

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

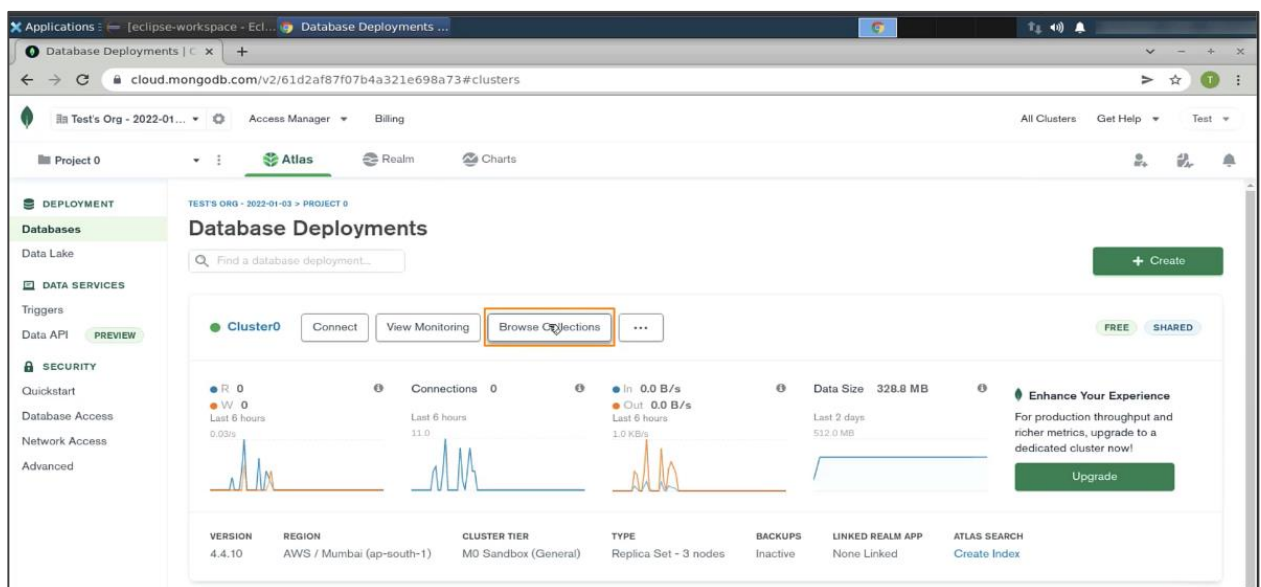
Prerequisites: None

Steps to be followed:

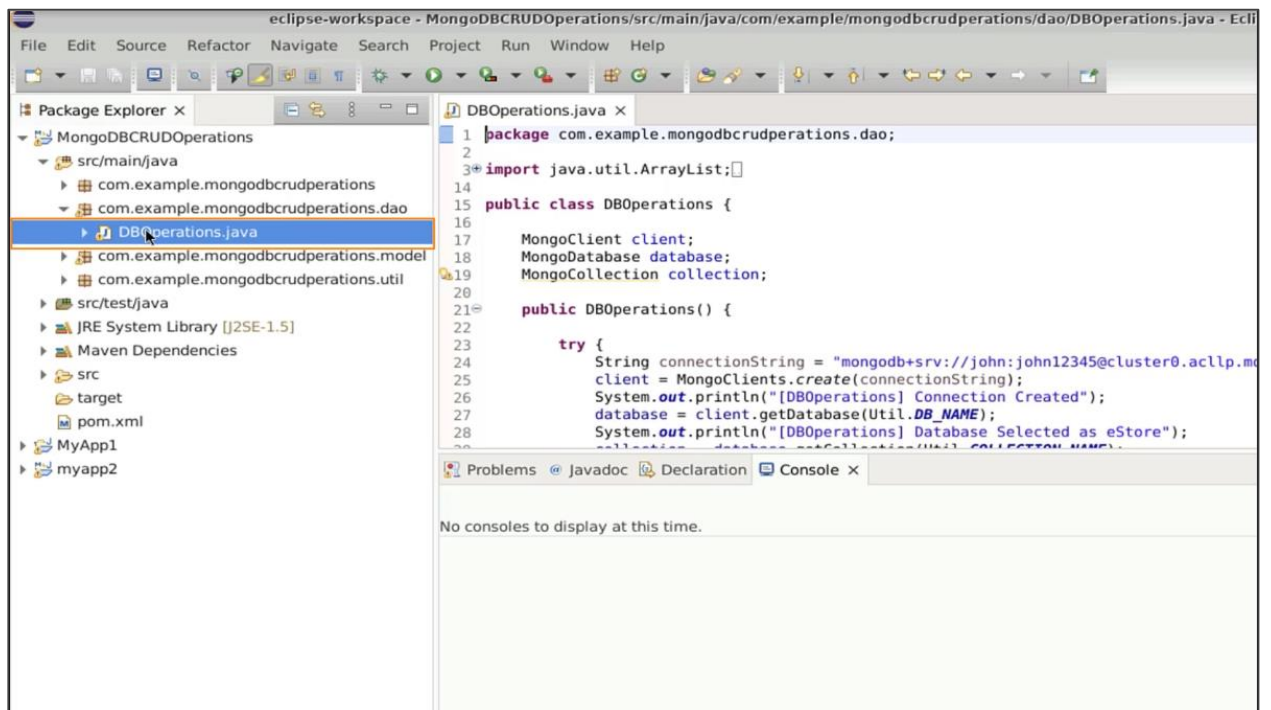
1. Execute the App.java file
2. Fetch the document

Step 1: Execute the App.java file

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

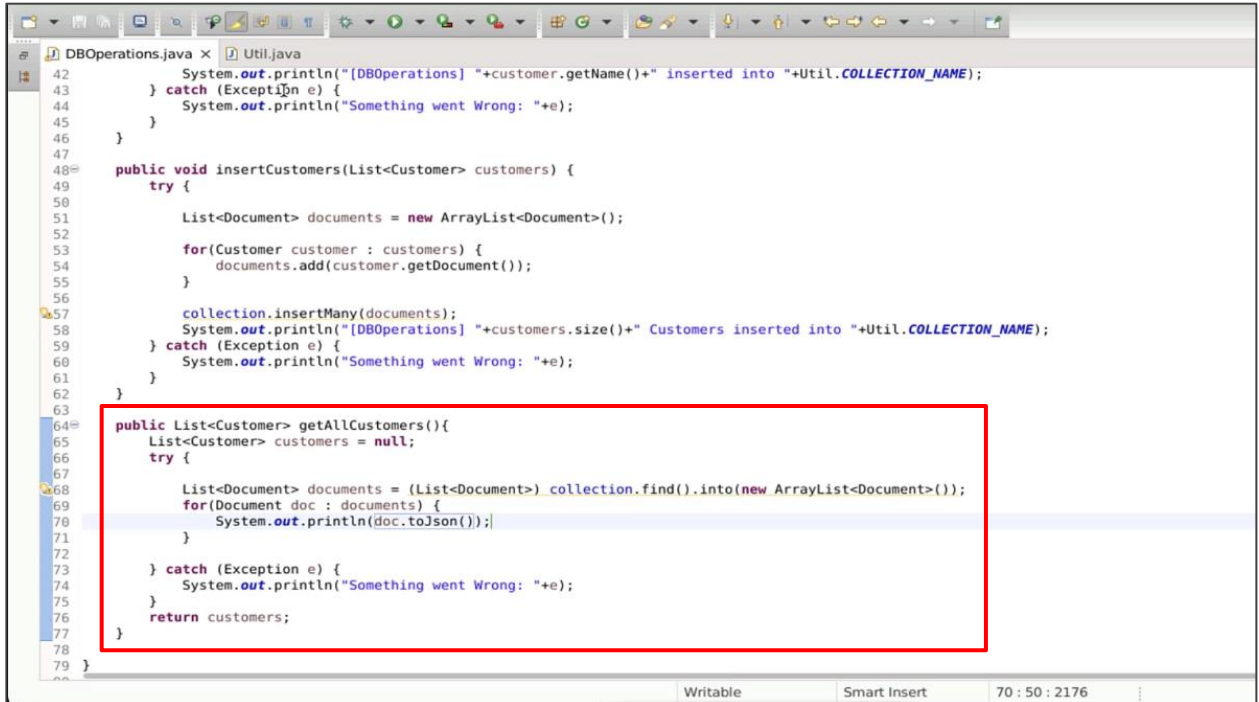


1.2 Under the **MongoDBCRUDOperations** project, open the **DBOperations.java** file



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:

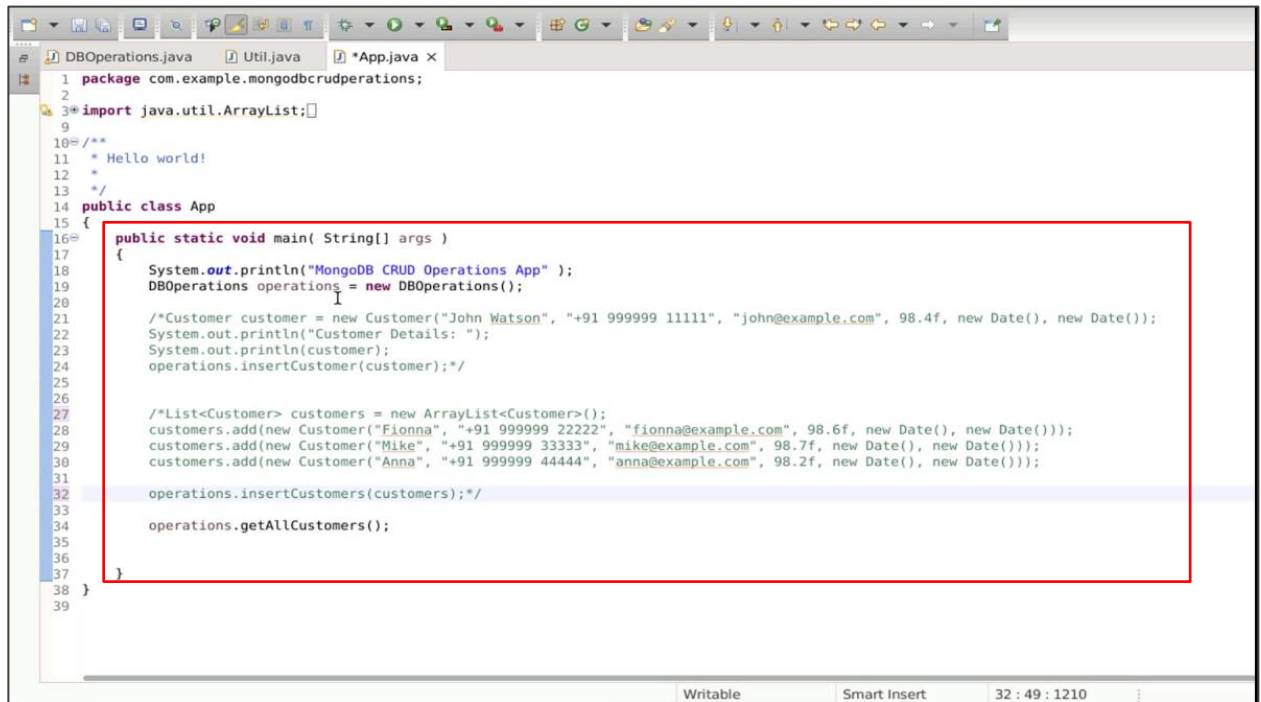


```

42      System.out.println("[DBOperations] "+customer.getName()+" inserted into "+Util.COLLECTION_NAME);
43  } catch (Exception e) {
44      System.out.println("Something went Wrong: "+e);
45  }
46  }
47
48  public void insertCustomers(List<Customer> customers) {
49      try {
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 List<Customer> getAllCustomers(){
65      List<Customer> customers = null;
66      try {
67
68          List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
69          for(Document doc : documents) {
70              System.out.println(doc.toJson());
71          }
72
73      } catch (Exception e) {
74          System.out.println("Something went Wrong: "+e);
75      }
76      return customers;
77  }
78
79  }

```

- 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 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10 /**
11  * Hello world!
12  *
13  */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         System.out.println("MongoDB CRUD Operations App" );
19         DBOperations operations = new DBOperations();
20
21         /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
22         System.out.println("Customer Details: ");
23         System.out.println(customer);
24         operations.insertCustomer(customer);*/
25
26         /*List<Customer> customers = new ArrayList<Customer>();
27         customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
28         customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
29         customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
30
31         operations.insertCustomers(customers);*/
32
33         operations.getAllCustomers();
34
35
36     }
37 }
```

Writable Smart Insert 32 : 49 : 1210

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

```

1 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5 /**
6  * Hello world!
7  */
8
9 public class App
10 {
11     public static void main( String[] args )
12     {
13         System.out.println("MongoDB CRUD Operations App" );
14         DBOperations operations = new DBOperations();
15
16         /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
17         System.out.println("Customer Details: ");
18         System.out.println(customer);
19         operations.insertCustomer(customer);*/
20
21         /*List<Customer> customers = new ArrayList<Customer>();
22         customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
23         customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
24         customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
25
26         operations.insertCustomers(customers);*/
27
28         operations.getAllCustomers();
29     }
30 }
31

```

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux64/java:
mongodb.diagnostics.Logging.Loggers shouldUseSLF4J
he classpath. Logging is disabled for the 'org.mongodb.driver' component
ated
ted as eStore
m eStore selected as customers
d47560840122e", "name": "John Watson", "phone": "+91 999999 11111", "ema
5da6c1fa283e9", "name": "Fionna", "phone": "+91 999999 22222", "email":
5da6c1fa283ea", "name": "Mike", "phone": "+91 999999 33333", "email": "m
5da6c1fa283eb", "name": "Anna", "phone": "+91 999999 44444", "email": "a

```

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:

```

42      System.out.println("[DBOperations] "+customer.getName()+" inserted into "+Util.COLLECTION_NAME);
43    } catch (Exception e) {
44      System.out.println("Something went Wrong: "+e);
45    }
46  }
47
48  public void insertCustomers(List<Customer> customers) {
49    try {
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 List<Customer> getAllCustomers(){
65    List<Customer> customers = null;
66    try {
67
68      List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
69      for(Document doc : documents) {
70        //System.out.println(doc.toJson());
71        System.out.println(doc.get("name"));
72        System.out.println(doc.get("intime"));
73      }
74    } catch (Exception e) {
75      System.out.println("Something went Wrong: "+e);
76    }
77  }
78  return customers;
79

```

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

```

App.java
mongodbcrudoperations;
ArrayList<

>

void main( String[] args )
{
    println("MongoDB CRUD Operations App" );
    $ operations = new DBOperations();

    customer = new Customer("John Watson", "+91 999999 1111", "john@example.com");
    println("Customer Details: ");
    println(customer);
    insertCustomer(customer);/*

> customers = new ArrayList<Customer>();
add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date(), new Date()));
add(new Customer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date()));
add(new Customer("Anna", "+91 999999 4444", "anna@example.com", 98.2f, new Date(), new Date()));
}
}

```

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot
MongoDB CRUD Operations App
Jan 04, 2022 2:12:25 PM com.mongodb.diagnostics.logging.Loggers shouldUseS
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
John Watson
Tue Jan 04 10:01:41 UTC 2022
Fionna
Tue Jan 04 10:08:24 UTC 2022
Mike
Tue Jan 04 10:08:24 UTC 2022
Anna
Tue Jan 04 10:08:24 UTC 2022

```


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      List<Document> documents = new ArrayList<Document>();
51
52      for(Customer customer : customers) {
53          documents.add(customer.getDocument());
54      }
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      customer.setName(doc.get("name").toString());
67      customer.setPhone(doc.get("phone").toString());
68      customer.setEmail(doc.get("email").toString());
69  }
70  }
71
72  return customer;
73  }
74
75  public List<Customer> getAllCustomers(){
76      List<Customer> customers = new ArrayList<Customer>();
77      try {
78
79          List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
80          for(Document doc : documents) {
81              customers.add(convertDocumentToCustomer(doc));
82          }
83      } catch (Exception e) {
84          System.out.println("Something went Wrong: "+e);
85      }
86      return customers;
87  }

```

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**.

The screenshot shows the Eclipse IDE with the **App.java** file open. The code in **App.java** is as follows:

```

1 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5
6 /**
7  * Hello world!
8  */
9
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println("MongoDB CRUD Operations App");
15         DBOperations operations = new DBOperations();
16
17         Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com");
18         System.out.println("Customer Details: ");
19         System.out.println(customer);
20         operations.insertCustomer(customer);
21     }
22 }

```

The console output shows the following messages:

```

<terminated> App [3] [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot...
MongoDB CRUD Operations App
Jan 04, 2022 2:16:29 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com, t
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, t
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, t
Customer [name=Anna, phone=+91 999999 44444, email=anna@example.com, t

```

The screenshot shows the Eclipse IDE with the `App.java` file open. The code defines a `main` method that creates a `Customer` object and inserts it into a MongoDB database. The console output shows the execution of the application, including the message "MongoDB CRUD Operations App" and the details of the inserted customer.

```

1 package com.example.mongodbcrudoperations;
2
3 import java.util.ArrayList;
4
5
6 /**
7  * Hello world!
8  */
9
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println("MongoDB CRUD Operations App" );
15         DBOperations operations = new DBOperations();
16
17         /*Customer customer = new Customer("John Watson", "+91 999999 1111", "john@example.com", "98.6f", new Date(), new Date());
18         System.out.println("Customer Details: ");
19         System.out.println(customer);
20         operations.insertCustomer(customer);*/
21
22         /*List<Customer> customers = new ArrayList<Customer>();
23         customers.add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", "98.6f", new Date(), new Date()));
24         customers.add(new Customer("Mike", "+91 999999 3333", "mike@example.com", "98.7f", new Date(), new Date()));
25         customers.add(new Customer("Anna", "+91 999999 4444", "anna@example.com", "98.2f", new Date(), new Date()));
26     }
27 }

```

Console Output:

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
s.logging.Loggers shouldUseSLF4J
gging is disabled for the 'org.mongodb.driver' component

-- customers
1111, email=john@example.com, temperature=null, intime=null, outtime=null
email=fionna@example.com, temperature=null, intime=null, outtime=null
mail=mike@example.com, temperature=null, intime=null, outtime=null
mail=anna@example.com, temperature=null, intime=null, outtime=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):

The screenshot shows the Eclipse IDE with the `DBOperations.java` file open. The code defines a `getCustomerByEmail` method that fetches a customer document from a MongoDB database based on the email address.

```

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         List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
78         for(Document doc : documents) {
79             customers.add(convertDocumentToCustomer(doc));
80         }
81     } catch (Exception e) {
82         System.out.println("Something went Wrong: "+e);
83     }
84     return customers;
85 }
86
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**:

```

10 /**
11  * Hello world!
12  *
13  */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         System.out.println("MongoDB CRUD Operations App" );
19         DBOperations operations = new DBOperations();
20
21         /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
22         System.out.println("Customer Details: ");
23         System.out.println(customer);
24         operations.insertCustomer(customer);*/
25
26
27         /*List<Customer> customers = new ArrayList<Customer>();
28         customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
29         customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
30         customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
31
32         operations.insertCustomers(customers);*/
33
34         List<Customer> customers = operations.getAllCustomers();
35         /*for(Customer customer : customers) {
36             System.out.println(customer);
37         }*/
38         customers.forEach(customer -> {
39             System.out.println(customer);
40         });
41
42         System.out.println("~~~~~");
43         System.out.println("Fetching customer with email: fionna@example.com");
44         Customer customer = operations.getCustomerByEmail("fionna@example.com");
45         System.out.println(customer);
46     }
47 }
  
```

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot
MongoDB CRUD Operations App
Jan 04, 2022 2:28:32 PM com.mongodb.diagnostics.logging.Loggers shouldUseS
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com, t
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, t
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, tempe
Customer [name=Anna, phone=+91 999999 44444, email=anna@example.com, tempe

Fetching customer with email: fionna@example.com
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, t
  
```

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

```

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    } catch (Exception e) {
83      System.out.println("Something went Wrong: "+e);
84    }
85    return customers;
86  }
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

```

Unexpected end of comment

Writable Smart Insert 92 : 74 : 2774

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.

```

# DBOperations.java x Util.java App.java Customer.java
62    }
63  }
64
65  public Customer convertDocumentToCustomer(Document doc) {
66    Customer customer = new Customer();
67
68    customer.setName(doc.get("name").toString());
69    customer.setPhone(doc.get("phone").toString());
70    customer.setEmail(doc.get("email").toString());
71
72    return customer;
73  }
74
75  public List<Customer> getAllCustomers(){
76    List<Customer> customers = new ArrayList<Customer>();
77    try {
78
79      List<Document> documents = (List<Document>) collection.find().into(new ArrayList<Document>());
80      for(Document doc : documents) {
81        customers.add(convertDocumentToCustomer(doc));
82      }
83    } catch (Exception e) {
84      System.out.println("Something went Wrong: "+e);
85    }
86    return customers;
87  }
88
89
90
91  public Customer getCustomerByEmail(String email) {
92    /*Document filter = new Document("email", email);
93    Document document = (Document) collection.find(filter).first();*/
94    Document document = (Document) collection.find(Filters.eq("email", email)).first();
95    Customer customer = convertDocumentToCustomer(document);
96    return customer;
97  }
98 }

```

```

App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot...
MongoDB CRUD Operations App
Jan 04, 2022 2:30:55 PM com.mongodb.diagnostics.logging.Loggers shouldUseS...
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'o...
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com, t...
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, t...
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, tempe...
Customer [name=Anna, phone=+91 999999 44444, email=anna@example.com, tempe...

Fetching customer with email: fionna@example.com
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, t...
  
```

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:

```

83
84 public Customer convertDocumentToCustomer(Document doc) {
85     Customer customer = new Customer();
86
87     customer.setName(doc.get("name").toString());
88     customer.setPhone(doc.get("phone").toString());
89     customer.setEmail(doc.get("email").toString());
90
91     return customer;
92 }
93
94 public List<Customer> getAllCustomers(){
95     List<Customer> customers = new ArrayList<Customer>();
96     try {
97
98         List<Document> documents = (List<Document>) collection
99             .find()
100             .sort(Sorts.ascending("name"))
101             .into(new ArrayList<Document>());
102         for(Document doc : documents) {
103             customers.add(convertDocumentToCustomer(doc));
104         }
105     } catch (Exception e) {
106         System.out.println("Something went Wrong: "+e);
107     }
108     return customers;
109 }
110
111 public Customer getCustomerByEmail(String email) {
112     /*Document filter = new Document("email", email);
113     Document document = (Document) collection.find(filter).first();*/
114     Document document = (Document) collection.find(Filters.eq("email", email)).first();
115     Customer customer = convertDocumentToCustomer(document);
116     return customer;
  
```

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

```

10= /**
11  * Hello world!
12  */
13  */
14  public class App
15  {
16=     public static void main( String[] args )
17     {
18         System.out.println("MongoDB CRUD Operations App");
19         DBOperations operations = new DBOperations();
20
21         /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
22         System.out.println("Customer Details: ");
23         System.out.println(customer);
24         operations.insertCustomer(customer);*/
25
26
27         /*List<Customer> customers = new ArrayList<Customer>();
28         customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
29         customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
30         customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
31
32         operations.insertCustomers(customers);*/
33
34         List<Customer> customers = operations.getAllCustomers();
35         /*for(Customer customer : customers) {
36             System.out.println(customer);
37         }*/
38         customers.forEach(customer -> {
39             System.out.println(customer);
40         });
41
42         System.out.println("~~~~~");
43         System.out.println("Fetching customer with email: fionna@example.com");
44         Customer customer = operations.getCustomerByEmail("fionna@example.com");
45         System.out.println(customer);
46
47     }

```

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

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.linux.x86_64_16.0.2.v20210721-1149/jre/bin/java (Jan 4, 2022, 2:42:07 PM - 2:42:10 PM)
MongoDB CRUD Operations App
Jan 04, 2022 2:42:07 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath. Logging is disabled for the 'org.mongodb.driver' component
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer {name=Anna, phone=+91 999999 44444, email=anna@example.com, temperature=null, intime=null, outtime=null}
Customer {name=Fionna, phone=+91 999999 22222, email=fionna@example.com, temperature=null, intime=null, outtime=null}
Customer {name=John Watson, phone=+91 999999 11111, email=john@example.com, temperature=null, intime=null, outtime=null}
Customer {name=Mike, phone=+91 999999 33333, email=mike@example.com, temperature=null, intime=null, outtime=null}
~~~~~
Fetching customer with email: fionna@example.com
Customer {name=Fionna, phone=+91 999999 22222, email=fionna@example.com, temperature=null, intime=null, outtime=null}

```

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.