Lesson 01 Demo 06

Reading Documents in MongoDB CRUD Operations

Objective: To read documents from MongoDB by executing the application and fetching a document based on the e-mail

Tools required: Eclipse IDE

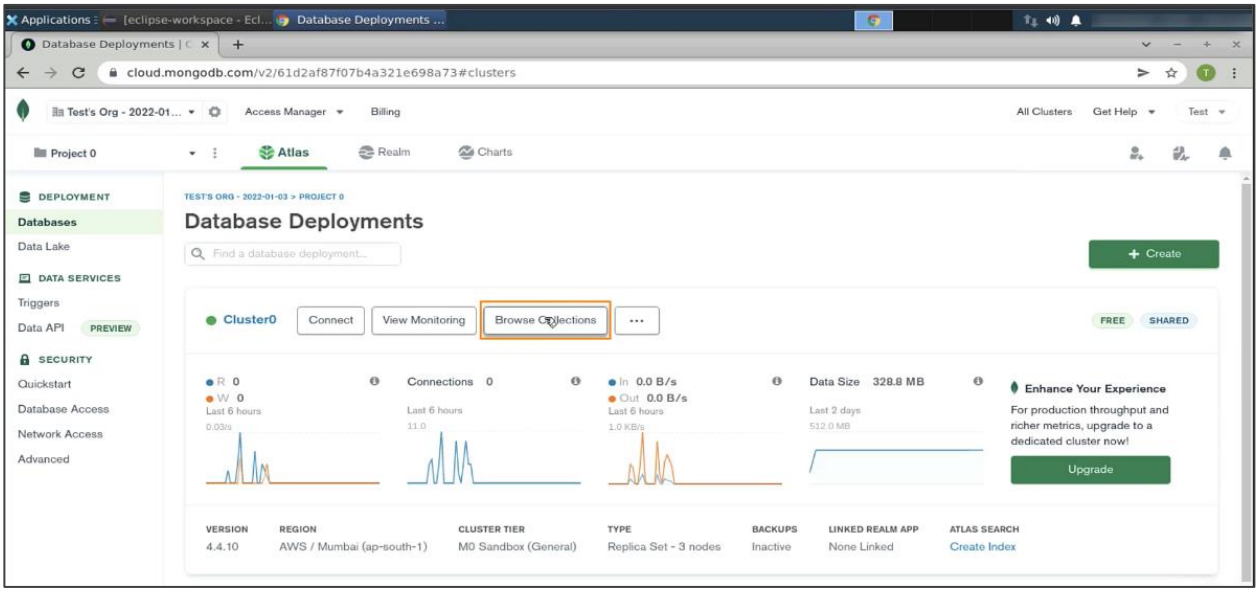
Prerequisites: None

Steps to be followed:

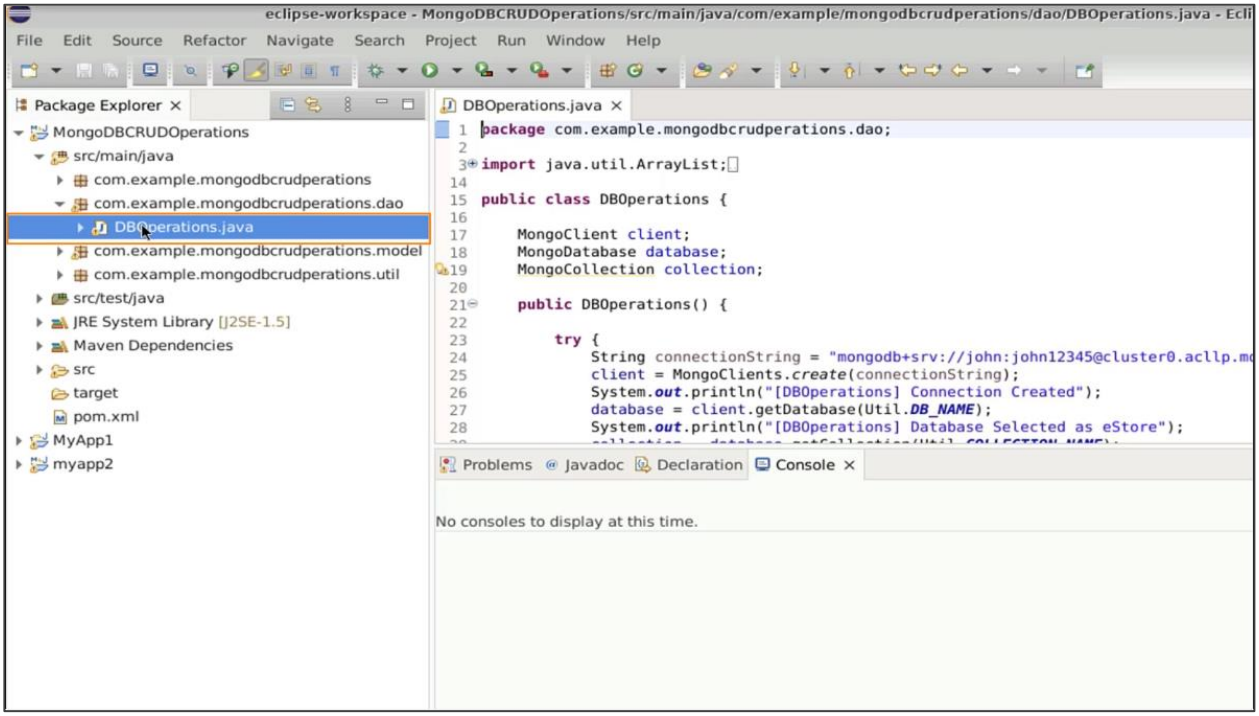
1. Execute the App.java file
2. Fetch the document

Step 1: Execute the App.java file

1.1 Go to the browser, navigate to the **MongoDB** database, and select the **Browse Collections** option

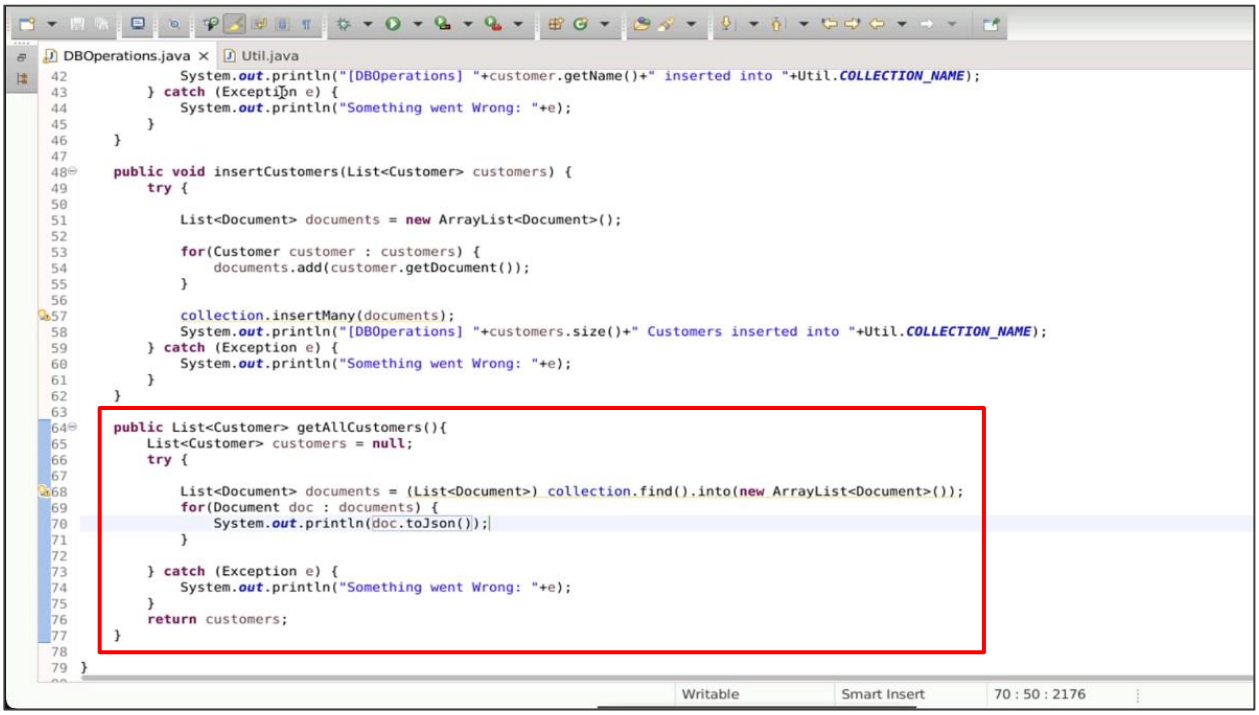


1.2 Under the **MongoDBCRUDOperations** project, open the **DBOperations.java** file

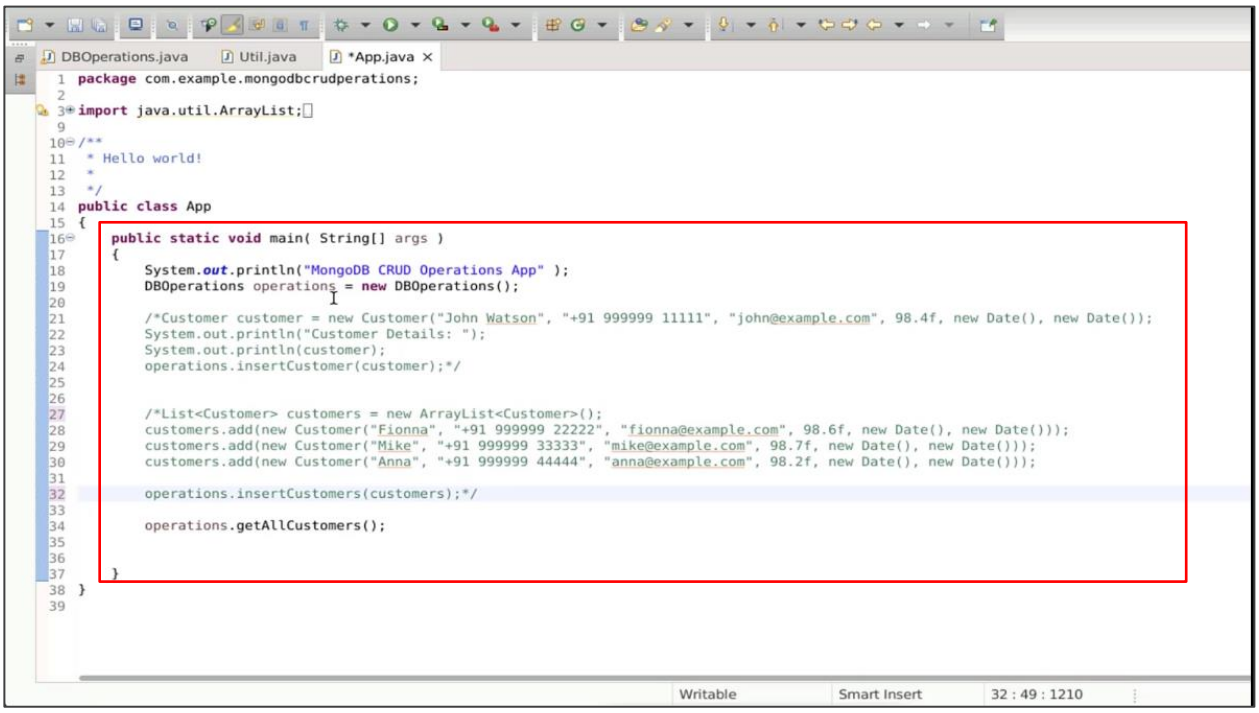


Note: Please refer to the previous demo on how to create the **MongoDBCRUDOperations** project

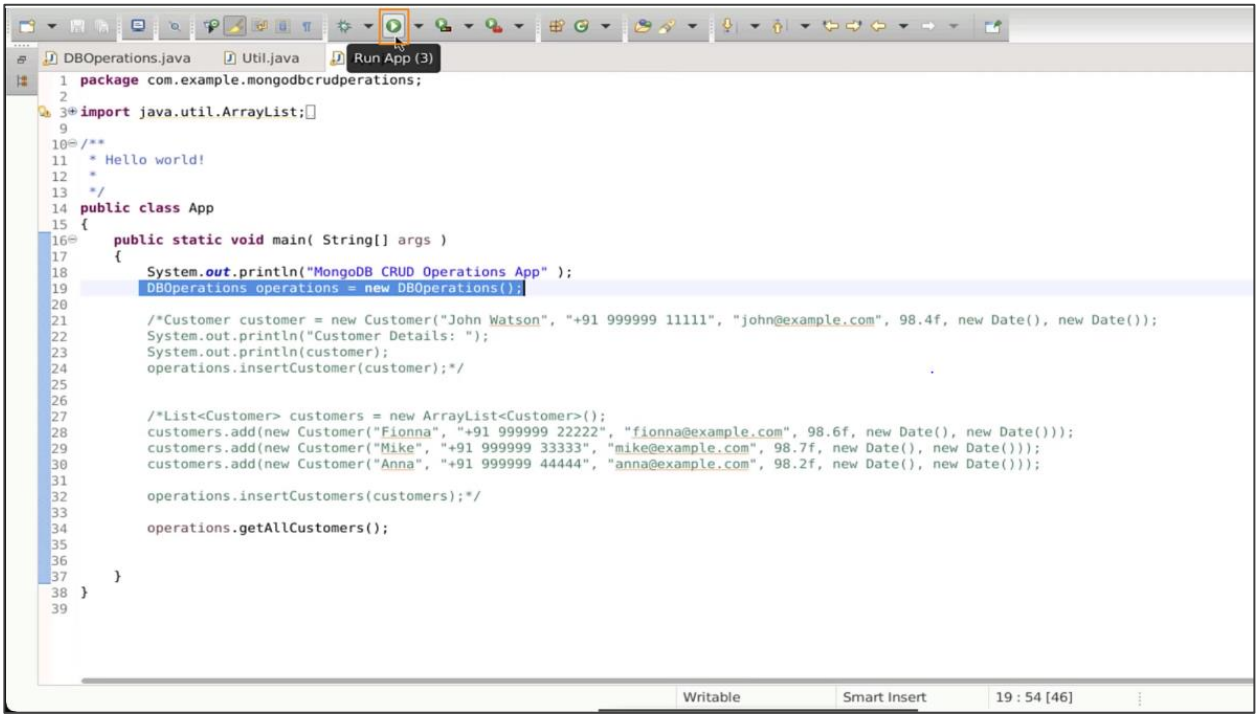
1.3 Write the following lines of code (from lines 64 to 76) to create a method **Customer** to return the list of customers:

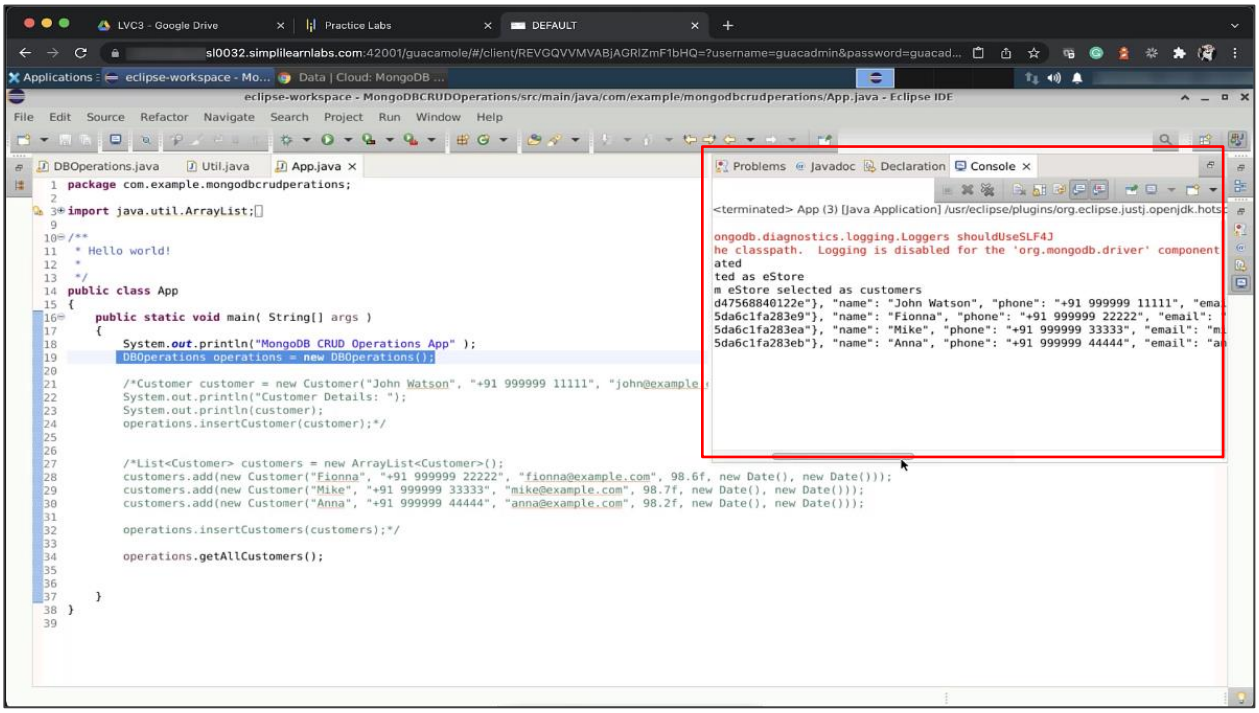


1.4 Go to the **App.java** file, write the following line of code (line 34) to call the **getAllCustomers** method, and then comment out the following lines of code (from lines 21 to 32):

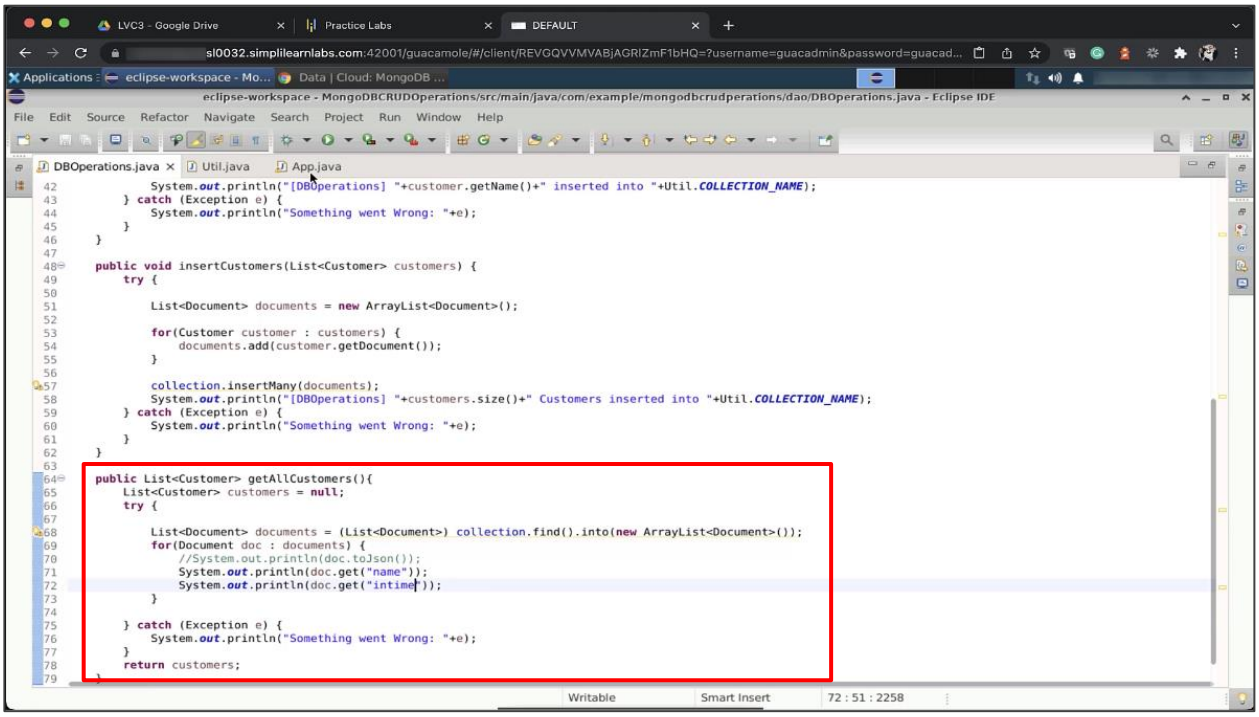


1.5 Now, run the code. You should see the following output with all the document data:

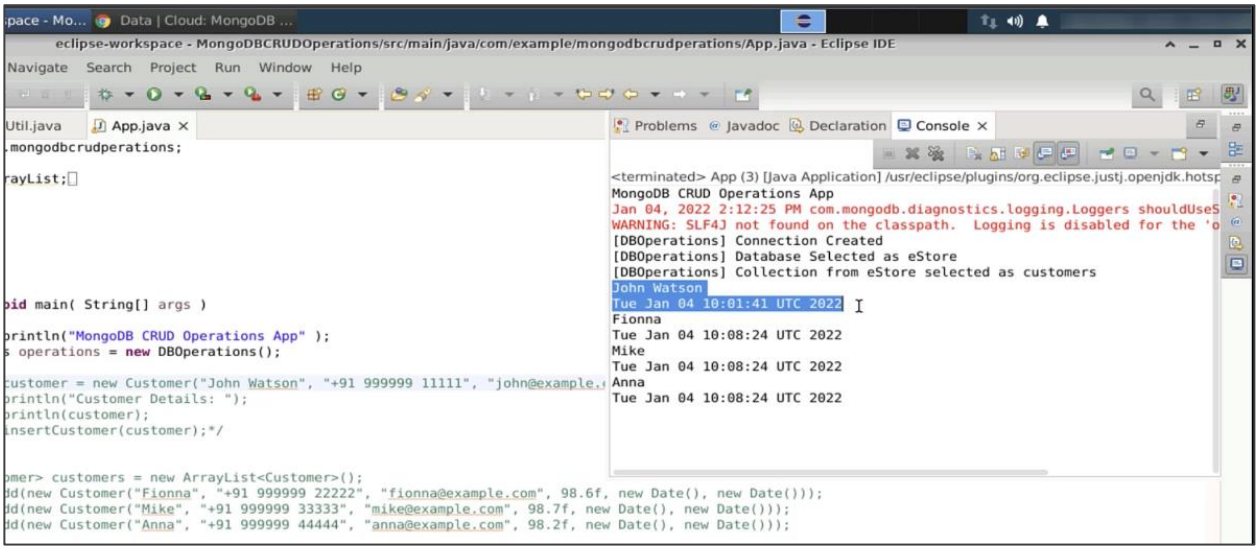




1.6 Go to the **DBOperations.java** file and write the following lines of code (lines 71, 72) to convert the documents into the **customer** data type:



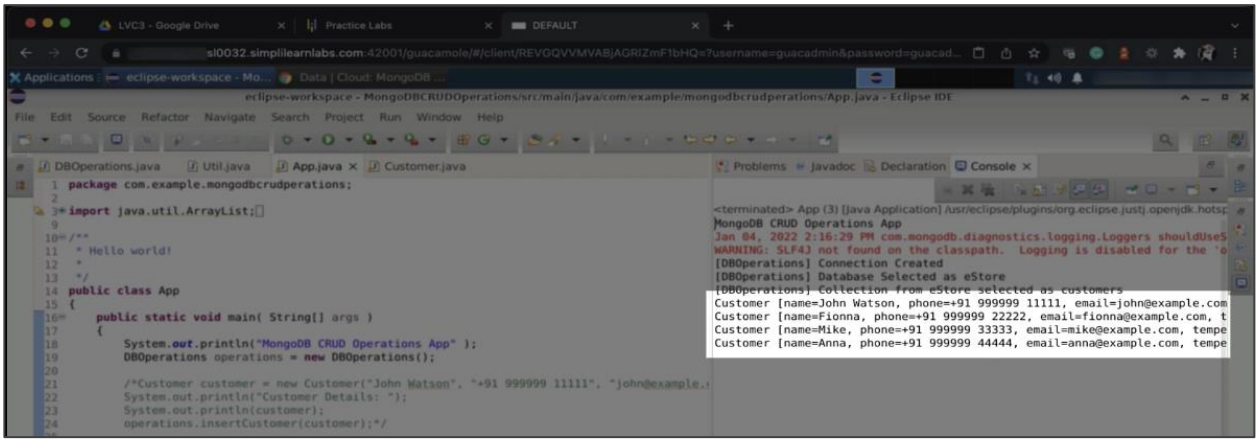
1.7 Go to the **App.java** file and run the application. You can see the entire details of customers.

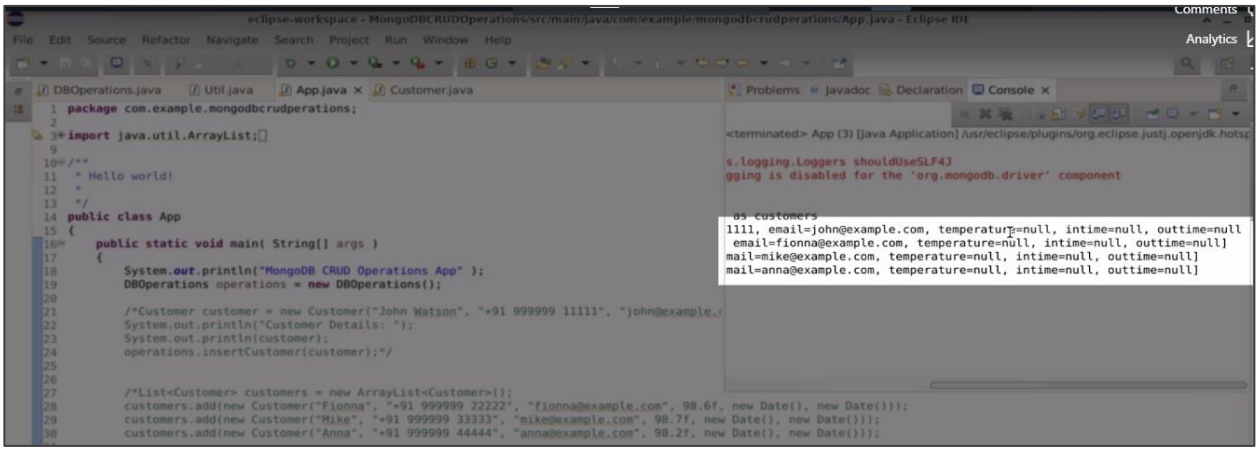


1.8 Go to the **DBOperations.java** file and write the following lines of code (from lines 66 to 70) to can define a document into a **Customer** object:

```
50
51          List<Document> documents = new ArrayList<Document>();
52
53          for(Customer customer : customers) {
54              documents.add(customer.getDocument());
55          }
56
57          collection.insertMany(documents);
58          System.out.println("[DBOperations] "+customers.size()+" Customers inserted into "+Util.COLLECTION_NAME);
59      } catch (Exception e) {
60          System.out.println("Something went Wrong: "+e);
61      }
62  }
63
64  public Customer convertDocumentToCustomer(Document doc) {
65      Customer customer = new Customer();
66
67          customer.setName(doc.get("name").toString());
68          customer.setPhone(doc.get("phone").toString());
69          customer.setEmail(doc.get("email").toString());
70
71      return customer;
72  }
73
74  public List<Customer> getAllCustomers(){
75      List<Customer> customers = new ArrayList<Customer>();
76      try {
77
78          List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
79          for(Document doc : documents) {
80              customers.add(convertDocumentToCustomer(doc));
81          }
82
83      } catch (Exception e) {
84          System.out.println("Something went Wrong: "+e);
85      }
86      return customers;
```

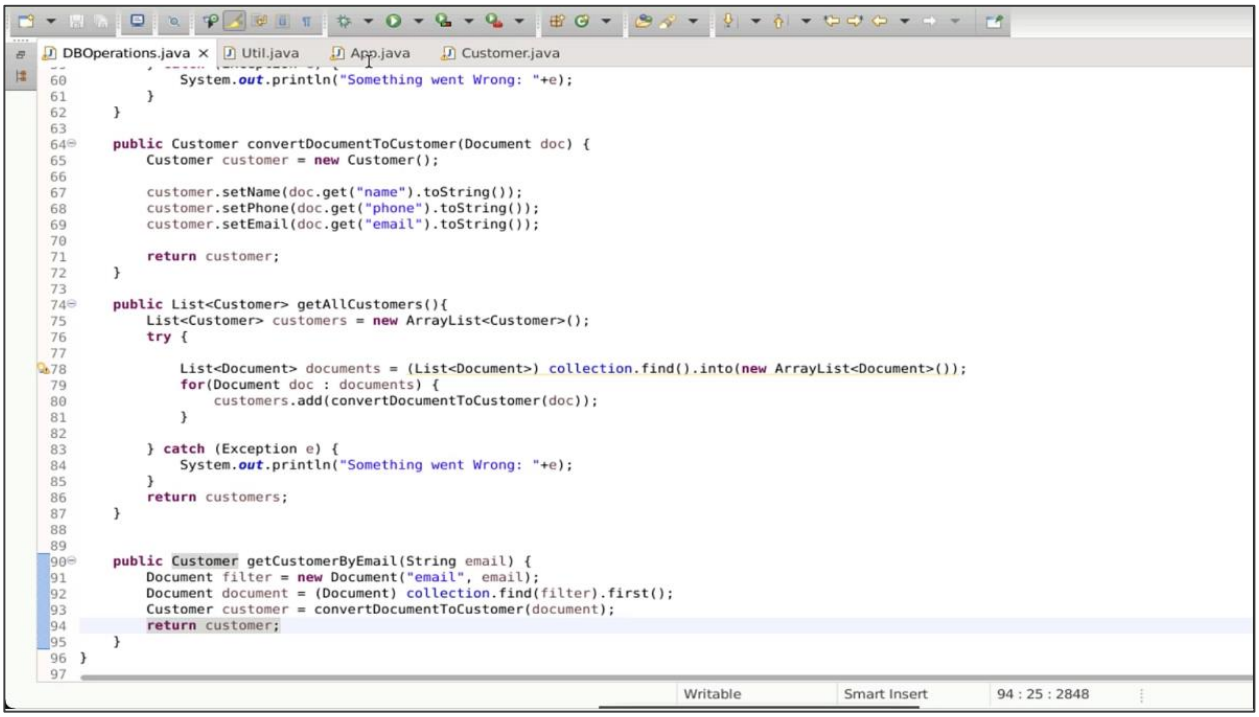
1.9 Now, go to the **App.java** file, call the **getAllCustomers()** method (lines 34 to 36), and run the code. You can see the list of customers, and the other details such as temperature and in-and-out time as **null.**





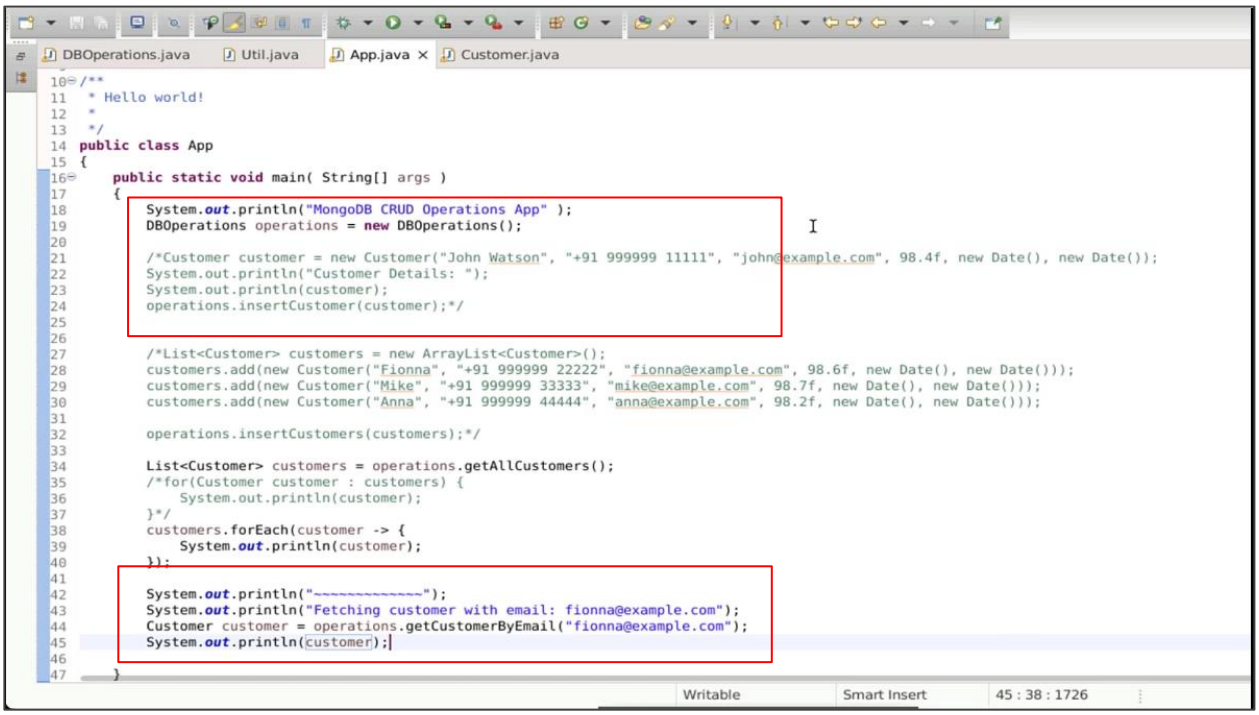
Step 2: Fetch the document

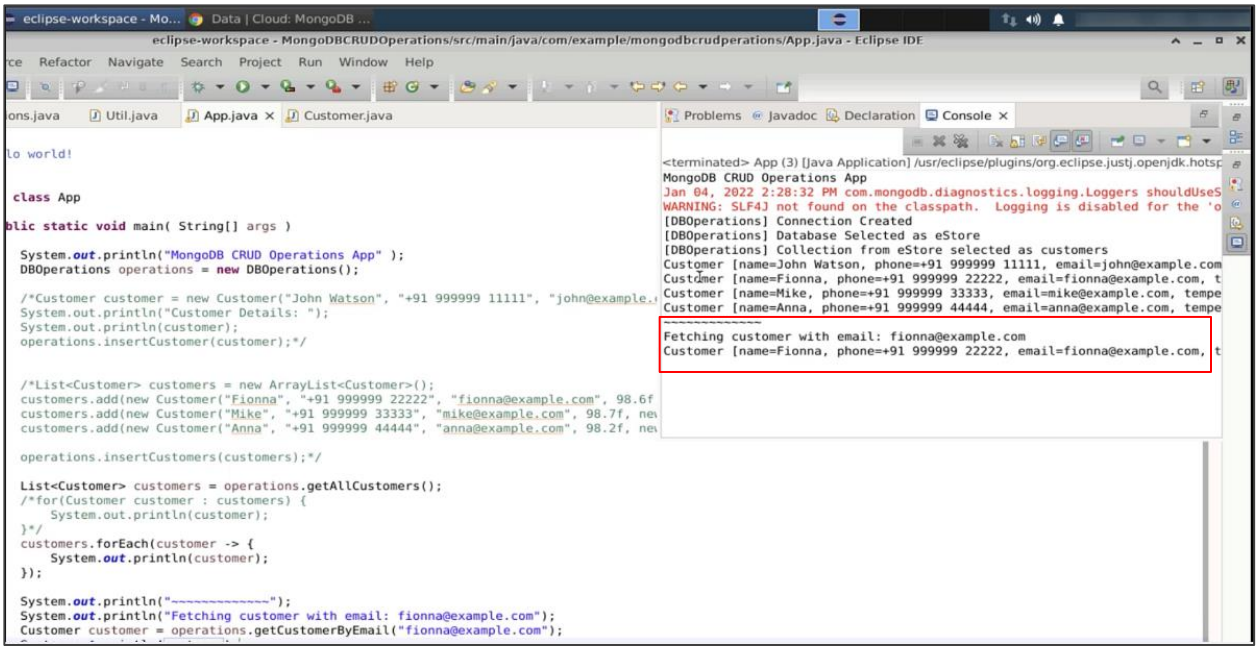
2.1 To fetch a document based on email, go to the **DBOperations.java** file and create a method **getCustomerByEmail** (from lines 90 to 94):



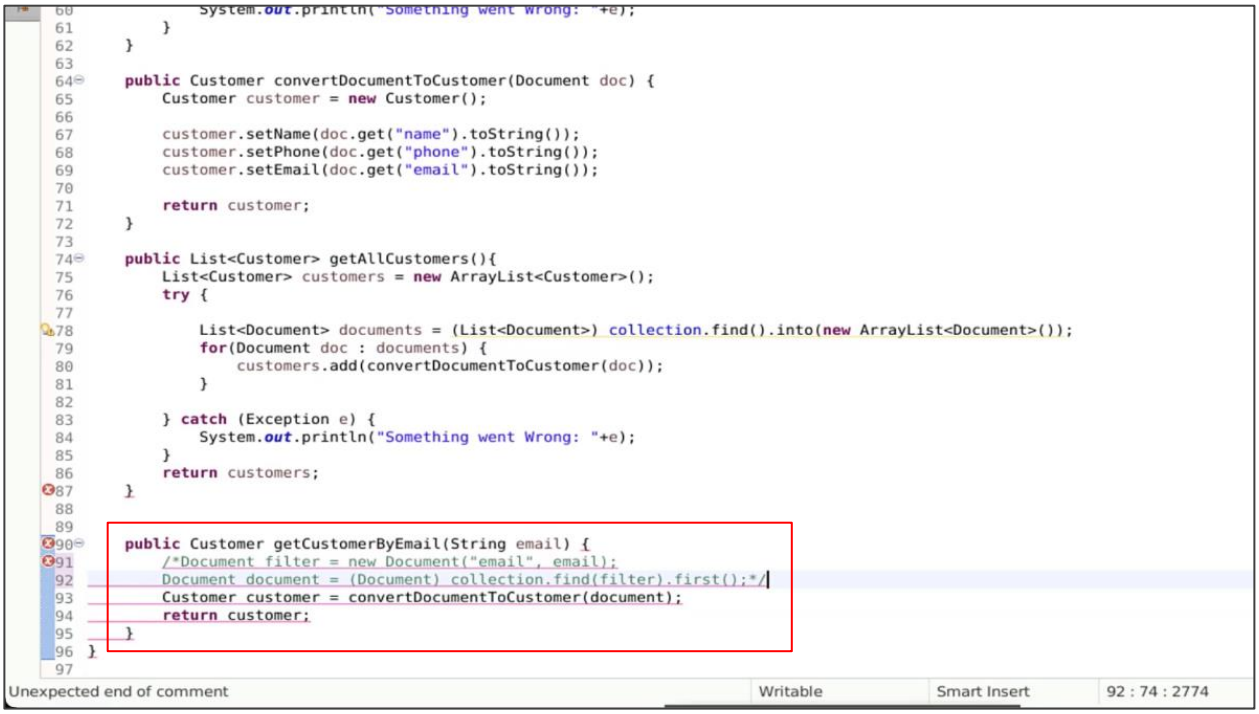

```
60              System.out.println("Something went Wrong: "+e);
61          }
62      }
63
64      public Customer convertDocumentToCustomer(Document doc) {
65          Customer customer = new Customer();
66
67          customer.setName(doc.get("name").toString());
68          customer.setPhone(doc.get("phone").toString());
69          customer.setEmail(doc.get("email").toString());
70
71          return customer;
72      }
73
74      public List<Customer> getAllCustomers(){
75          List<Customer> customers = new ArrayList<Customer>();
76          try {
77
78              List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
79              for(Document doc : documents) {
80                  customers.add(convertDocumentToCustomer(doc));
81              }
82
83          } catch (Exception e) {
84              System.out.println("Something went Wrong: "+e);
85          }
86          return customers;
87      }
88
89
90      public Customer getCustomerByEmail(String email) {
91          Document filter = new Document("email", email);
92          Document document = (Document) collection.find(filter).first();
93          Customer customer = convertDocumentToCustomer(document);
94          return customer;
95      }
96  }
97
```

2.2 Go to the **App.java** file, add a print statement to display the customer email (from lines 42 to 45), and run the code. The following output can be seen when the customer details are fetched using the email **fionna@example.com**:

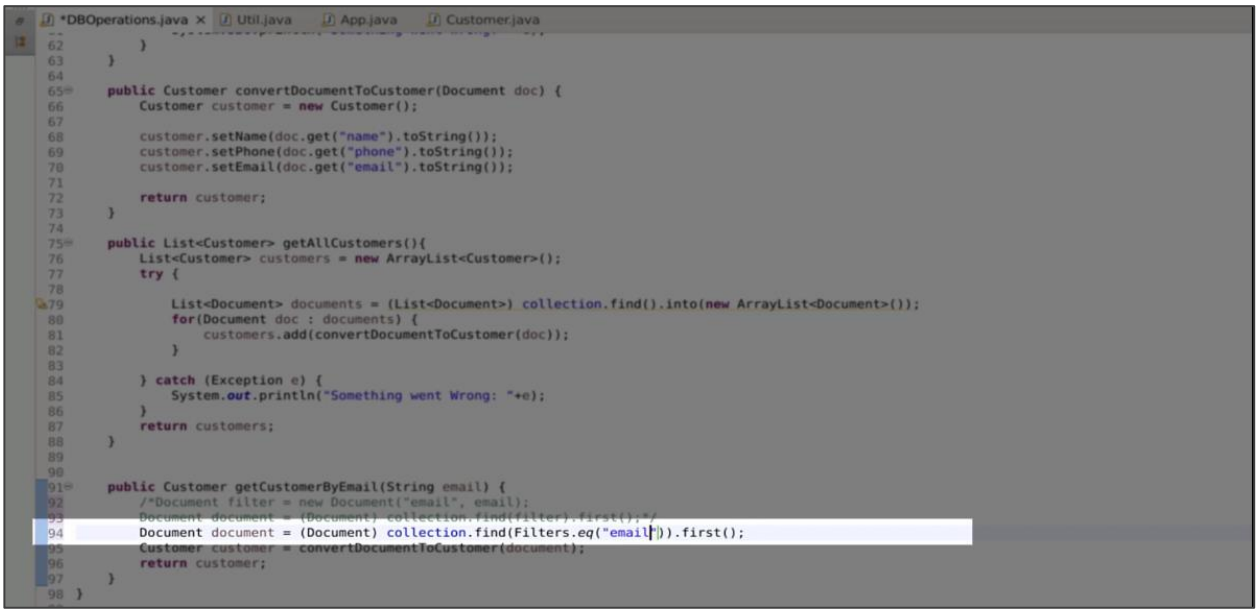


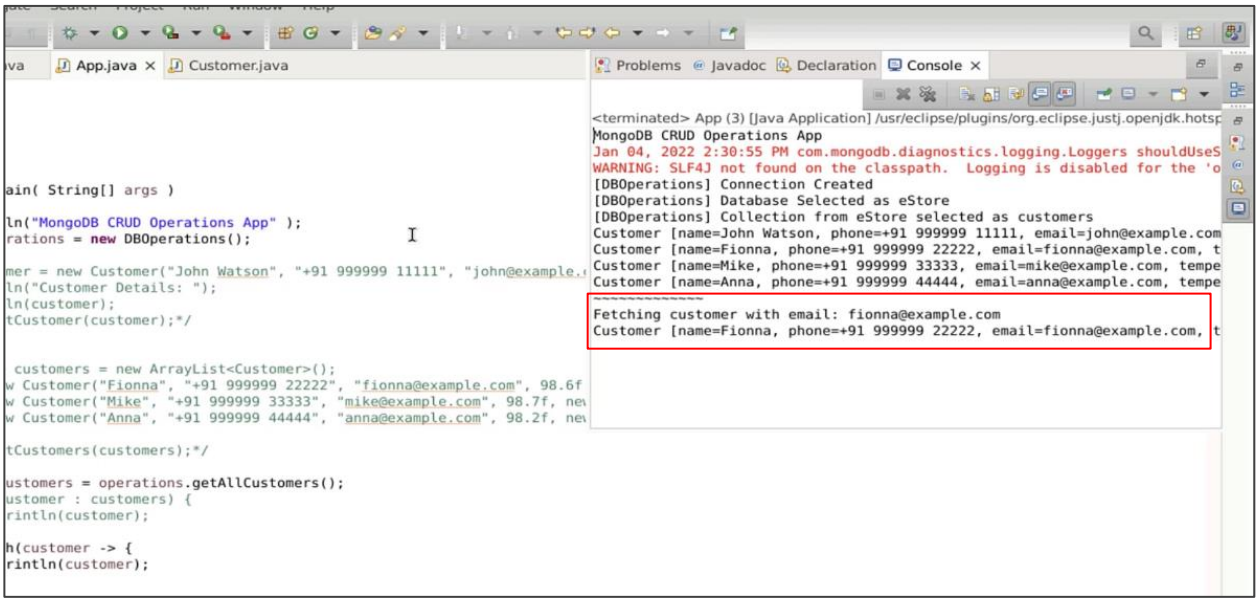


2.3 Go to the **DBOperations.java** file, and mark the following lines of code as comments (lines 91, 92):



2.4 Write the code (line 94) to pass the equal function as an input to the filters, go to the **App.java** file, and run the code. The output shown will be similar to the previous output, but it is a different way of executing them.

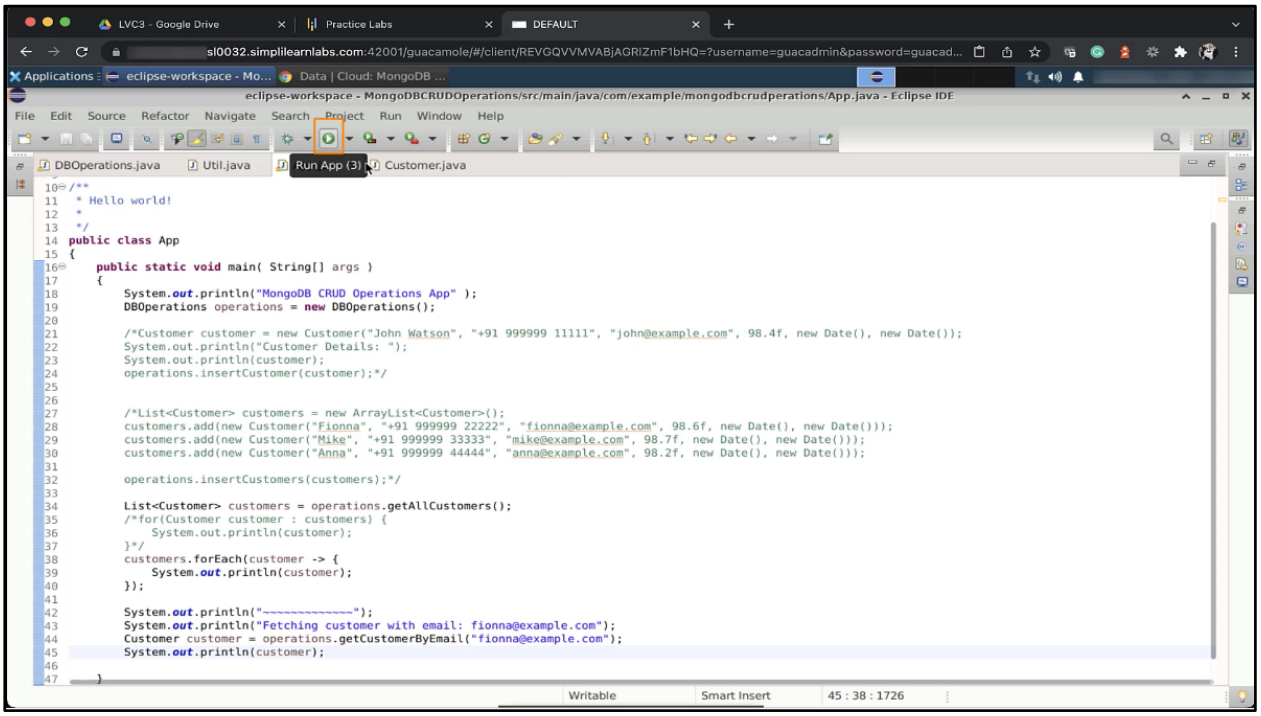




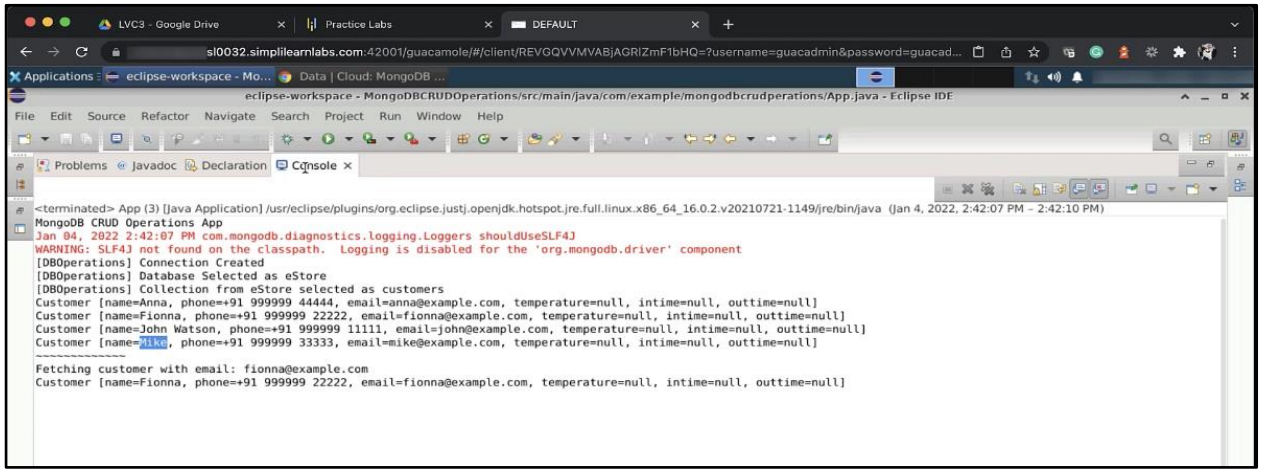
2.5 Go to the **DBOperations.java** file and write the following lines of code (lines 81 & 82) to sort the documents in ascending order:

```
66      public Customer convertDocumentToCustomer(Document doc) {
67          Customer customer = new Customer();
68
69          customer.setName(doc.get("name").toString());
70          customer.setPhone(doc.get("phone").toString());
71          customer.setEmail(doc.get("email").toString());
72
73          return customer;
74      }
75
76      public List<Customer> getAllCustomers(){
77          List<Customer> customers = new ArrayList<Customer>();
78          try {
79
80              List<Document> documents = (List<Document>) collection
81                      .find()
82                      .sort(Sorts.ascending("name"))
83                      .into(new ArrayList<Document>());
84              for(Document doc : documents) {
85                  customers.add(convertDocumentToCustomer(doc));
86              }
87
88          } catch (Exception e) {
89              System.out.println("Something went Wrong: "+e);
90          }
91          return customers;
92      }
93
94
95      public Customer getCustomerByEmail(String email) {
96          /*Document filter = new Document("email", email);
97          Document document = (Document) collection.find(filter).first();*/
98          Document document = (Document) collection.find(Filters.eq("email", email)).first();
99          Customer customer = convertDocumentToCustomer(document);
100         return customer;
```

2.6 Go to the **App.java** file and run the code



The following output shows the data sorted in ascending order of name:



To explore more, similar operations can be performed with different data and conditions. This is how we can read the documents in MongoDB by fetching details based on email.

By following these steps, you have successfully read documents from MongoDB by executing the application and fetching a document based on the e-mail.