

Contents

Classes & Objects

Java Class

```
class classname                // class definition
{
    type variablename;          // fields declaration
    default constructor         // constructor declaration
    type methodname (parameter - list) // methods declaration.
    {
        method – body;
    }
}
```

Example :

```
class Book
{
    String bookName;
    String authorName;
    int noPages;
    float price;
    String displayname( )
    {
        System.out.println("Name of the book is " + bookName);
    }
}
```

Access Modifiers

Location/Access Modifier	Private	Default	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

this – points to the current class instance.

```
class Account{
int a;
int b;
public void setData(int a , int b){
    a=a;
    b=b;
}
public static void main(String args[]){
    Account object1 = new Account();
    object1.setData(2,3);
}
```

the compiler gets confused whether the variable on the left hand side of an operator is an instance variable or a class variable

```
public void setData(int a , int b){
    this.a=a;
    this.b=b;
}
public static void main(String args[]){
    Account object1 = new Account();
    object1.setData(2,3);
}
```

use keyword "this" instead of instance variable

after declaring "this" keyword, now we will call the setData method, and see what happens.

Packages – avoids namespace collision

package pack1;

class Teacher

class Student

package pack2;

class Student

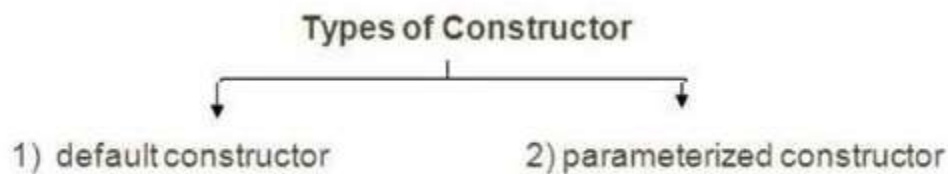
class Courses

```
import pack1.*;
import pack2.*;
pack1.Student stud1;
pack2.Student stud2;
Teacher teacher1;
Courses course1;
```

Use * to import all classes in a package.

Package Name	Classes Available
java.lang	Object class, String class, System class, wrapper classes.
java.util	Utility classes, such as Date & collections (hashmap, treeset, arraylist, hashtable, treemap, etc.)
java.io	Input & output classes for writing & reading from streams & files

Constructors



Rule : If there is no constructor in a class, compiler automatically creates a default constructor

```

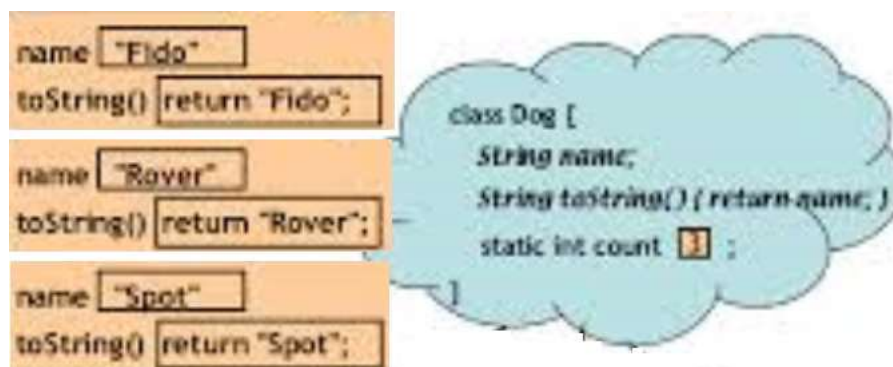
class Student{
    String course;
    String address;
    int semester;

    //Default Constructor - NO ARGUMENT
    public Student() {
        //Creates space in memory for the object and initializes its fields to default.
    }

    //Parameterized Constructor - ALL ARGUMENTS
    public Student(String s1, String s2, int i1){
        this.course=s1;
        this.address=s2;
        this.semester=i1;
    }

    //Parameterized Constructor - FEW ARGUMENTS
    public Student(String s1){
        this.course=s1;
    }
}
  
```

static



Things which can be marked as static	Methods, variables, nested classes
Things which can NOT be marked as static	Constructors, classes, interface, local variables