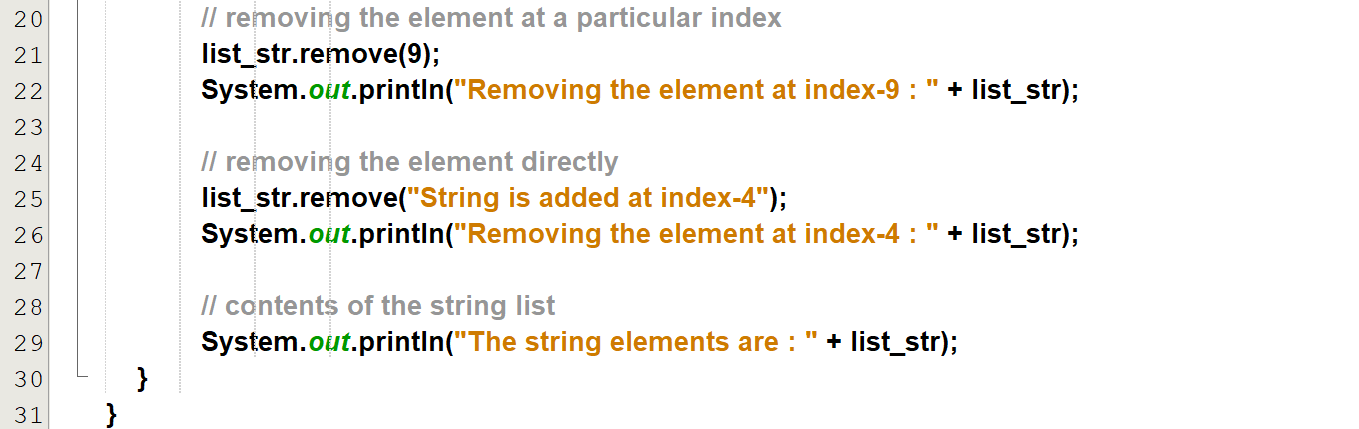
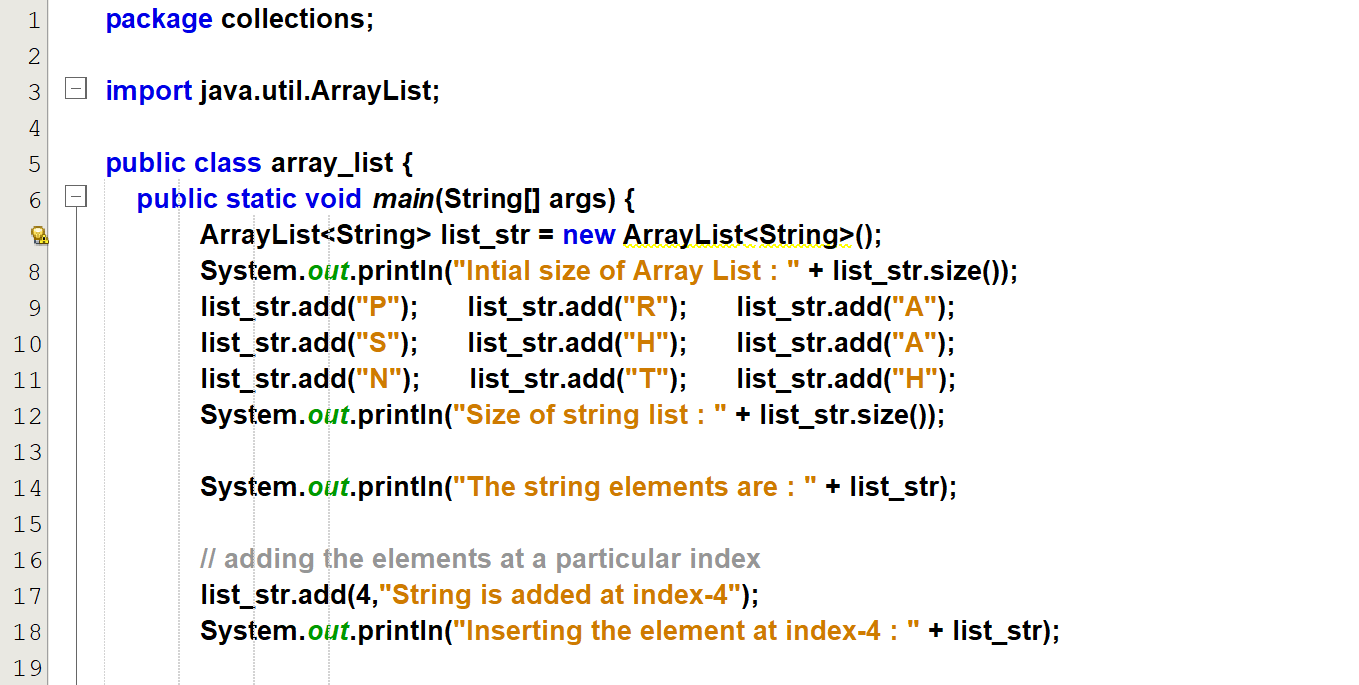
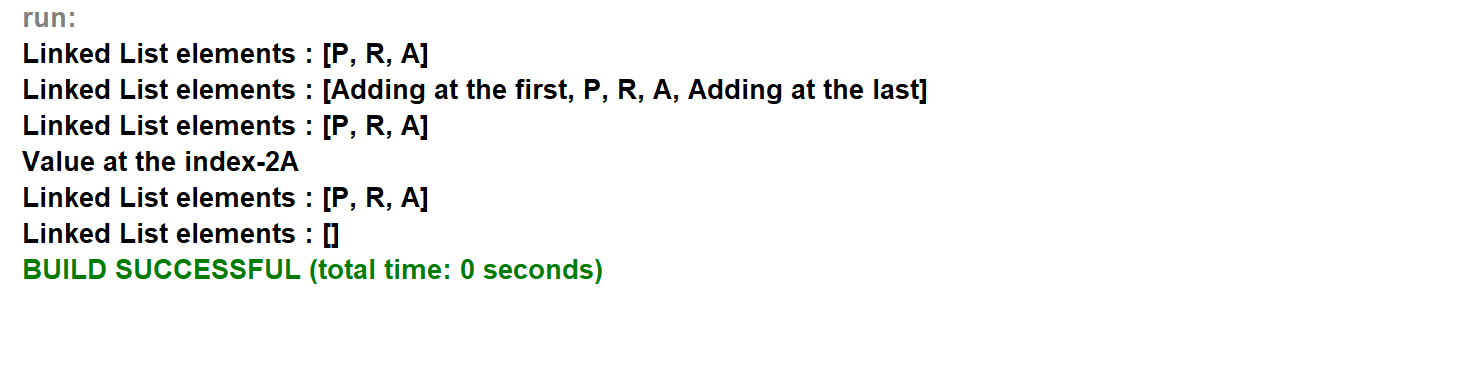
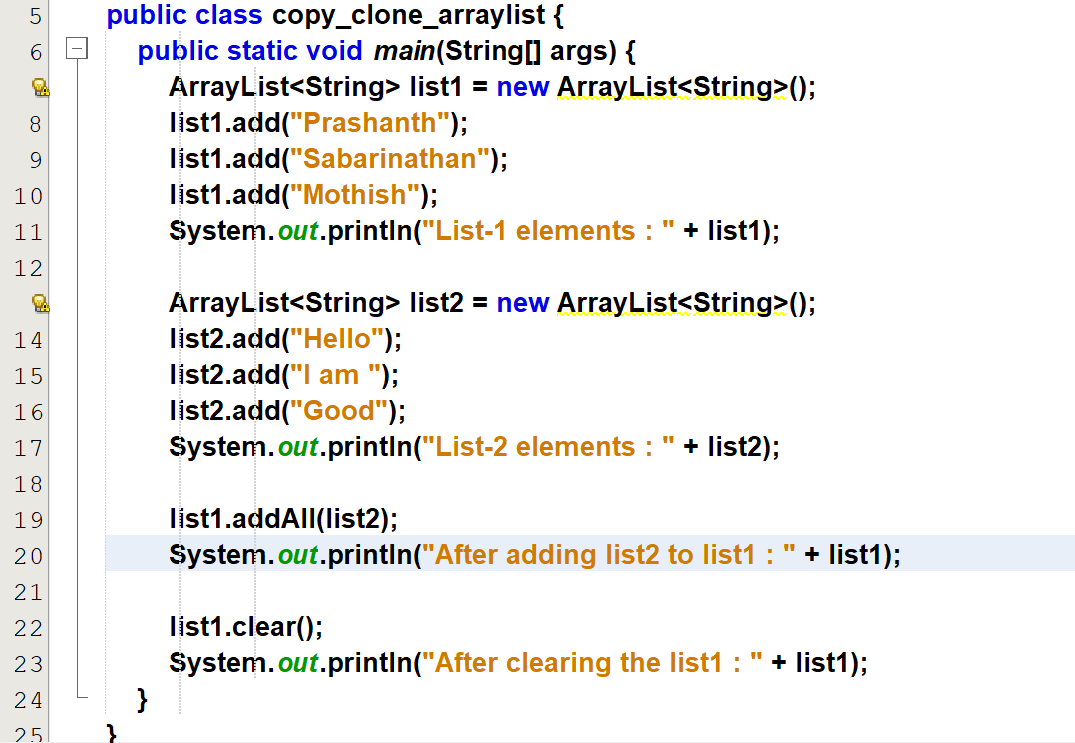
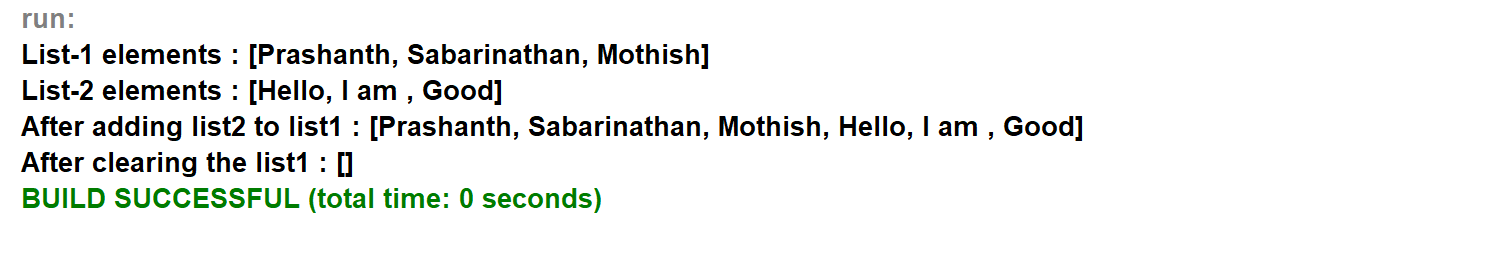
# **Refer ArrayList.java**



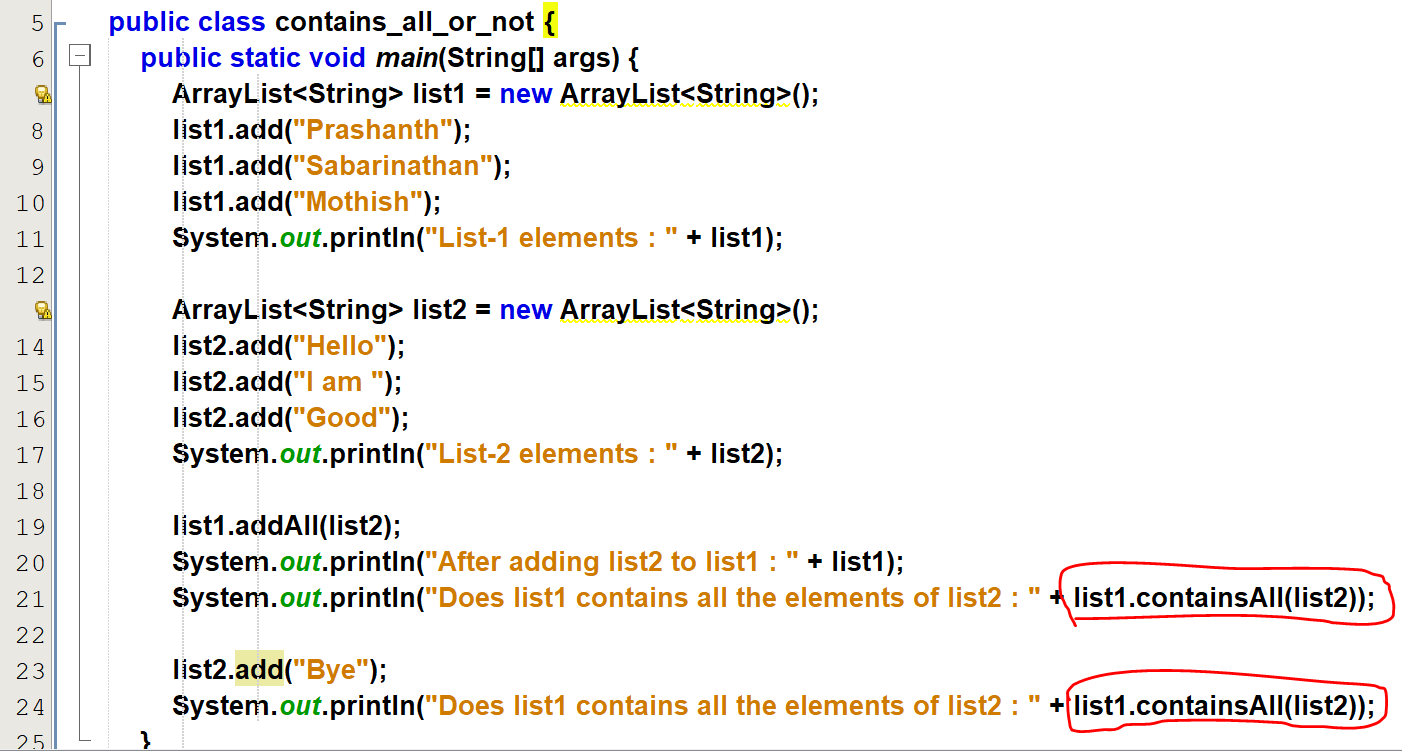
Output:  


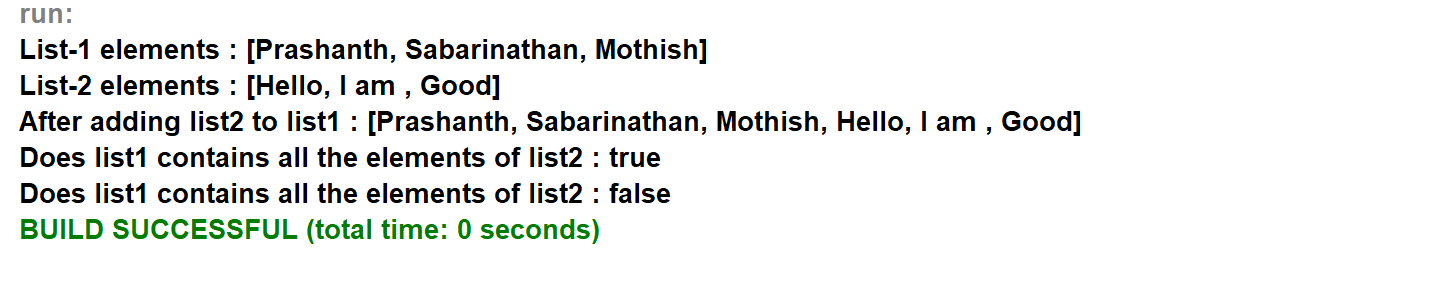
# **Refer copy\_clone\_ArrayList.java**



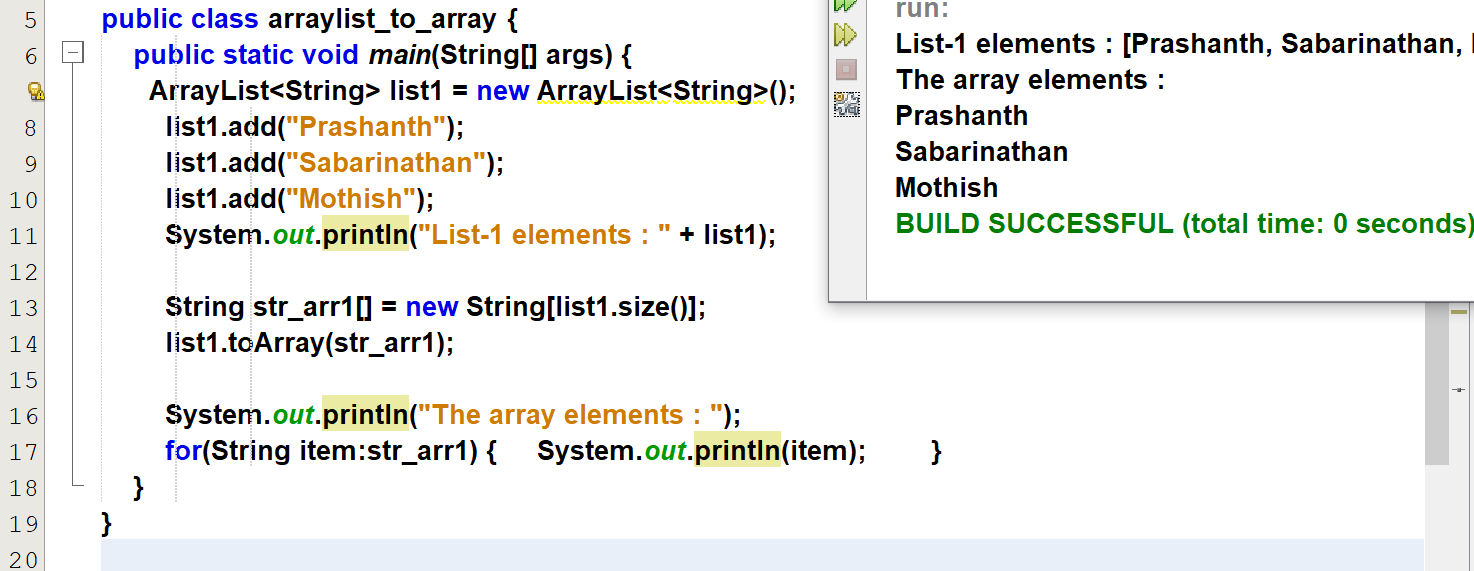
Output:  


# **Refer contains\_all\_or\_not.java**

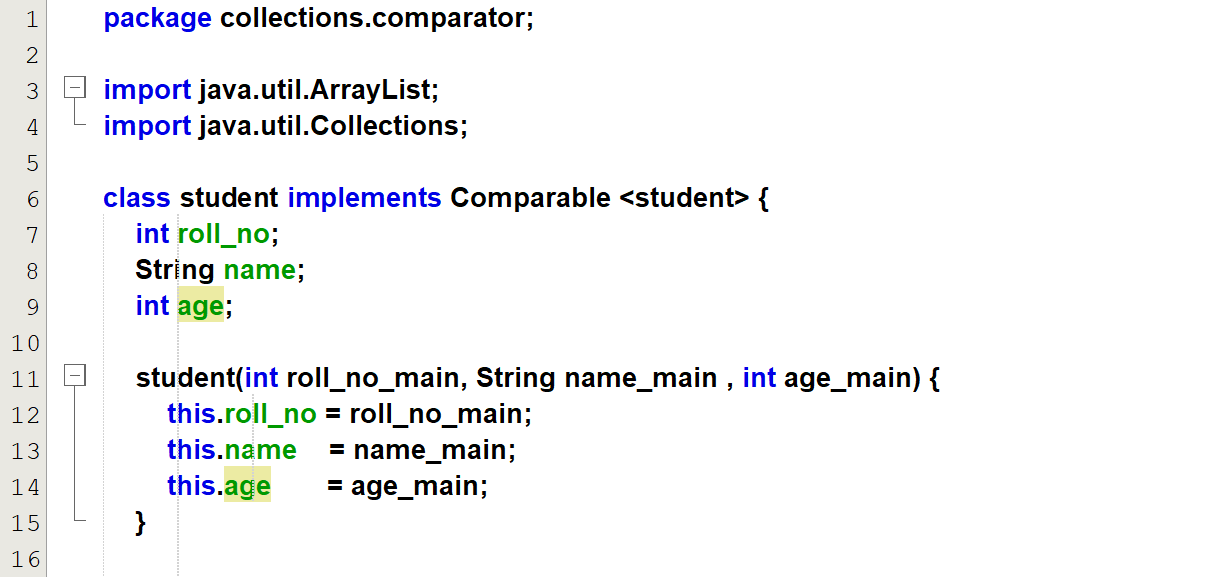
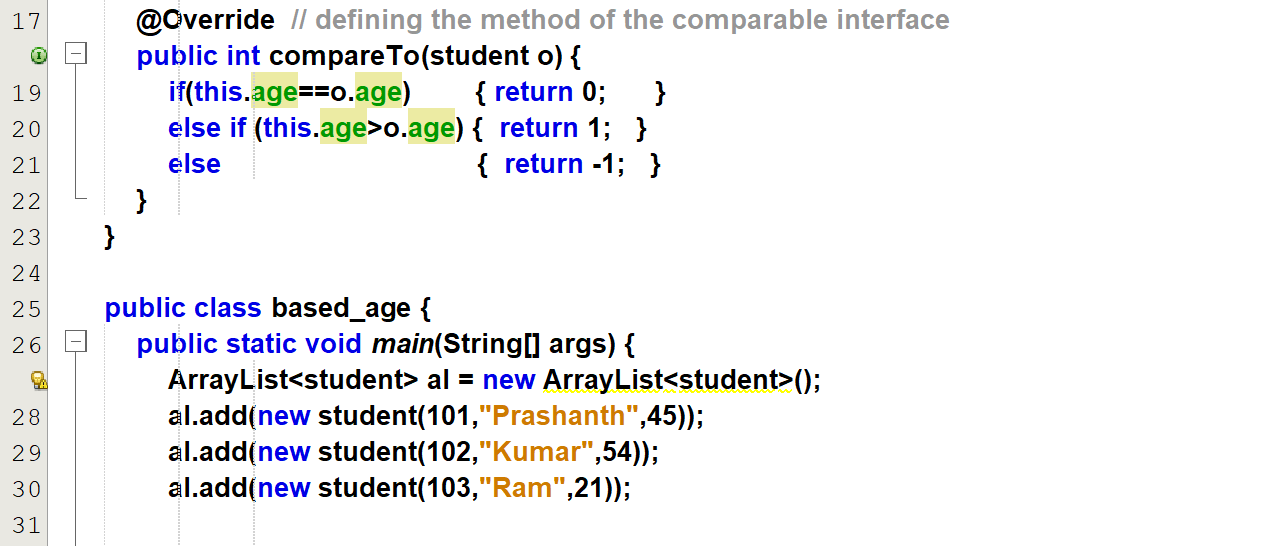
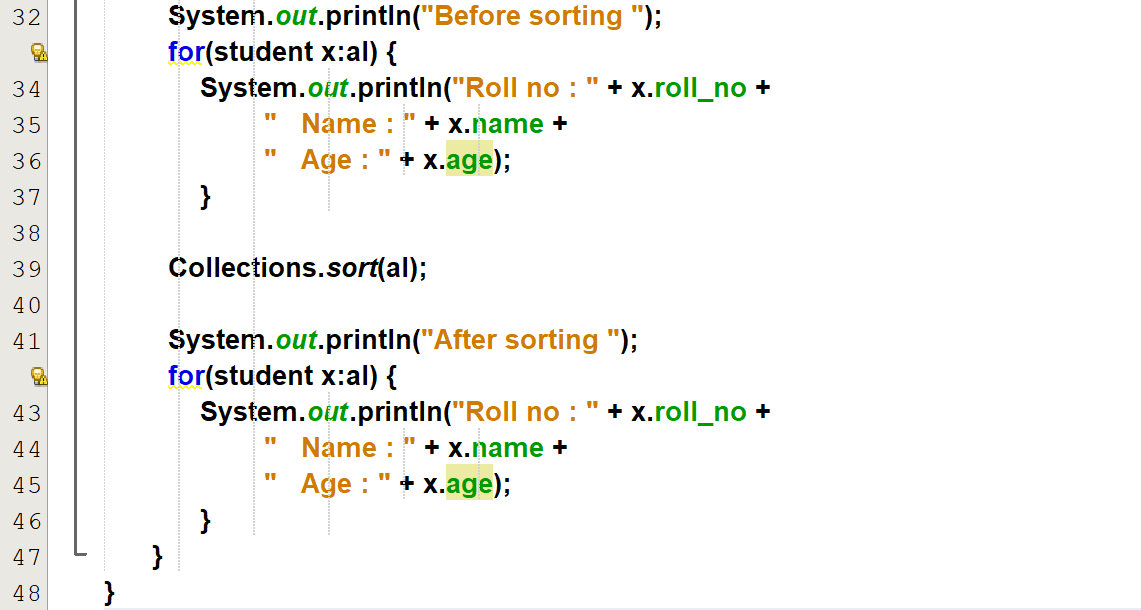


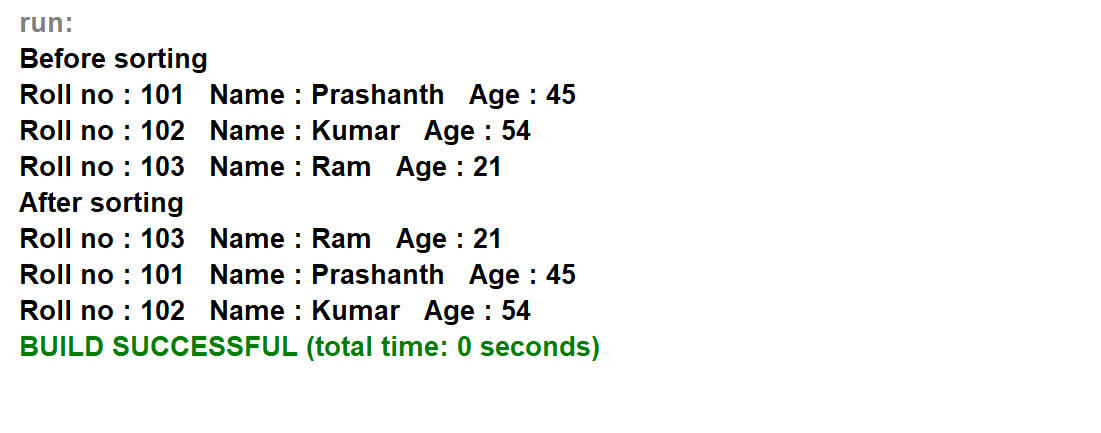
Output:  


# **Refer arraylist\_to\_array.java**

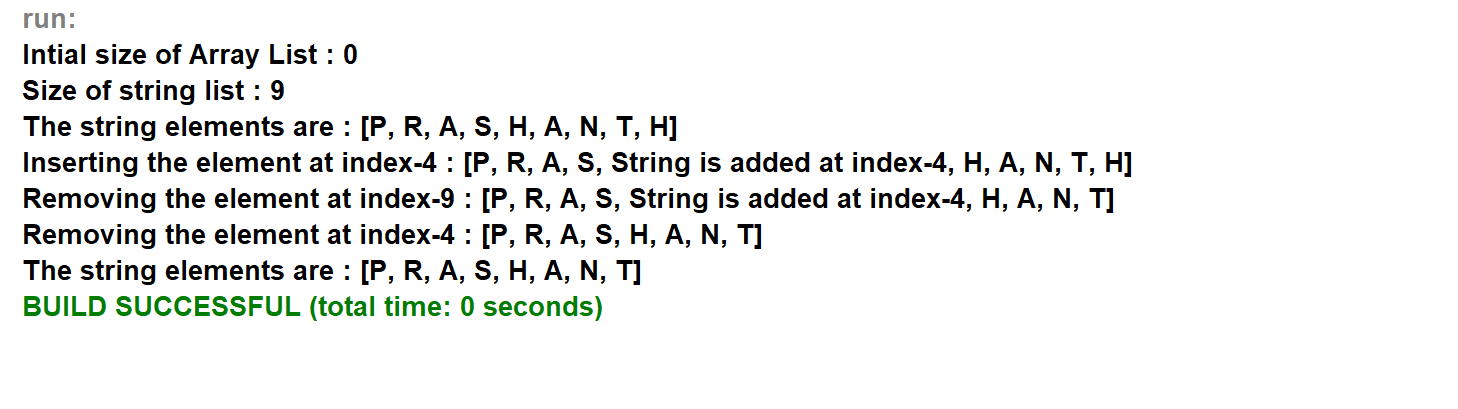


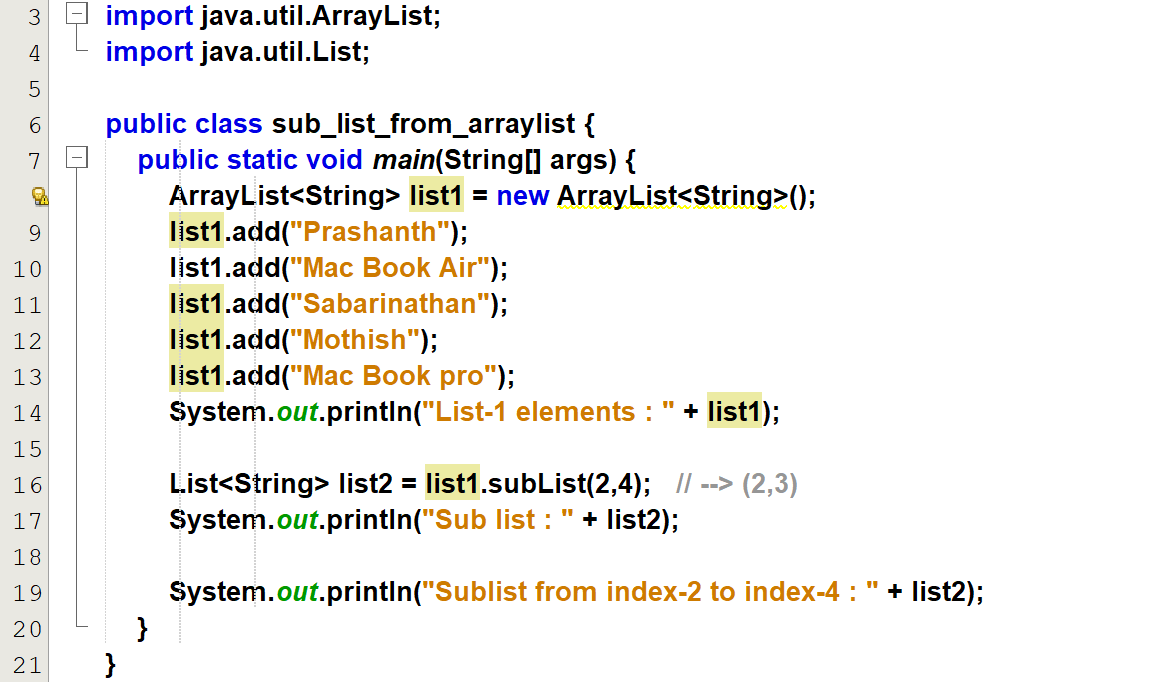
# **Refer swap\_two\_elements.java**

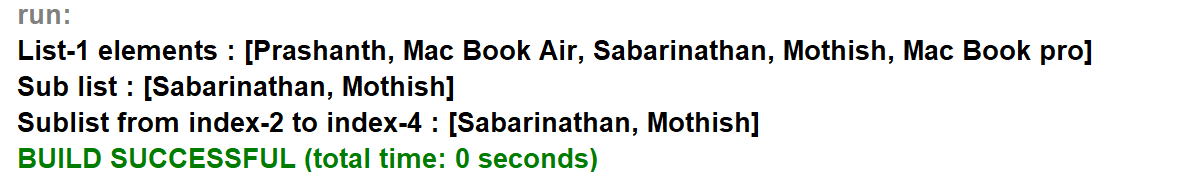
  
  


Output:  


# **Refer linked\_list.java**

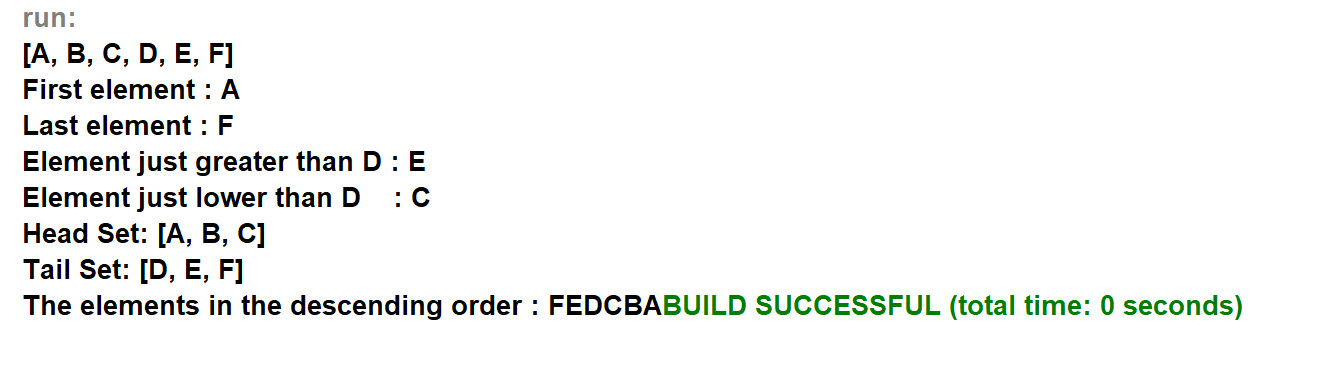
Output:

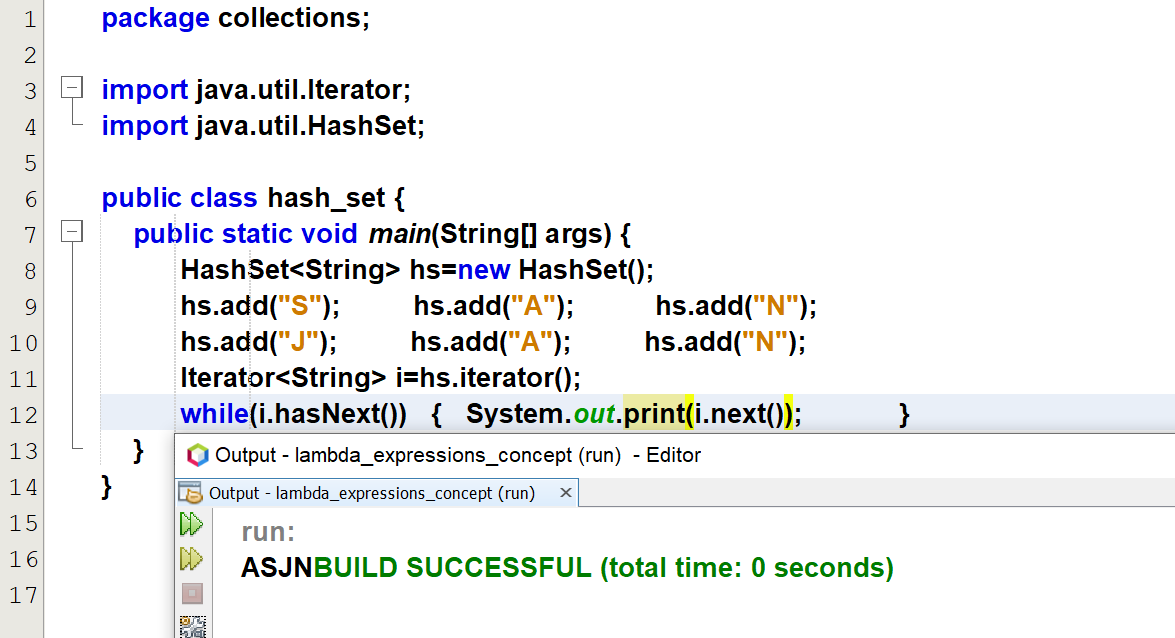
**Refer sub\_list\_from\_ArrayList.java**



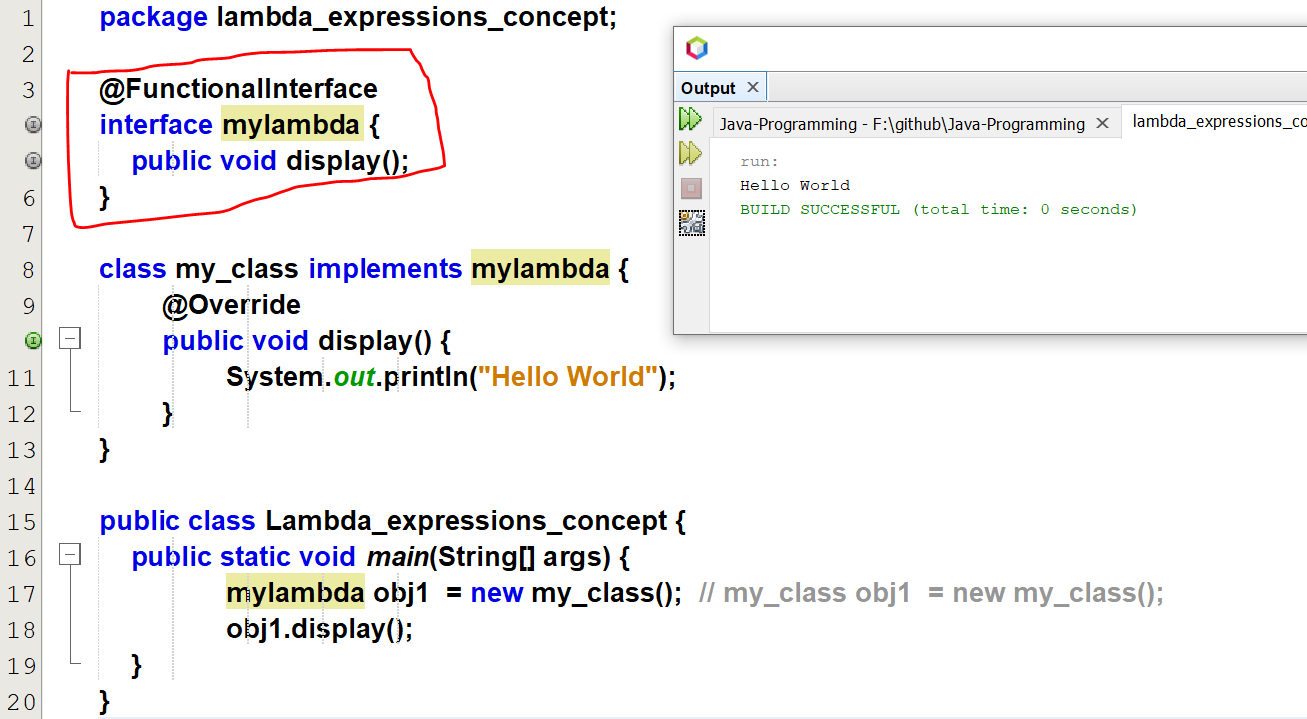
# **Refer tree\_set.java**

Output:

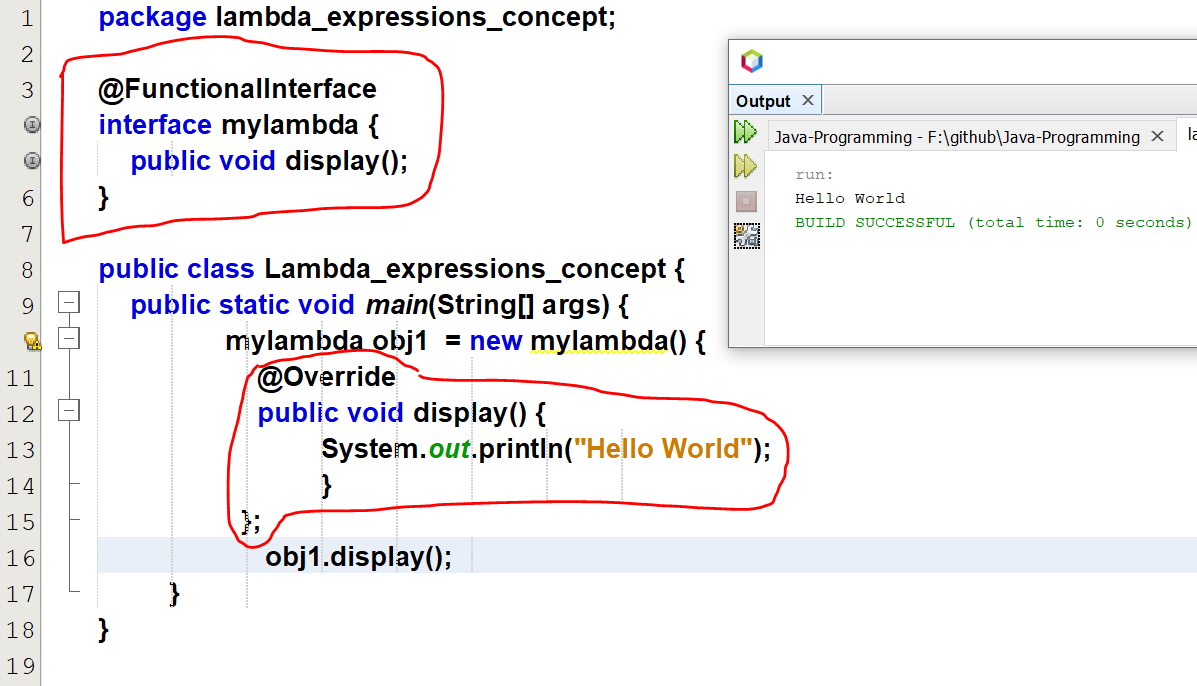


**Refer hash\_set.java**

@FunctionalInterface 🡪 A single function inside a interface.



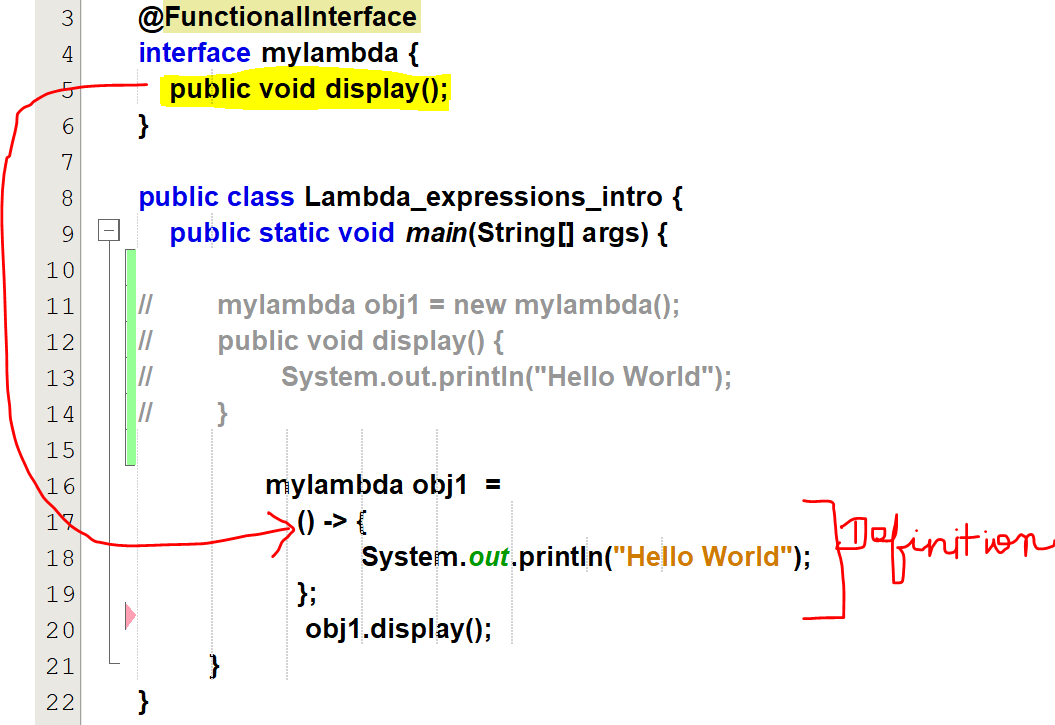
Using anonymous inner class.



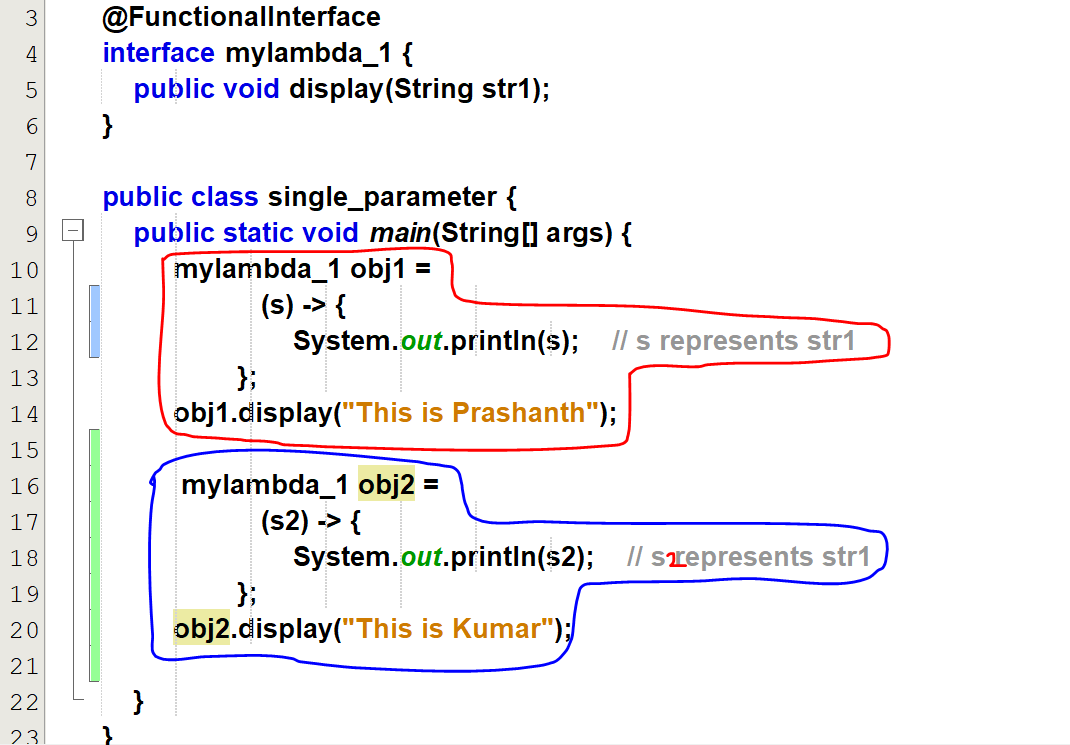
# **Refer Lambda\_expressions\_intro.java**

Lambda expression

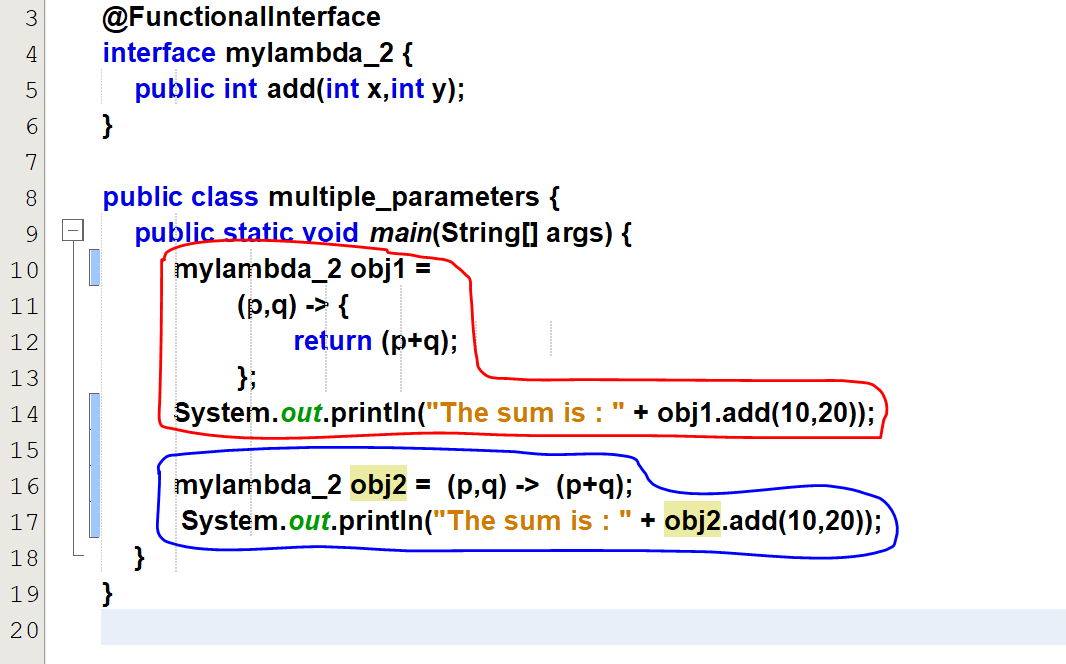
1. No need of right side to equal. ( new mylambda() )
2. No need of method name. ( public void display() )

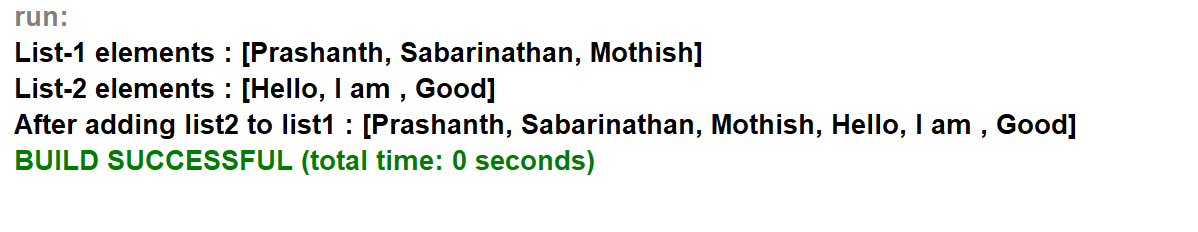


# **Refer single\_parameter.java**

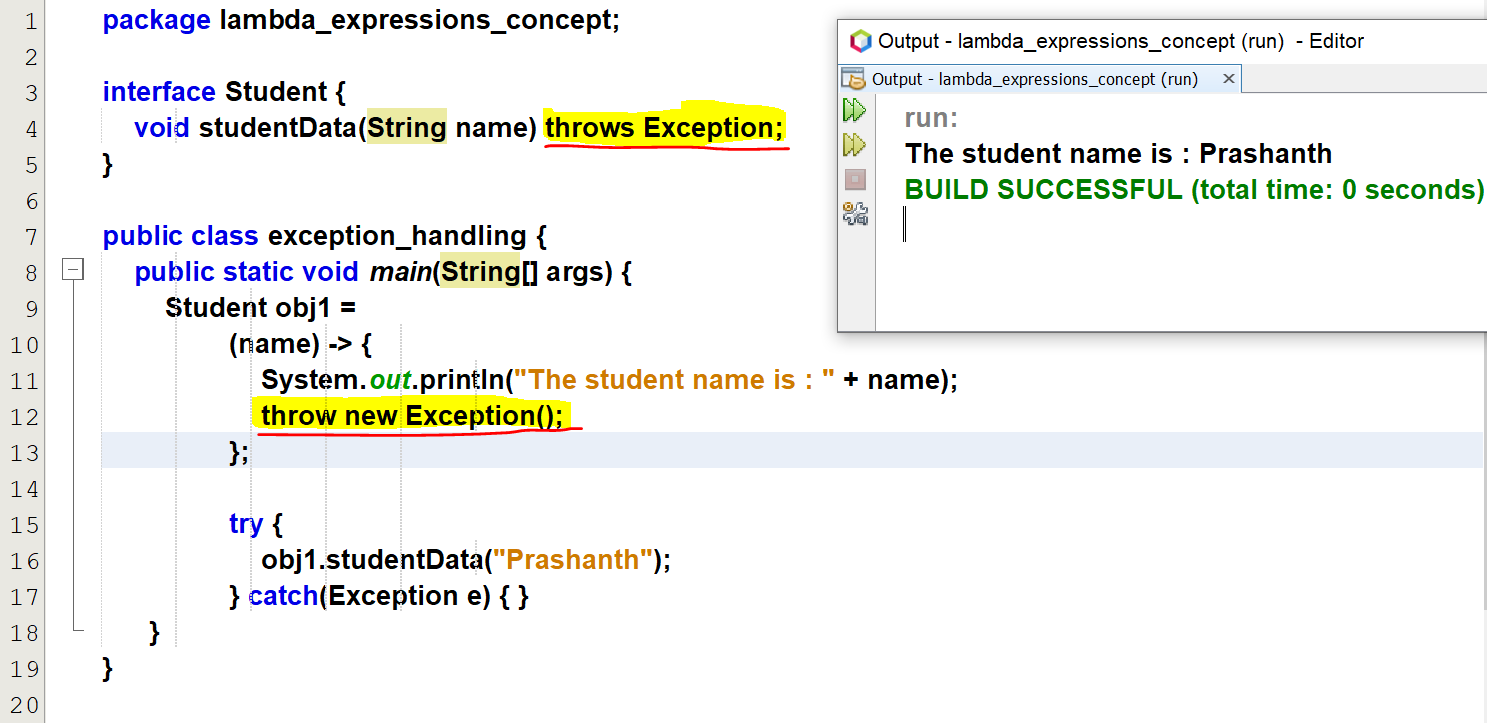


# **Refer multiple\_parameters.java**



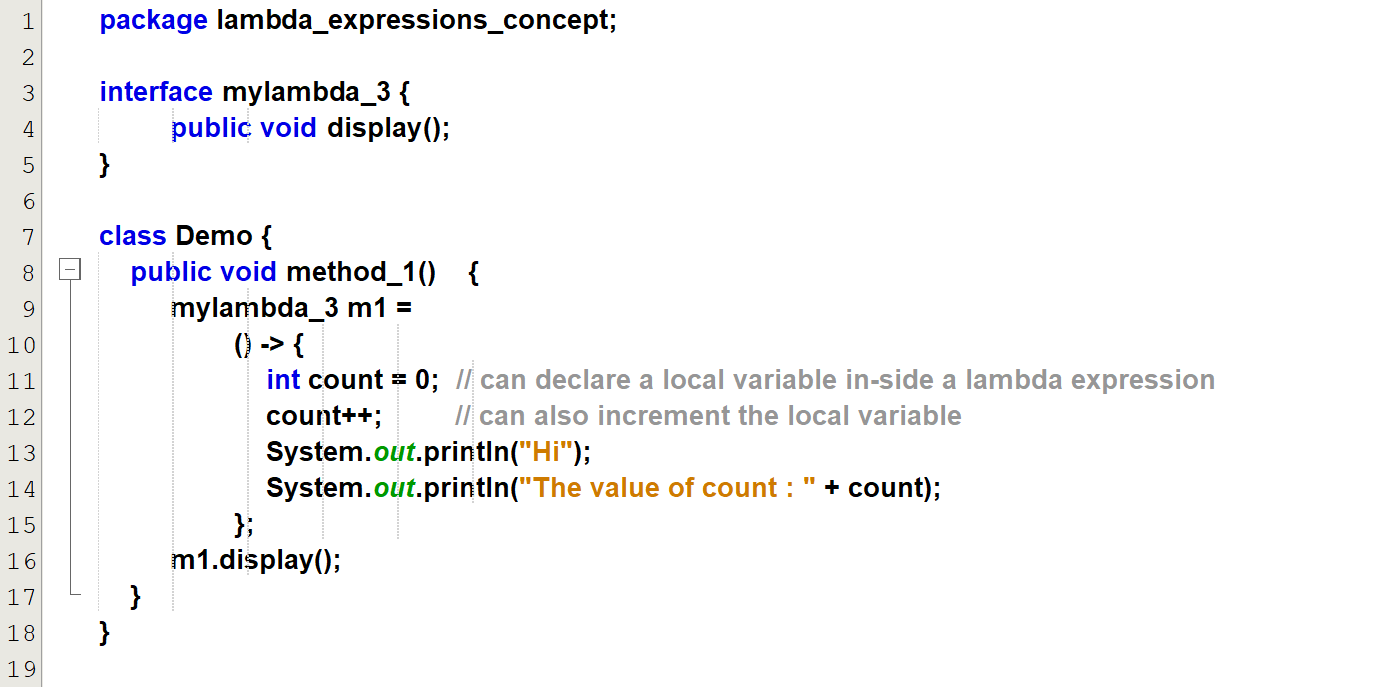
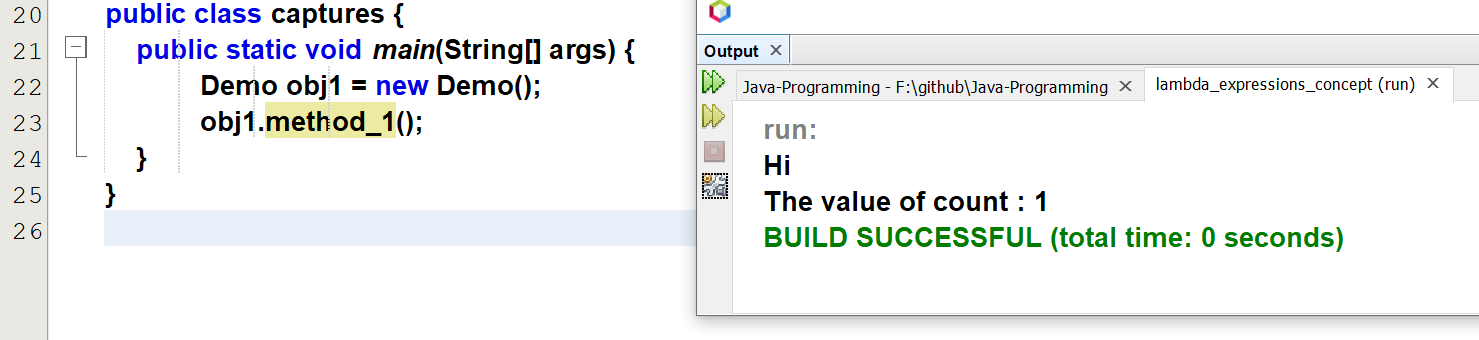
Output:  


# **Refer exception\_handling.java**



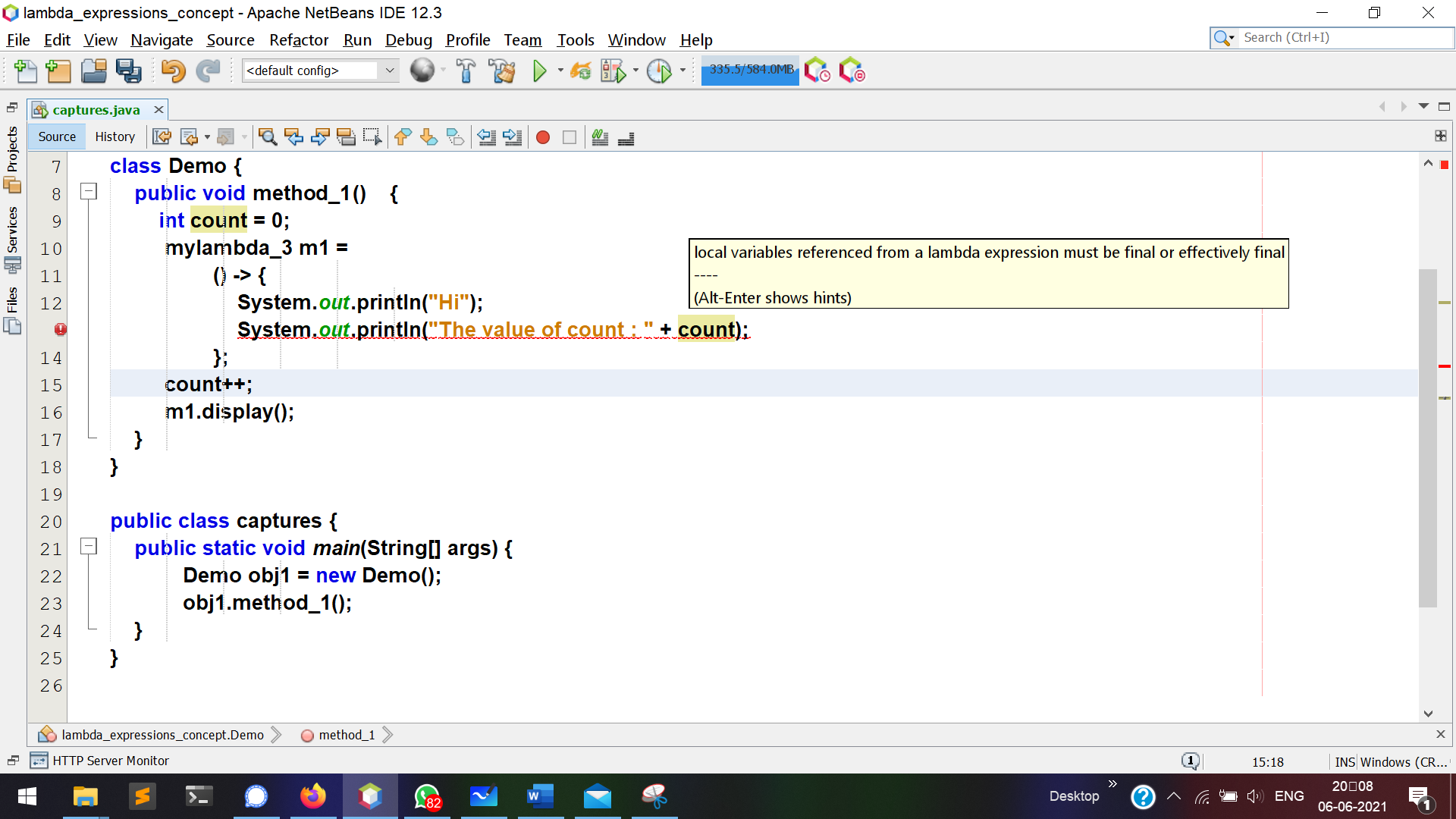
# **Refer captures.java**

A lambda expression can have its own variable and can be accessed within it.

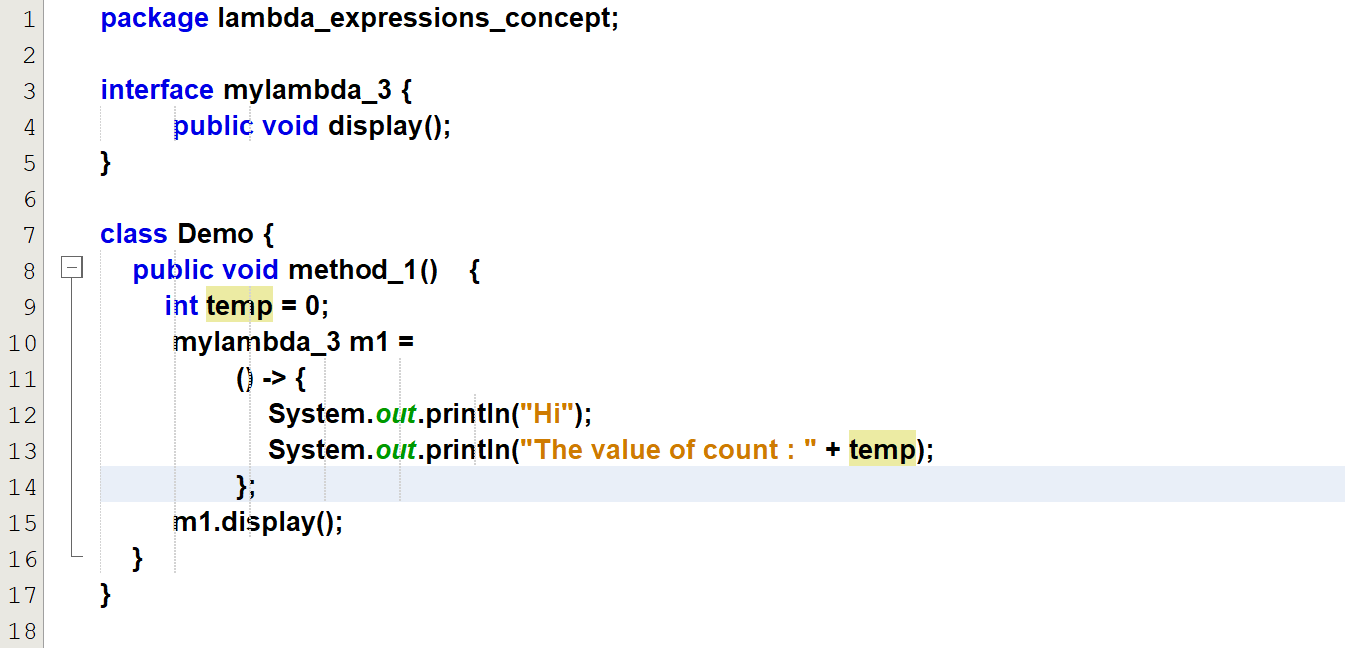
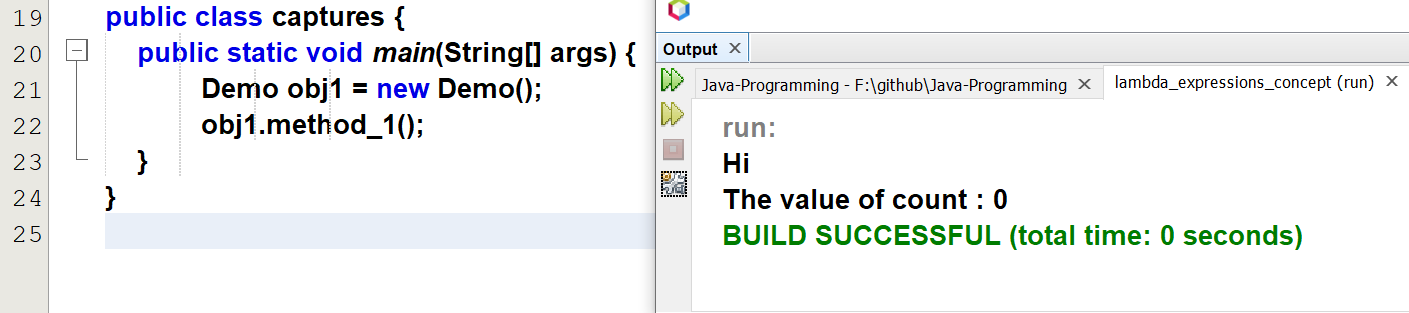
  


A lambda expression can access the local variable (or) capture the local variable only if they are final / they are never modified inside the method.

A lambda expression can access the variables of a method in which this expression is defined but those variables must be final/effectively final.



A lambda expression can access/modify the instance variable (i.e temp) , that instance variable may or may not be final.

# **Refer lambda\_expression\_asArguments.java**

