**Lambda Expressions nested blocks**

|  |  |
| --- | --- |
| import **java.util.ArrayList**;  import **java.util.Collections**;  import **java.util.Comparator**;  **public** **class** second {  **public** **static** **void** main(**String**[] args) {  *//WITH LAMBDA EXPRESSION - more about collection class*  **Employee** snow = new Employee("Jon snow", 10);  **Employee** tygerrian = new Employee("Tygerrian", 5);  **Employee** ned = new Employee("Ned Stark", 25);  **Employee** lenisters = new Employee("Kingslayer", 35);  **ArrayList**<**Employee**> employees = new **ArrayList**<>();  employees.add(snow);  employees.add(tygerrian);  employees.add(ned);  employees.add(lenisters);  *//with lambda implementation*  Collections.sort(employees, (emps1, emps2) **->**  emps1.getName().compareTo(emps2.getName()));  for (**Employee** emp : employees){  System.out.println(emp.getName());  }  *//this gonna concat and make it upper case as well,*  *//using return, NESTED*  upper up = (s1, s2) **->** {  **String** results = s1.toUpperCase() + s2.toUpperCase();  return results;  };  **String** sillystring = dotheinterface(up, employees.get(0).getName(), employees.get(1).getName());  System.out.printf(sillystring);  }  *//takes an object for the instance*  **public** **final** **static** **String** dotheinterface(upper u, **String** s1, **String** s2){  return u.upperConcat(s1, s2);  }  }  **class** Employee{  **String** name;  **int** experience;  *//constructor*  **public** Employee(**String** name, **int** experience) {  this.name = name;  this.experience = experience;  }  *//getter and setter*  **public** **String** getName() {  return name;  }  **public** **void** setName(**String** name) {  this.name = name;  }  **public** **int** getExperience() {  return experience;  }  **public** **void** setExperience(**int** experience) {  this.experience = experience;  }  }  **interface** upper{  **public** **String** upperConcat(**String** s1, **String** s2);  } | **Jon snow**  **Kingslayer**  **Ned Stark**  **Tygerrian**  **JON SNOWKINGSLAYER** |

**LAMBDA CALLED FROM A CLASS**

|  |  |
| --- | --- |
| import **java.util.ArrayList**;  import **java.util.Collections**;  import **java.util.Comparator**;  **public** **class** second {  **public** **static** **void** main(**String**[] args) {  *//WITH LAMBDA EXPRESSION - more about collection class*  **Employee** snow = new Employee("Jon snow", 10);  **Employee** tygerrian = new Employee("Tygerrian", 5);  **Employee** ned = new Employee("Ned Stark", 25);  **Employee** lenisters = new Employee("Kingslayer", 35);  **ArrayList**<**Employee**> employees = new **ArrayList**<>();  employees.add(snow);  employees.add(tygerrian);  employees.add(ned);  employees.add(lenisters);  Another**class** an = new Anotherclass();  String s = an.dosomething();  System.out.println(s);  }  *//takes an object for the instance*  **public** **final** **static** **String** dotheinterface(upper u, **String** s1, **String** s2){  return u.upperConcat(s1, s2);  }  }  **interface** upper{  **public** **String** upperConcat(**String** s1, **String** s2);  }  **class** Anotherclass{  **public** **String** dosomething(){  return second.dotheinterface(new upper(){  **public** **String** upperConcat(**String** s1, **String** s2){  return s1.toUpperCase() + s2.toUpperCase();  }  }, "String1", "String2");  }  } | **PS F:\java> javac second.java**  **PS F:\java> java second**  **STRING1STRING2** |

**Getting the names of the class – without using lambda expression**

|  |  |
| --- | --- |
| **public** **class** second {  **public** **static** **void** main(**String**[] args) {  Another**class** an = new Anotherclass();  String s = an.dosomething();  System.out.println(s);  }  *//takes an object for the instance*  **public** **final** **static** **String** dotheinterface(upper u, **String** s1, **String** s2){  return u.upperConcat(s1, s2);  }  }  **interface** upper{  **public** **String** upperConcat(**String** s1, **String** s2);  }  **class** Anotherclass{  **public** **String** dosomething(){  System.out.println("name of the Anotherclass " + getClass().getSimpleName());  return second.dotheinterface(new upper(){  **public** **String** upperConcat(**String** s1, **String** s2){  *//anonymous class doesnot have a name that is why, no output will come*  System.out.println("name of the anonymous " + getClass().getSimpleName());  return s1.toUpperCase() + s2.toUpperCase();  }  }, "String1", "String2");  }  } | **PS F:\java> java second**  **name of the Anotherclass Anotherclass**  **name of the anonymous**  **STRING1STRING2** |

**With lambda expression**

|  |  |
| --- | --- |
| **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Anotherclass an = **new** Anotherclass();  String s = an.dosomething();  System.***out***.println(s);  }  *//takes an object for the instance* **public final static** String dotheinterface(upper u, String s1, String s2){  **return** u.upperConcat(s1, s2);  } }  **interface** upper{  **public** String upperConcat(String s1, String s2); }  **class** Anotherclass{  **public** String dosomething(){  upper uc = (s1, s2) -> {  System.***out***.println(**"the name of the anonymous class is : "** + getClass().getName());  String result = s1.toUpperCase() + s2.toUpperCase();  **return** result;  };  System.***out***.println(**"the name of the Another class is : "** + getClass().getName());  **return** Main.*dotheinterface*(uc, **"string1"**, **"string2"**);  } } | **the name of the Another class is : com.company.Anotherclass**  **the name of the anonymous class is : com.company.Anotherclass**  **STRING1STRING2** |

**Using a nested code block**

|  |  |
| --- | --- |
| package com.company;  public class Main {   public static void main(String[] args) {  Anotherclass an = new Anotherclass();  String s = an.dosomething();  System.*out*.println(s);  }  *//takes an object for the instance* public final static String dotheinterface(upper u, String s1, String s2){  return u.upperConcat(s1, s2);  } }  interface upper{  public String upperConcat(String s1, String s2); }  class Anotherclass{  public String dosomething(){  {  upper u = new upper() {  @Override  public String upperConcat(String s1, String s2) {  return s1.toUpperCase() + s2.toUpperCase();  }  };  System.*out*.println("name of the class is : " + getClass().getName());  return Main.*dotheinterface*(u, "String1", "String2");  }  } } | **name of the class is : com.company.Anotherclass**  **STRING1STRING2** |

**If we want to use a LOCAL VARIABLE outside the anonymous class, and we need to use a FINALLY**

|  |  |
| --- | --- |
| **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Anotherclass an = **new** Anotherclass();  String s = an.dosomething();  System.***out***.println(s);  }  *//takes an object for the instance* **public final static** String dotheinterface(upper u, String s1, String s2){  **return** u.upperConcat(s1, s2);  } }  **interface** upper{  **public** String upperConcat(String s1, String s2); }  **class** Anotherclass{  **public** String dosomething(){  **int** i = 0;  {  upper u = **new** upper() {  @Override  **public** String upperConcat(String s1, String s2) {  **return** s1.toUpperCase() + s2.toUpperCase();  }  };  System.***out***.println(**"name of the class is : "** + getClass().getName());  i++;  System.***out***.println(**"i from the codeblock : "** + i);  **return** Main.*dotheinterface*(u, **"String1"**, **"String2"**);  }  } }  **name of the class is : com.company.Anotherclass**  **i from the codeblock : 1**  **STRING1STRING2** | **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Anotherclass an = **new** Anotherclass();  String s = an.dosomething();  System.***out***.println(s);  }  *//takes an object for the instance* **public final static** String dotheinterface(upper u, String s1, String s2){  **return** u.upperConcat(s1, s2);  } }  **interface** upper{  **public** String upperConcat(String s1, String s2); }  **class** Anotherclass{  **public** String dosomething(){  **int** i = 0;  {  upper u = **new** upper() {  @Override  **public** String upperConcat(String s1, String s2) {  System.***out***.println(**"from anonymous class i "** + i);  **return** s1.toUpperCase() + s2.toUpperCase();  }  };  System.***out***.println(**"name of the class is : "** + getClass().getName());  i++;  System.***out***.println(**"i from the codeblock : "** + i);  **return** Main.*dotheinterface*(u, **"String1"**, **"String2"**);  }  } }  **Error:(27, 68) java: local variables referenced from an inner class must be final or effectively final** |

**Solving the error using finally**

|  |  |
| --- | --- |
| **package** com.company;  **public class** Main {   **public static void** main(String[] args) {  Anotherclass an = **new** Anotherclass();  String s = an.dosomething();  System.***out***.println(s);  }  *//takes an object for the instance* **public final static** String dotheinterface(upper u, String s1, String s2){  **return** u.upperConcat(s1, s2);  } }  **interface** upper{  **public** String upperConcat(String s1, String s2); }  **class** Anotherclass{  **public** String dosomething(){  **int** i = 0;  upper u = (s1, s2) -> {  System.***out***.println(**" name of the class is : "** + getClass().getName());  System.***out***.println(**"i = "** + i);  String result = s1.toUpperCase() + s2.toUpperCase();  **return** result;  };   *//i++; //you cant re initialize it here* System.***out***.println(**"i = "** + i);  System.***out***.println(**"class name is : "** + getClass().getName());  **return** Main.*dotheinterface*(u, **"String1"**, **"String2"**);  } } | **i = 0**  **class name is : com.company.Anotherclass**  **name of the class is : com.company.Anotherclass**  **i = 0**  **STRING1STRING2** |