.***out***.println(ns2.j);

System.***out***.println(**new** NonStaticBlock());

System.***out***.println(**new** NonStaticBlock().i);

System.***out***.println(**new** NonStaticBlock().j);

ns2.i=123;

ns2.j=456;

System.***out***.println(ns1.i);

System.***out***.println(ns1.j);

System.***out***.println(ns2.i);

System.***out***.println(ns2.j);

}

{

System.***out***.println("Non-Static Block 1");

i=20;

j=30;

}

{

System.***out***.println("Non-Static Block 2");

i=200;

j=300;

{

i=222;

j=333;

}

}

}

o/p:

Non-Static Block 1

Non-Static Block 2

222

333

Non-Static Block 1

Non-Static Block 2

222

333

Non-Static Block 1

Non-Static Block 2

staticNonStatic.NonStaticBlock@15db9742

Non-Static Block 1

Non-Static Block 2

222

Non-Static Block 1

Non-Static Block 2

333

222

333

123

456

In class contains two types of members

1. Static
2. Non-static

|  |  |
| --- | --- |
| Static | Non-static |
| Static members are created using static keyword on variable and method. | Non-static members are created without using static keyword on variable and method. |
| Static member belongs to class. | Non-static member belongs to object. |
| Static members are one per class. | Each object has its own copy of non-static members. |
| All static members can be accessed by using . (dot) operator on the class name. | Non-static members can be accessed by using reference variable and **.** (dot) operator. |
| Java provides separate block to initialize static variables of the programs, this block is known as Static Initialization Block (SIB). | Java provides separate block to initialize non-static variables of the programs, this block is known as **Instance Initialization Block (IIB)**. |
| Before executing main method, SIB will be executed first, when the class is loaded. | IIB gets executed when an object of the class is created. |
| SIB is executed only once when the class is loaded. | IIB is called (multiple times) each time an object is created. |
| Static variables are initialized when the class is loaded. | Non-static variables are initialized when the respective object is created. |
| We can have any number of SIB. | We can have any numbers of IIB. |
| The execution of the block is sequential (top to bottom). | The execution of the block is sequential (top to bottom). |
| The static variables have the recent values initialized by last SIB. | The non-static variables have the recent values initialized by last IIB. |
| Static members also called class members. | Non-static members are also called Reference members, Instance members, Object Members. |

**Assignment: WAP to implement a non-static method which initializes variable with passed value and also returns the same value.**