**What are packages?**

Packages are nothing but folders. Problem-> you can not make two files with the same name in the same location.

Scenario: for example, you want to make two files with the same name in the project each will serve its own specific purpose. But in the same location you cannot make another file with the same name. So, the packages concept come onto the picture. You made different packages with like package **A** and package **B.** Now package a has a file name **greetings.java** and the package b also has the file with the name **greetings.java** and now it is possible to create different files with the same name.

Packages com.jamal.introduction.a

The file is inside the com folder, inside jamal folder, inside introduction folder, inside a folder

**Static ?**

We use static when we want to access the method of the class without using the object or creating the object t of that class.

The properties and methods that are independent to object is called static.

The static properties and methods are object independent, so it can be accessed via class name directly.

Without reference to the object, without being created the object the static variable can be accessed using the class name and the property.

**For example:**

Class human{

Int age;

String Name;

Static long population

}

Now the population should be accessed through the class name(human) because it is conventional.

Like:

human.population

**NOTE:** in static method or class you can not use anything that is not static. Like you cannot make an object and then call the function from non-static class in main function of java which is static.

Anything that is not static belong to an object.

You can call a static function/method in another non-static method. But cannot non-static in static.

Class main{

Static void fun(){

greeting (); //wrong because this fun function is limited. It does not depend on object so it does not accommodate the non-static properties and methods .

}

Void greeting (){

fun(); // True because this greeting function is not limited. It depend on object so it also accommodate that the properties and methods that does not depend on object

}

}

**=>Other way** to call the non-static in static is by making the object of that class and then using that object call that non-static function.

Class main{

Static void fun(){

Main obj=new Main(); // object making of main class

obj. greeting(); //calling the method using object

}

Void greeting (){

fun();

}}

**=>Static blocks**

Class staticcls{

Static int a=4;

Static int b;

Static{

System.out.println(“I am in static block”);

System.out.println(“the value of b is”+ b\*4);

}

The static block runs only once when the class is loaded for the first time , when the first object of the class created if we create another object then the static block will not execute.

**=>Inner Classes**

Outer classes can not be static because they does not depend on another class so they should have to be have an object. But inner classes can be static.

Accessing the outside class inside another class:

class Test {  
 public Test(String a) {  
 System.*out*.println("Here is the Test child class: " + a);  
 }  
 //This Test class depends on the the object so it can be directly access in  
 // the main function its object  
 }  
  
public class InnerClasses {  
 public static void main(String[] args) {  
 System.*out*.println("The main function");  
 Test t1 = new Test("Umer");  
 }  
}

Accessing the inner class inside the main function of parent class via static keyword

public class InnerClasses {  
 class Test {  
 public Test(String a) {  
 System.*out*.println("Here is the Test child class: " + a);  
 }  
//This Test class depends on the outer class InnerClasses so it //cannot makeits own object in main function to run. so we should have to use //static class that make this test class independent then we can access the //test class by making its object in main  
  
 }  
 public static void main(String[] args) {  
 System.*out*.println("The main function");  
 Test t1 = new Test("Umer");  
 }  
}