Java Anonymous Classes & Lambda Expressions

→ Anonymous class is nothing but a class without any name.
→ They are used to override a class method or interface.
→ They help us to write more concise & readable codes.

Syntax:

```
//Demo can be interface or abstract
class

 Demo t= new Demo1() {

   // data members & methods
   public void Demo_methods()
   {
      - - - -
      - - - -
   }

 };
```

E.g. code:

```
@FunctionalInterface
interface Animal{
    void bark();
    }

class Dog implements Animal{

    @Override
    public void bark(){
        sout("Dog barks!");
                }

}

~~class Anon~~
class AnonDemo{
    p s v m(){
     Dog Bruno = new Dog();
     Bruno. bark();
     }
}
```

→ In the above ~~method~~ e.g. Animal is a Functional Interface containing a bark() method inside it.

→ Class Dog implements the Animal interface & overrides the bark () method.

→ Bruno is an object of Dog class on which we are running the bark () method.

→

O/p:

Dog barks!

The same o/p can be generated without creating the Dog class. This is the scenario when the ~~Anon~~ Anonymous classes come into picture. With the help of it we can declare & instantiate the class at the same time

E.g.

```
@FunctionalInterface
interface Animal{
    void bark();
          }

class AnonDemo{
        p s v m (          ){
        Animal Bruno = new Animal(){
          @Override
        public void bark(){
            sout("Dog barks!");
                  }
        };
        Bruno.bark();
        }
}
```

In the above code, we've created the Bruno object by referencing the animal interface. So, that's how we've overriden the bark() method without creating any seperate class.

<u>Ways to create an anonymous Java class:</u>

→ By extending a class
→ By implementing an interface

⊙ By extending a class.

```
abstract class Vehicle {

    abstract void drive();
                  }


class AnonDemoByClass {
    p   s   v   m  (   ) {
      Vehicle car = new Vehicle ();
         @ Override
    void drive (){
    sout ("I'm driving a car");
         }
    };

   car. drive();
   }
}
```

o/p
I'm driving a car

③ __By Implementing an Interface__

```
@FunctionalInterface

interface Human{
      void walk();
    }


class AnonDemo{
        p  s  v  m(   ){
          Human John = new Human(){

        @Override
        public void walk(){
            sout("John walks");
          }
      };
    John.walk();
    }
}
```

o/p:

```
John walks
```

Lambda Expressions

→ lambda expressions were ~~introd~~ introduced in Java 8.

→ They are similar to methods but don't need a name.

Syntax:

(parameter 1, parameter 2) -> {code to be executed}

E.g.

```
@FunctionalInterface

interface lambaExp{
  void meth1(~~a~~int a, int b);
  }

class lambaExpDemo{
  p s v m (          ){

   lambaExp obj= (a,b)->{

   sout("The value of a & b is:"+a+","+b);

   };
  obj.meth1(5,10);

  }
```

o/p: The value of a & b is: 5,10