**Classes in Java:**

A class is a blue print from which individual objects are created.

A sample of a class is given below:

A class can contain any of the following variable types.

* **Local variables .** variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
* **Instance variables .** Instance variables are variables within a class but outside any method. These variables are instantiated when the class is loaded. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
* **Class variables .** Class variables are variables declared with in a class, outside any method, with the static keyword.

A class can have any number of methods to access the value of various kind of methods.

General form of a class is as follows:

Class className

{

type1 dataName1;

type2 dataName2;

;

typeN dataNameN;

returntype1 methodname1(argument list)

{

;

;

}

Returntype2 methodname2(argument list)

{

;

;

}

returntypeN methodnameN(argument list)

{

;

;

}

}

****

The class definition includes the following components:

* Access modifier Keyword to specify how external access to this class is managed. Options

include public or private.

* Class keyword A mandatory keyword.
* Instance variables Variables (constants) defined outside of a method and available to all

methods in the class.

* Class variables Variables (constants) defined with the static keyword – a single instance

of the variable is created which is shared by all instances of the class.

* Instance variables are created when the class is loaded initially and can be

set and accessed even before new instances of the class are created.

* Local variables Variables (constants) defined inside a method. The variable scope is

inside the method where it is declared.

* Instance methods Functions (subroutines) that operate on instances of the class.
* Class methods Functions (subroutines) that operate on class variables of the class.
* Constructors Methods that have the same name as the class and are automatically called to create new instances of the class (initialise instance variables).

**Example**

class square

{

# int side;

# int area;

# void squarearea()

# {

# area=side\*side;

# }

# void display()

# {

# System.out.println(“the area is “ + area);

# }

# In the above example, a class named square has two data members *side* and *area* and two methods *squarearea()* and *display().*

# The variables that are declared inside a class can be accessed directly like in *squarearea()* and *display()* methods. But this is also due to the fact that by default the variables and methods which are not specified with any access specifier (private, public, protected). But when they are going to be accessed outside of their class or from other classes they have to be prefixed with the object instantiation of that class followed by a dot operator (.)

# 

**Methods**

Each method is defined within a class, and can be equivalent to a subroutine, a procedure or a

function in other programming languages.

For each method, the following forms part of the definition:

* Access modifier Keyword to specify how external access to this method is managed.

Options include:

* public - accessible from other classes
* private - not accessible outside this class
* protected - accessible from sub-classes of this class
* default - accessible from other classes in the same package

(the default keyword does not appear but is assumed).

* Static keyword A mandatory keyword for class methods.
* Return type The data type returned by this method. If the method does not return a

value, the data type void must be used.

* Method name The name of the method.
* Arguments A comma separated list of datatypes and names passed as parameters to

this method.

* Method body Java statements that provide the functionality of this method.

Unless a method does not return a value (and has a signature specifying a void return type), the

method must end with a ‘return’ statement to provide the value to the calling method. A method

may have several exit points (return statements) – each statement must return the same type.

# Creating an Object:

As mentioned previously a class provides the blueprints for objects. So basically an object is created from a class. In java the new key word is used to create new objects.

There are three steps when creating an object from a class:

* **Declaration .** A variable declaration with a variable name with an object type.
* **Instantiation .** The 'new' key word is used to create the object.
* **Initialization .** The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Example of creating an object is given below:

|  |
| --- |
| class test{public static void main(String args[]){ Declaration  instantiation  initialization square obj1=new() square();obj1.side=10;obj1.squarearea();obj1.display();}} |

If we compile and run the above program then it would produce following result:

# Accessing Instance Variables and Methods:

Instance variables and methods are accessed via created objects. To access an instance variable the fully qualified path should be as follows:

|  |
| --- |
| /\* First create an object \*/  ObjectReference = new Constructor();  /\* Now call a variable as follows \*/  ObjectReference.variableName;  /\* Now you can call a class method as follows \*/  ObjectReference.MethodName(); |

## Example:

This example explains how to access instance variables and methods of a class:

|  |
| --- |
| class Puppy{    int puppyAge;  public Puppy(String name){  // This constructor has one parameter, *name*.  System.out.println("Passed Name is :" + name );  }  public setAge( int age ){  puppyAge = age;  } |