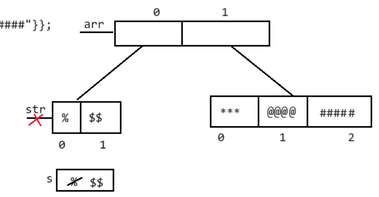
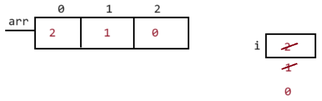
```
String [][] arr = { {"%", "$$"}, {"***", "@@@@", "#####"}};
for(String [] str : arr)
{
    for(String s : str)
    {
        System.out.println(s);
        if(s.length() == 4) //Line n1
            break; //Line n2
    }
    break; //Line n3
}
```

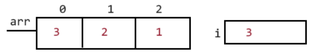



```
int [] arr = {2, 1, 0};
for(int i : arr) {
    System.out.println(arr[i]);
}
```

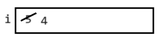


0
1
2

```
int [] arr = {3, 2, 1};
for(int i : arr)
{
    System.out.println(arr[i]);//AIOBE
}
```




```
int i;
outer: do
    {
        i = 5;
        inner:
        while (true)
        {
            System.out.println(i--);
            if (i == 4) {
                break outer;
            }
        }
    } while (true);
```



5

```
class Student
{
    //instance variables
    String name;
    int age;
    float height;
}
```

Before Compilation

```
class Student
{
    //instance variables
    String name;
    int age;
    float height;
    Student()
    {
        super();
    }
}
```

[Added by compiler]
default constructor
(Compiler)

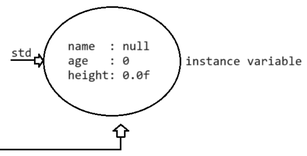
AfterCompilation

classname same as methodname => Constructor

```
//Constructing the object
Student std = new Student();
```

constructor
(meaningful values to the instance variables)

creates an object



instance variable

```
class Student
{
    //instance variables
    String name;
    int age;
    float height;

    //Parameterized constructor
    Student(String name, int age, float height)
    {
        this.name = name;
        this.age  = age;
        this.height = height;
    }
}
```

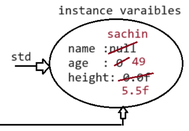
#3
gives meaningful values
to the instance variables

```
    #4
Student std = new Student("sachin",49,5.5f);
```

Creates an Object

constructor
#2

(#1)



1. Creates an object
2. Load the .class file and scan for instance variable allocated memory supply default values.
3. Call the constructor(either default or parameterized)
4. return the reference.

```
class Student
{
    //instance variables
    String name;
    int age;
    float height;

    //Parameterized Constructor
    Student(String name, int age, float height)
    {
        this.name = name;
        this.age  = age;
        this.height = height;
    }

    //Zero Argument Constructor
    Student()
    {
        name   = "dhoni";
        age    = 41;
        height = 5.6f;
    }
}
```

good practise

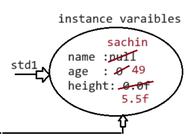
not a good practise

Constructor Overloading

```
//Constructing the object
Student std1 = new Student("sachin",49,5.5f);

//getting the values from instance variables
System.out.println("Name   is :: "+std1.name);
System.out.println("Age    is :: "+std1.age);
System.out.println("Height is :: "+std1.height);
```

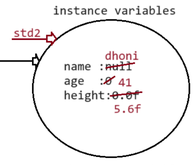
sachin
49
5.5




```
//Constructing the object
Student std2 = new Student();

//getting the values from instance variables
System.out.println("Name   is :: "+std2.name);
System.out.println("Age    is :: "+std2.age);
System.out.println("Height is :: "+std2.height);
```

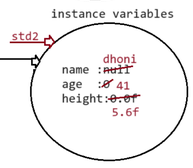
dhoni
41
5.6




```
//Constructing the object
Student std3 = new Student();

//getting the values from instance variables
System.out.println("Name   is :: "+std3.name);
System.out.println("Age    is :: "+std3.age);
System.out.println("Height is :: "+std3.height);
```

dhoni
41
5.6



```
class Student
{
    //instance variables
    String name;
    int age;
    float height;

Student(String name, int age, float height)
{
    System.out.println("CALLING THE CONSTRUCTOR");
    this.name = name;
    this.age  = age;
    this.height = height;
}
}
```

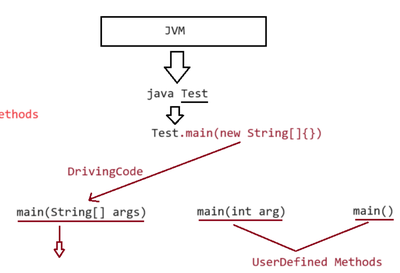
constructor
(It gets called automatically at the time
of Object construction)

```
void Student(String name,int age, float height)
{
    System.out.println("CALLING THE METHOD");
    this.name = name;
    this.age = age;
    this.height = height;
}
```

Method(Programmer should call this method using reference)

```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("Inside String[] args");
    }
    public static void main(int arg)
    {
        System.out.println("Inside int arg");
    }
    public static void
    {
        System.out.println("Inside zero argument");
    }
}
```

Overloading static methods
(Yes, possible)



```
class Student
{
    static String nationality = "IND";
    String name;
    int age;

    Student(String name,int age)
    {
        this.name = name;
        this.age  = age;
    }
}
```

static variable

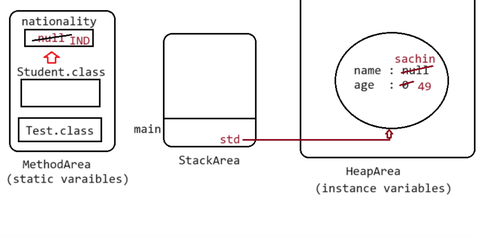
instance variable

Constructor
(shadowing avoided using this)

```
Student std= new Student("sachin",49);
```

#1 #2 #3

```
System.out.println("Name        is :: "+std.name); //sachin
System.out.println("Age         is :: "+std.age); //49
System.out.println("Nationality is :: "+Student.nationality);//IND
```



```
class Student
{
    String name;
    int age;
    static String nationality = "IND";

    Student(String name,int age)
    {
        System.out.println("Constructor got called");
        this.name = name;
        this.age  = age;
    }
```

#6 instance varaibles

#4 static variables

#7

constructor

```
    static
    {
        System.out.println("Static Block :: Loading of Student.class file");
    }
```

#5

static block

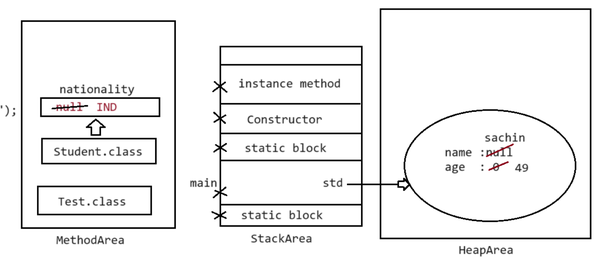
```
    public void dispStdDetails()
    {
        System.out.println("Inside instance method");
        System.out.println("Name is          :: "+name);
        System.out.println("Age  is          :: "+age);
        System.out.println("Nationality  is :: "+nationality);
    }
}
```

#10

sachin 49

IND

instance area



```
class Test
{
    static
    {
        System.out.println("Loading of Test.class file");
    }
}
```

#1

static block

```
    public static void main(String[] args)
    {
        System.out.println("Inside main()");
        Student std= new Student("sachin",49);
        std.dispStdDetails();
    }
```

#2 #8 #3 #9

static method