C O D E 2 S U C C E E D

Help You Grow
(http://www.code2succeed.com/)

LAMBDAS VS ANONYMOUS INNER CLASSES

September 10, 2016 (Http://Www.Code2succeed.Com/Lambdas-Vs-Anonymous-Inner-Classes/) Kavi (Http://Www.Code2succeed.Com/Author/Kavi/)

Anonymous inner classes are used by java programmer as “ad hoc” functionality i.e where and when they are needed. Since Java 8, we can use lambda expressions instead of anonymous inner classes. It might seems that both inner classes and lambda expressions are similar as they both are used to implement ad hoc functionality. However there are some differences between lambda expression and inner classes which are as follow.

1) Syntax

Syntax of both lambda expression and anonymous inner classes are quite different. The notation for Anonymous inner classes is detailed while for lambda expression, it is concise. Here is the example to demonstrate difference in syntax between anonymous inner classes and lambdas.

```
1  //syntax of anonymous inner class
2  public static void main(String[] args) {
3      Runnable r = new Runnable() {
4          @Override
5          public void run() {
6              System.out.println("in run");
7          }
8      };
9
10     Thread t = new Thread(r);
11     t.start();
12 }
13
14 //syntax of lambda expression
15 public static void main(String[] args) {
16     Runnable r = ()->{System.out.println("in run");};
17     Thread t = new Thread(r);
18     t.start();
19 }
```

2) Variable Binding

Anonymous inner classes have support to access variables from the enclosing context. It has access to all final variables of its enclosing context. Below is an example to demonstrate this

```
public static void main(String[] args) {
    final int cnt = 0;
    Runnable r = new Runnable() {
        @Override
        public void run() {
            System.out.println("in run" + cnt);
        }
    };

    Thread t = new Thread(r);
    t.start();
}
```

The anonymous inner class above has access to cnt variable because it is defined as final in enclosing context. Lambda Expression too has support to variable binding as demonstrated below.

```
public static void main(String[] args) {
    int cnt = 16;
    Thread t = new Thread(r);
    t.start();*/

    Runnable r = ()->{System.out.println("in run"+cnt);};
    Thread t = new Thread(r);
    t.start();
    cnt++; // compilation error
}
```

However, as you can see in above example cnt variable is not a final variable still it can be used inside lambda expresssion. Also, cnt++ throws compilation error. The difference is that the variables from the enclosing context that are used inside a lambda expression are implicitly final. i.e compiler automatically add final keyword before cnt variable.

Note : Since Java 8, the explicit final declaration is no longer needed for anonymous inner classes as well. Both lambda expressions and anonymous inner classes are treated the same regarding variable binding.

3) Scoping

An anonymous inner class is a class, which means that it has scope for variable defined inside the inner class as demonstrate in below example.

```
public static void main(String[] args) {
    final int cnt = 0;
    Runnable r = new Runnable() {
        @Override
        public void run() {
            int cnt = 5;
            System.out.println("in run" + cnt);
        }
    };

    Thread t = new Thread(r);
    t.start();
}
```

However, lambda expression is not a scope of its own, but is part of the enclosing scope. Here is the example to demonstrate the differences

```
1 public static void main(String[] args) {
2     final int cnt = 0;
3     Runnable r = ()->{
4         int cnt = 5; //compilation error
5         System.out.println("in run"+cnt);};
6     Thread t = new Thread(r);
7     t.start();
8 }
```

Both anonymous inner classes and lambda expression have cnt as enclosing context variable. Since anonymous inner classes has its own scope we can define cnt variable inside run method. However lambda expression is not a scope of its own we can't define cnt variable again in lambda expression. In case we try to do so, code will show compilation error as "*Lambda expression's local variable cnt cannot redeclare another local variable defined in an enclosing scope*".

Similar rule applies for super and this keyword when using inside anonymous inner class and lambda expression. In case of anonymous inner class this keyword refers to local scope and super keyword refers to the anonymous class's super class. While in case of lambda expression this keyword refers to the object of the enclosing type and super will refer to the enclosing class's super class.

4) Performance

At runtime anonymous inner classes require class loading, memory allocation and object initialization and invocation of a non-static method while lambda expression is pure compile time activity and don't incur extra cost during runtime. So performance of lambda expression is better as compare to anonymous inner classes.

Stay tuned for more updates !

Related Posts:

1. Convert String to Stream to String (http://www.code2succeed.com/convert-string-stream-string/)
2. Convert String to boolean in java (http://www.code2succeed.com/convert-string-to-boolean-in-java/)
3. User-defined Exceptions (http://www.code2succeed.com/user-defined-exception/)
4. Get keys from HashMap in Java (http://www.code2succeed.com/get-keys-hashmap-java/)
5. Clone method in Java (http://www.code2succeed.com/clone-method-java/)
6. Stringbuffer in java (http://www.code2succeed.com/stringbuffer-in-java/)
7. String vs StringBuffer vs StringBuilder (http://www.code2succeed.com/string-vs-stringbuffer-vs-stringbuilder/)
8. Instance variables and Class variables (static variables) in Java (http://www.code2succeed.com/instance-variables-class-variables-static-variables-java/)
9. Immutability of Strings (http://www.code2succeed.com/immutability-of-strings/)
10. String Comparison in Java (http://www.code2succeed.com/string-comparison-in-java/)

Core Java (Http://Www.Code2succeed.Com/Category/Java/Core-Java/), Java (Http://Www.Code2succeed.Com/Category/Java/), Java (Http://Www.Code2succeed.Com/Category/Interview-Questions/Java-Interview-Questions/)

Core Java (Http://Www.Code2succeed.Com/Tag/Core-Java/)

2 thoughts on “Lambdas vs Anonymous Inner Classes”

Nikhil says:

December 1, 2016 at 6:45 pm (http://www.code2succeed.com/lambdas-vs-anonymous-inner-classes/#comment-78)

appreciate your work!!

Bhuvi.S says:

September 18, 2018 at 11:34 am (http://www.code2succeed.com/lambdas-vs-anonymous-inner-classes/#comment-1498)

Appreciating the persistence you put into your blog and detailed information you provide.

Thanks for one marvelous posting! I enjoyed reading it; you are a great author.