**Static**

**Static Variabels**

1. static variables are also known as **class variables**.
2. static variables will remain same for different instance/objects of class but for every new object instance variables will be initialized to new value.
3. If static variable is a reference to an object that which is stored in the normal sections of the heap. Those objects are not stored in PermGen space.

**static int id = 1; //**the value 1 is stored in the permgen area.

**static Employee emp = new Employee ();** **//the reference(i.e. emp ) is**

**//stored in the permgen area, the object is not.**

**static String *company*="XYZ pvt ltd"; //the reference(i.e. *company*) is**

**//stored in the permgen area, but  "XYZ pvt ltd" is not,**

**//"XYZ pvt ltd" gets stored in String pool.**

**Static method**

1. We need not to create instance of class for accessing static methods
2. Static methods can access all static variables, but cannot access non-static (instance variables)
3. Because static methods are class methods, access to them is always resolved during compile time only.
4. Static method cannot be overridden with non-static method, any attempt to do this **will** cause compilation error.
5. Non-static method cannot be overridden with static method, any attempt to do this **will** cause compilation error.
6. **Static method cannot be overridden,** any attempt to do this will **not** cause compilationerror, but the results won’t be same when we would have overridden non-static methods.



**Static Class**

1. StaticNestedClass can be **abstract or** [**final**](http://www.javamadesoeasy.com/2015/05/final-keyword-in-java-20-salient.html).
2. StaticNestedClass can be **private, protected, public.**
3. Static nested **classes can declare static initialization blocks**
4. Static nested **classes can declare static initialization blocks**

**Nested Class**

**Two types :**

1. **static nested**
2. **non-static nested class(inner class)**

**non-Static nested class(inner class)**

1. Has access to all fields of outer class (static and non-static including private)
2. All fields and methods of outer class are accessed without using the outerClass.FIELD/METHOD. Simply use the variables and methods as if they are declared in the nested class.
3. Members of the inner class are known only within the scope of the inner class and may not be used by the outer class. However we can always access the members of inner class in outer class with reference object.
4. Non static inner class **cannot** have **static** variables while they can have **static** **final** **variables.** Similarly they also cannot declare static methods. Static methods can only be declared in top level class or static nested class.
5. It is good practice to instantiate inner class only in containing class. However it is always possible to instantiate it in some other class using the reference of outer class.

**Static nested class**

1. Top level class can never be static in java.
2. static class are also known as static nested classes
3. Only **static variables and methods of outer class** can be accessed inside static nested class. However they can have their own static as well as non-static variables and methods.

There are two special kinds of **inner** (non-static) classes:

1. [**Local classes**](https://docs.oracle.com/javase/tutorial/java/javaOO/localclasses.html)
2. [**Anonymous classes**](https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html)**.**

**Local classes**

Local classes are classes that are defined in a *block*, which is a group of zero or more statements between balanced braces. You typically find local classes defined in the body of a **method**. It can also be in a **for loop**.

1. A local class has access to the members of its enclosing class
2. In addition, a local class has access to local variables. However, a local class can only access local variables that are declared final.
3. Starting in Java SE 8, a local class can access local variables and parameters of the enclosing block that are final or effectively final
4. Just like inner class local class cannot have static variables or methods. However they have access to static variables of outer class.

**NOTE**: Interface cannot be defined inside a block.



**Anonymous Class:**

Anonymous classes enable you to make your code more concise. They enable you to declare and instantiate a class at the same time. They are like local classes except that they do not have a name. Use them if you need to use a local class only once.



Ex 2:



In this example, the method invocation btn.setOnAction specifies what happens when you select the Say 'Hello World' button. This method requires an object of type EventHandler<ActionEvent>. The EventHandler<ActionEvent> interface contains only one method, handle. Instead of implementing this method with a new class, the example uses an anonymous class expression.

# Enum Types

An enum type is a special data type that enables for a variable to be a set of predefined constants. You should use enum types any time you need to represent a fixed set of constants. That includes natural enum types such as the planets in our solar system and data sets where you know all possible values at compile time—for example, the choices on a menu, command line flags, and so on.

The java enum constants are static and final implicitly. Java Enums can be thought of as classes that have fixed set of constants.

* enum may implement many interfaces but cannot extend any class because it internally extends Enum class.

The java compiler internally adds the values() method when it creates an enum. The values() method returns an array containing all the values of the enum.