

Lesson 01 Demo 02

Creating a Table in DB and Implementing ORM with OOPs

Objective: To create a customer model, a table in DB, and implement ORM with OOPs for managing customer data in a Java application

Tool required: Eclipse IDE

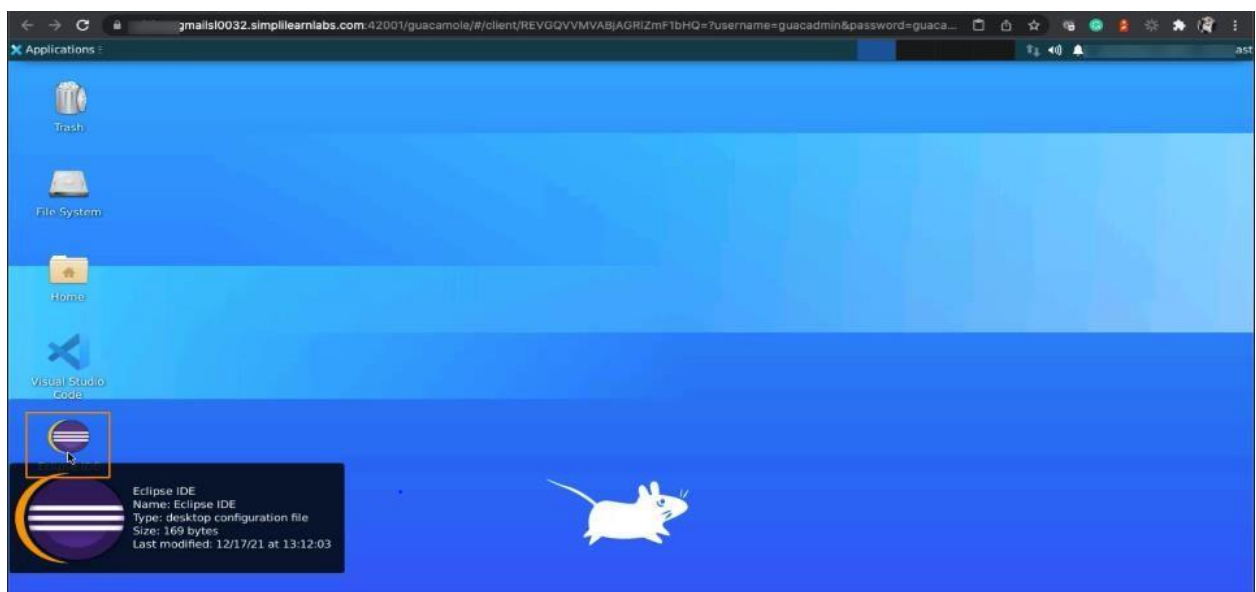
Prerequisites: Lesson 01 Demo 01

Steps to be followed:

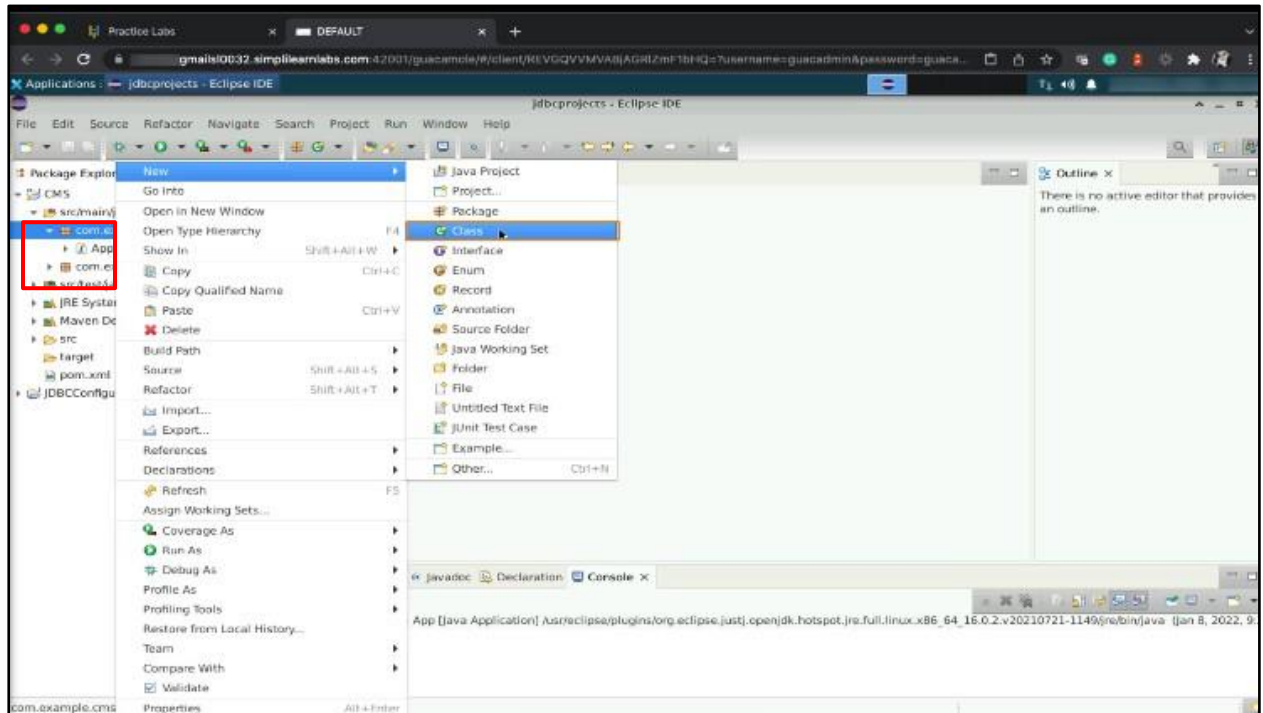
1. Create a new class
2. Write a customer object in App.java
3. Open the terminal window
4. Declare a method for a customer in the DAO interface

Step 1: Create a new class

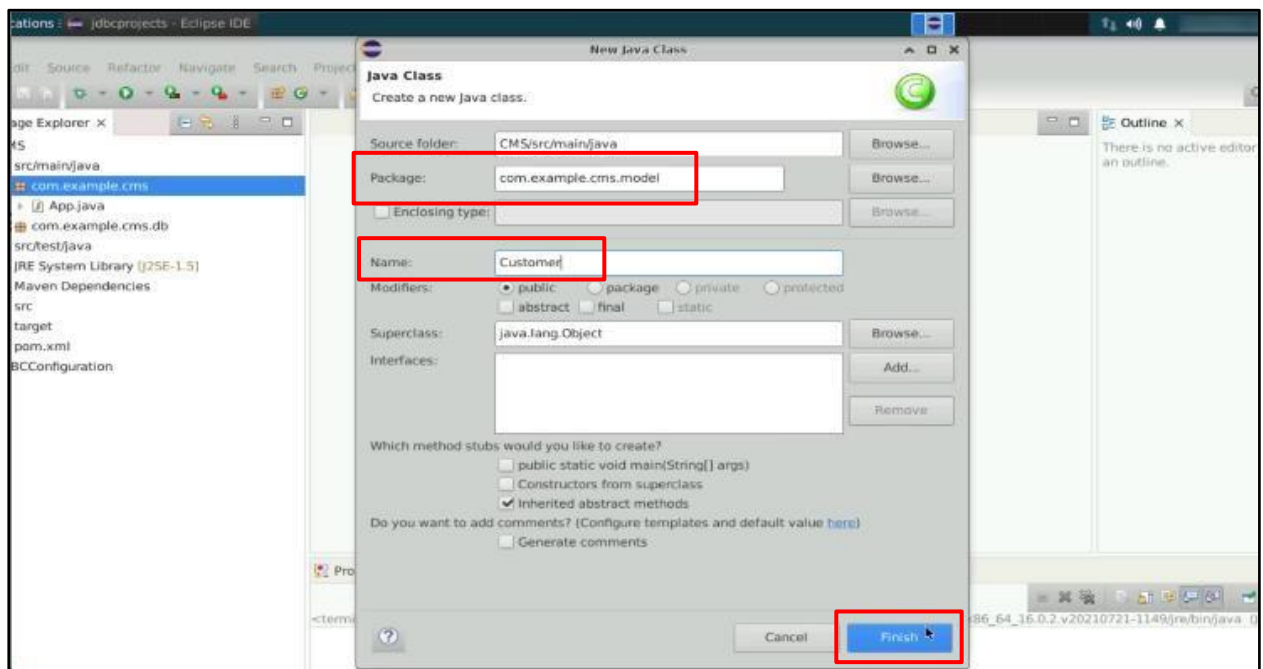
1.1 Open **Eclipse IDE** in your lab as shown in the screenshot below:



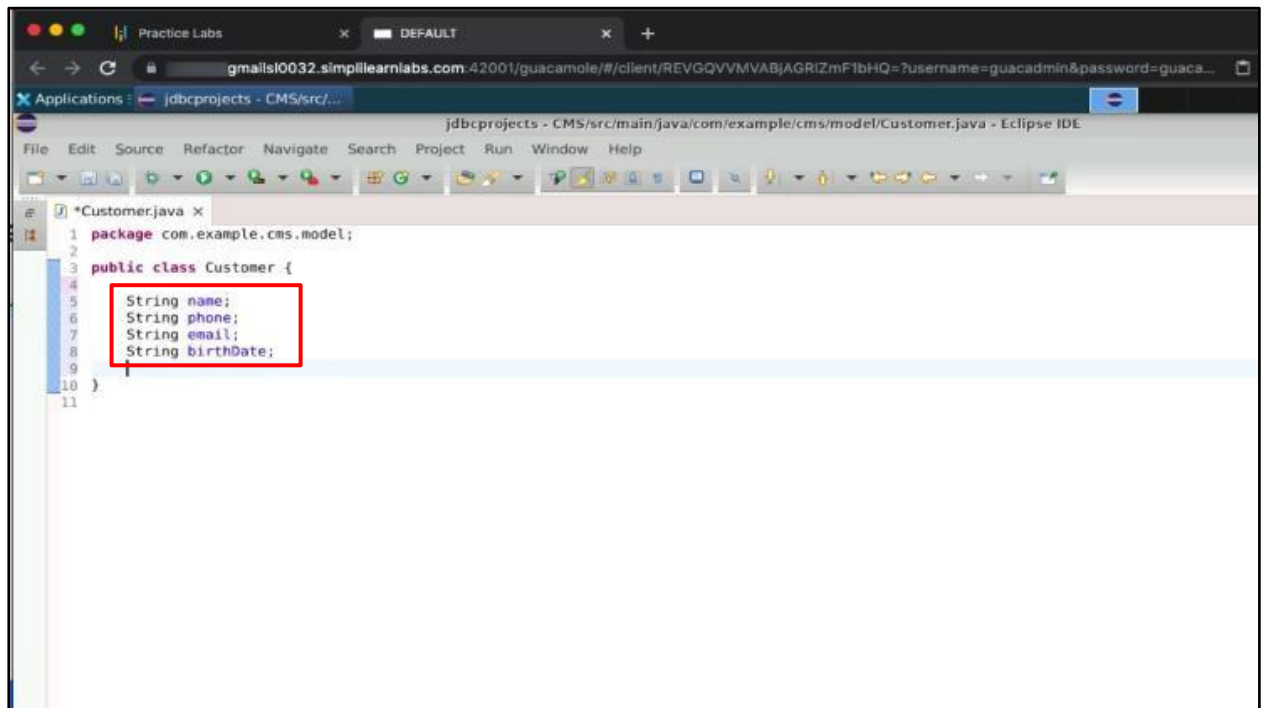
1.2 Right-click on **com.example.cms** in the **src** folder, select **New**, and click on **Class** to create a new class as shown in the screenshot below:



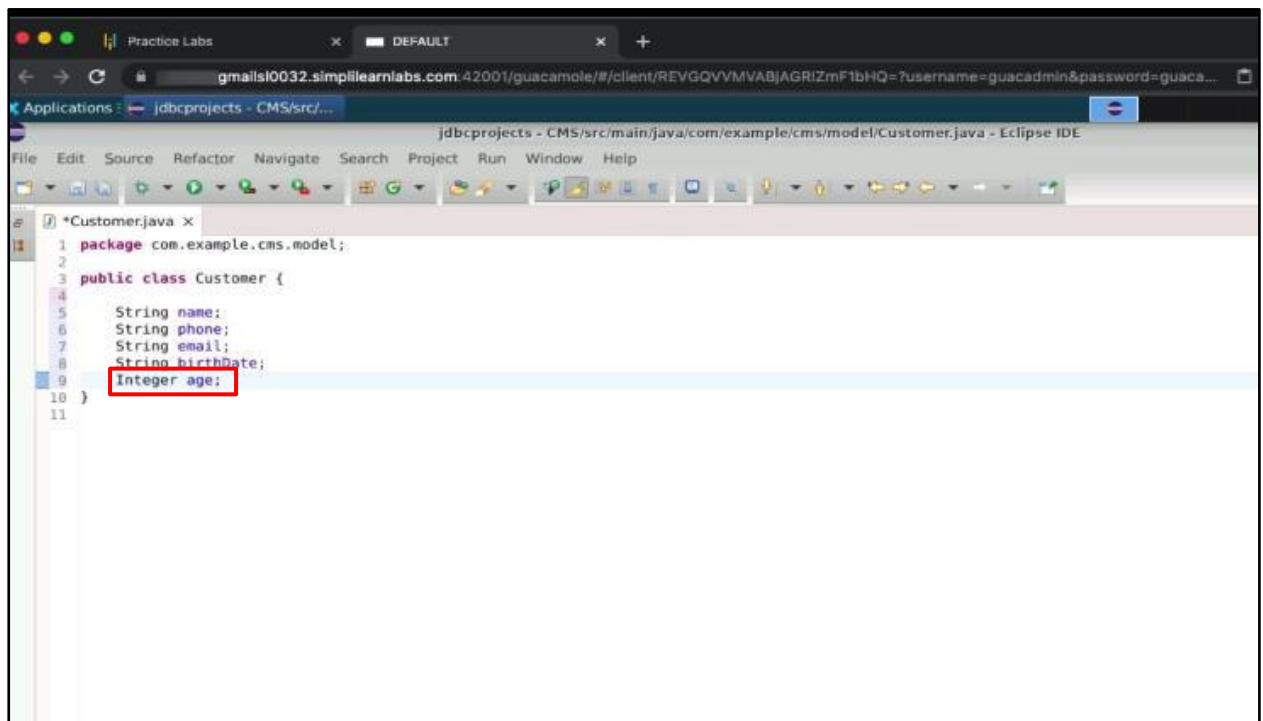
1.3 Add a name for class, package and click on **Finish** as shown in the screenshot below:



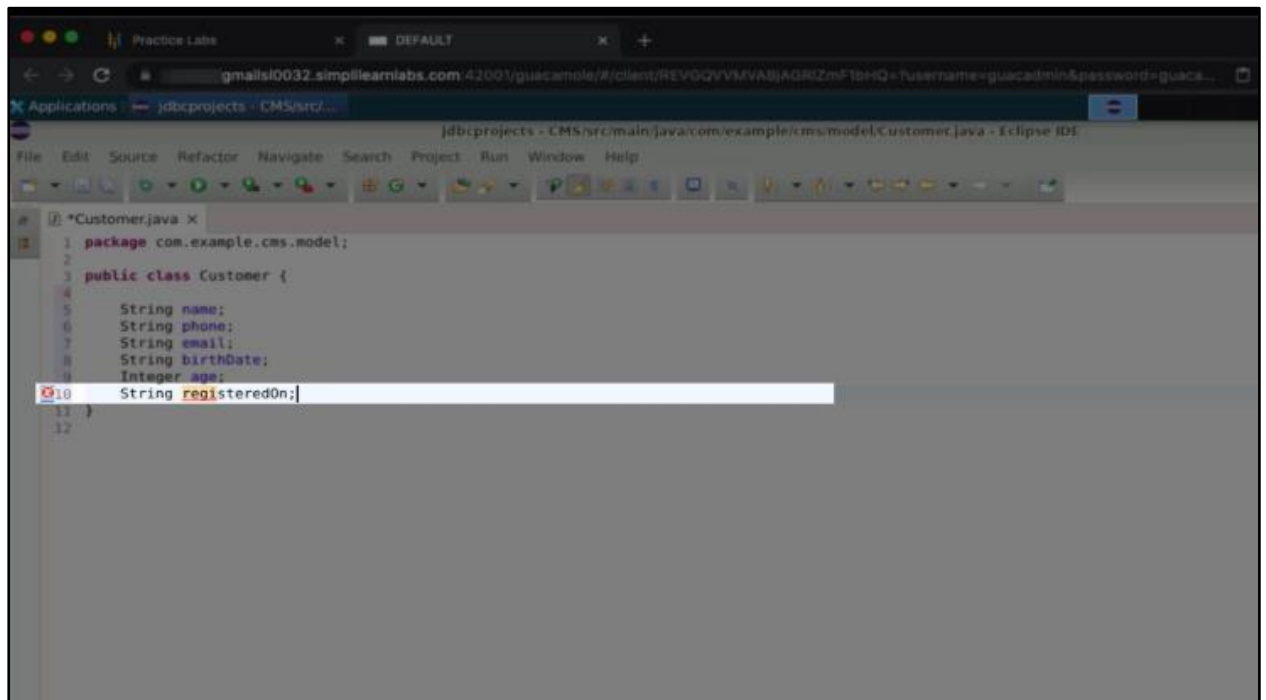
1.4 Add string attributes for the **Customer** database as shown in the below screenshot:



1.5 Add an integer attribute for **age** as shown in the below screenshot:



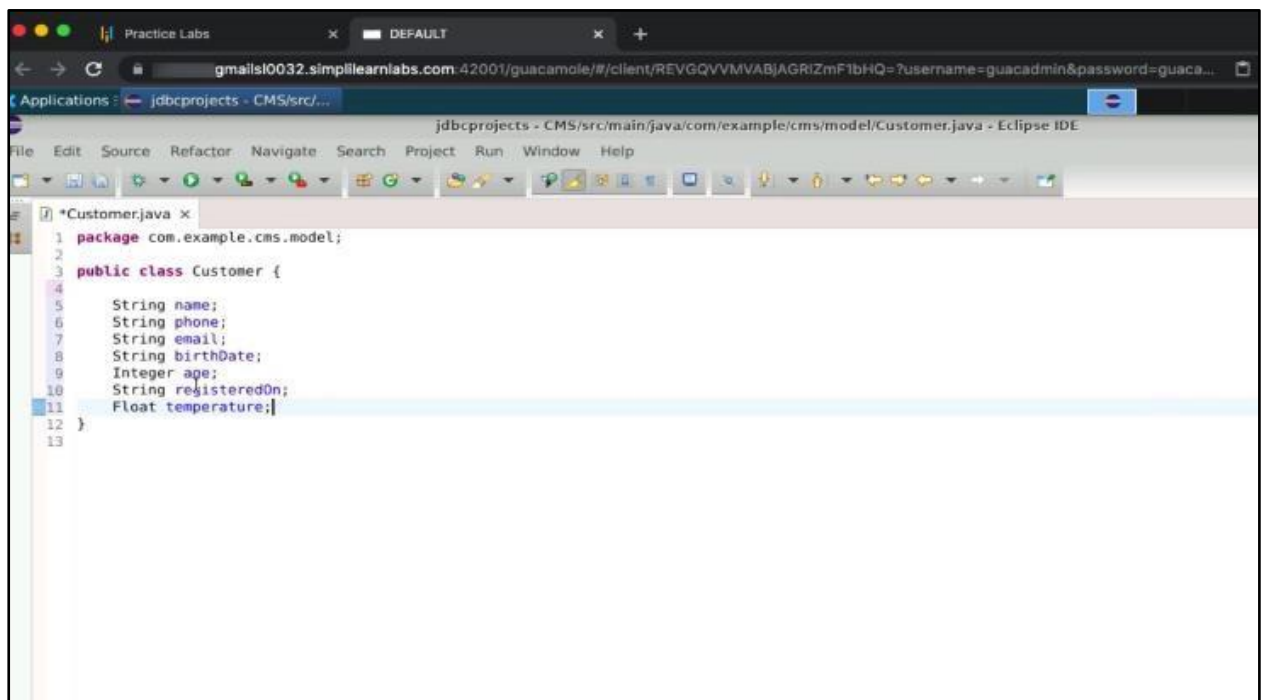
1.6 Add a string attribute for **registeredOn**:



```

1 package com.example.cms.model;
2
3 public class Customer {
4
5     String name;
6     String phone;
7     String email;
8     String birthDate;
9     Integer age;
10    String registeredOn;
11
12 }
  
```

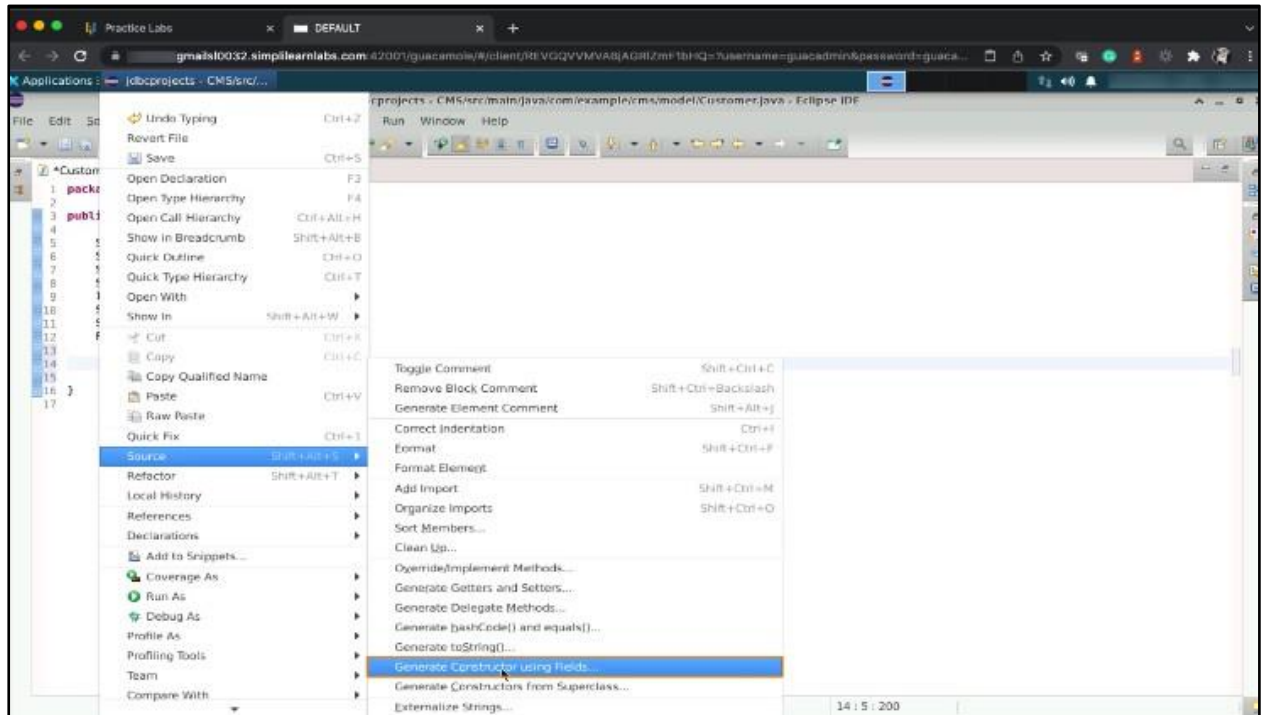
1.7 Add a floating-point attribute for **temperature**



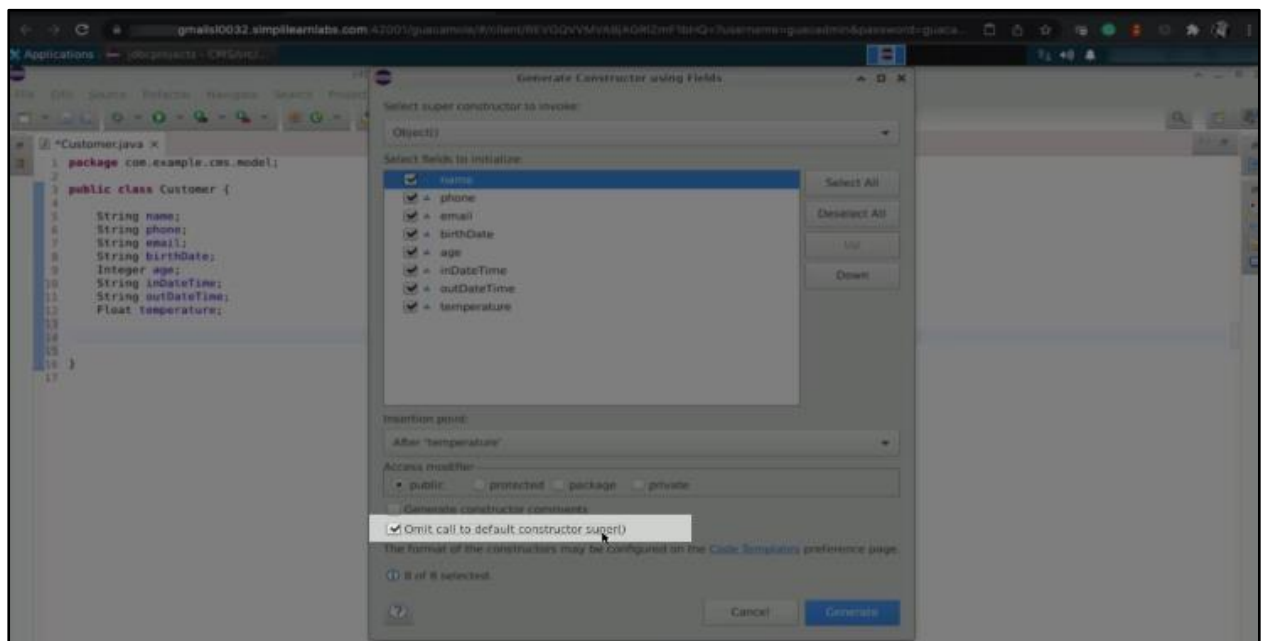
```

1 package com.example.cms.model;
2
3 public class Customer {
4
5     String name;
6     String phone;
7     String email;
8     String birthDate;
9     Integer age;
10    String registeredOn;
11    Float temperature;
12 }
13
  
```

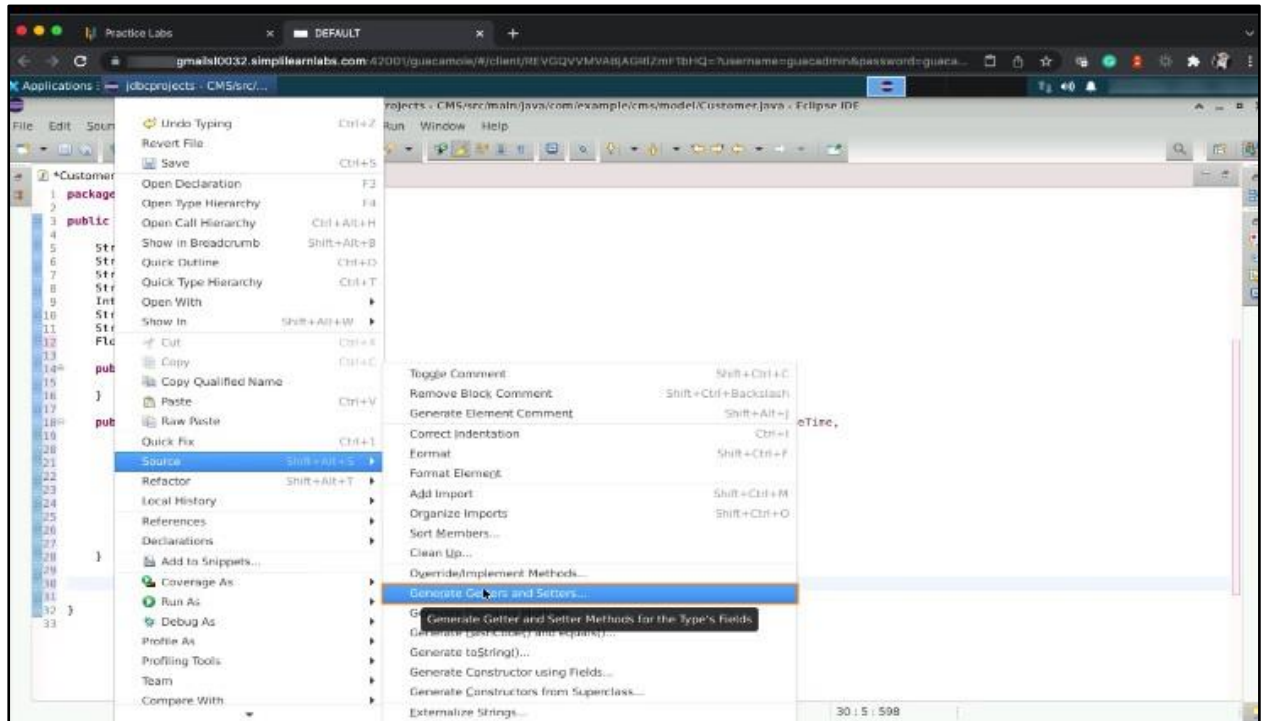
1.8 Right-click on class, select **Source**, and click on **Generate Constructor using Fields** to create a constructor



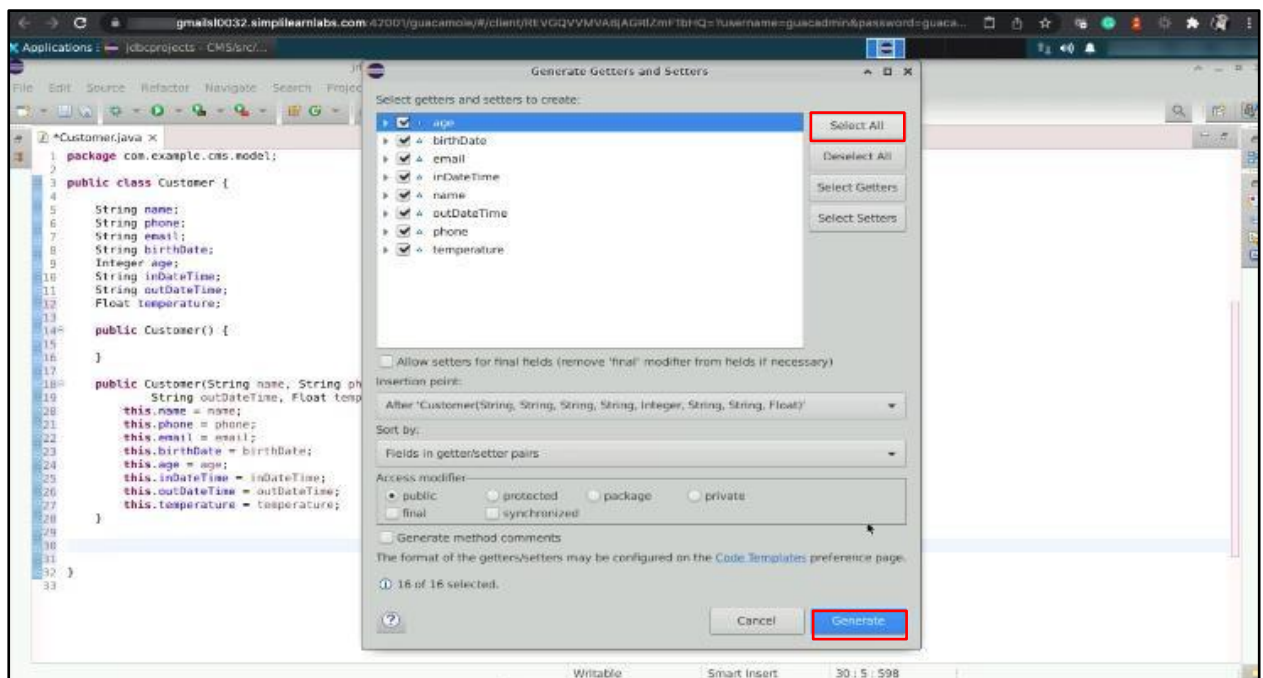
1.9 Mark the checkbox **Omit call to default constructor super()** and click on **Generate**. The constructor will be created.



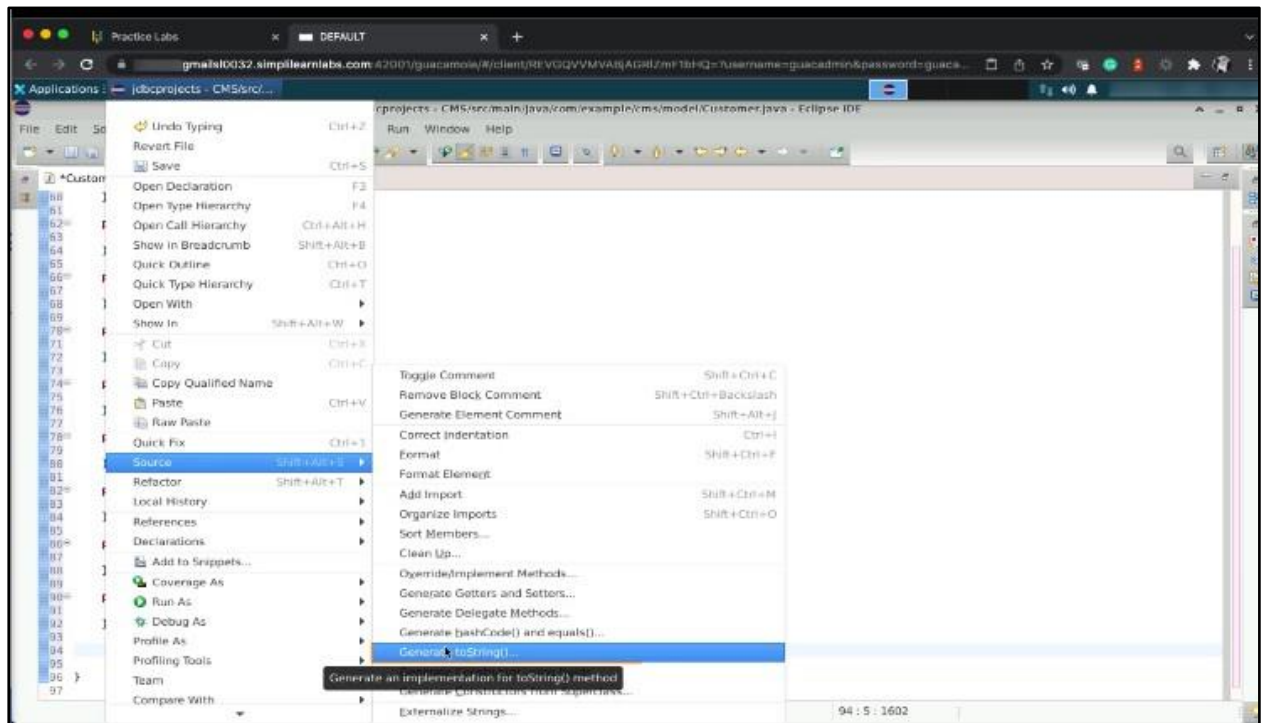
1.10 Right-click on class, select **Source**, and click on **Generate Getters and Setters** to write individual attributes



1.11 Click on **Select All** and then on **Generate**

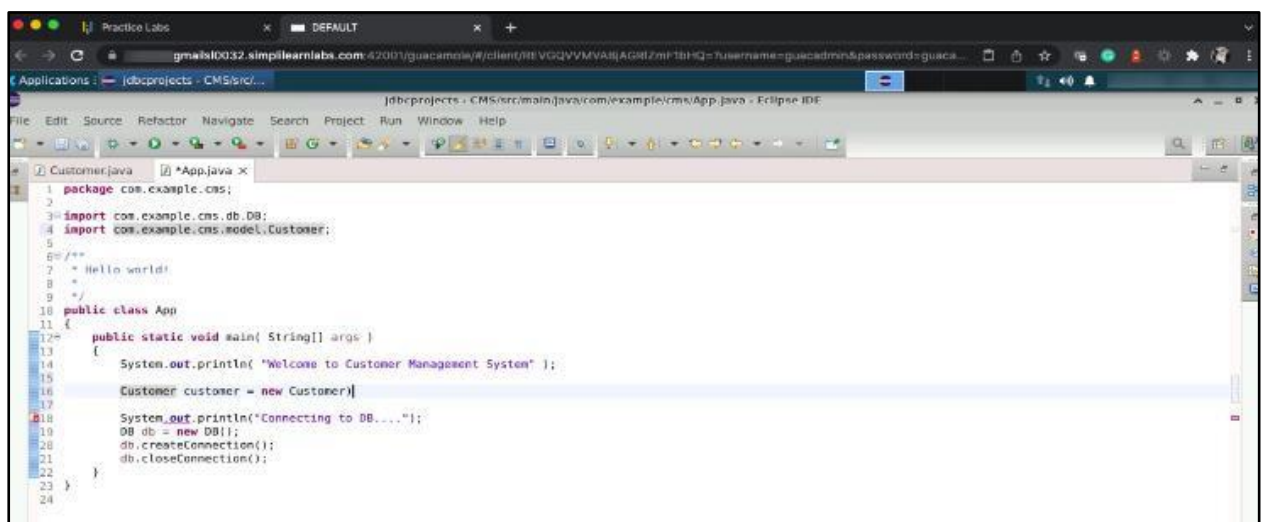


1.12 Right-click **Source** and select **Generate toString()** to decide the location of data in the object as shown in the screenshot below:

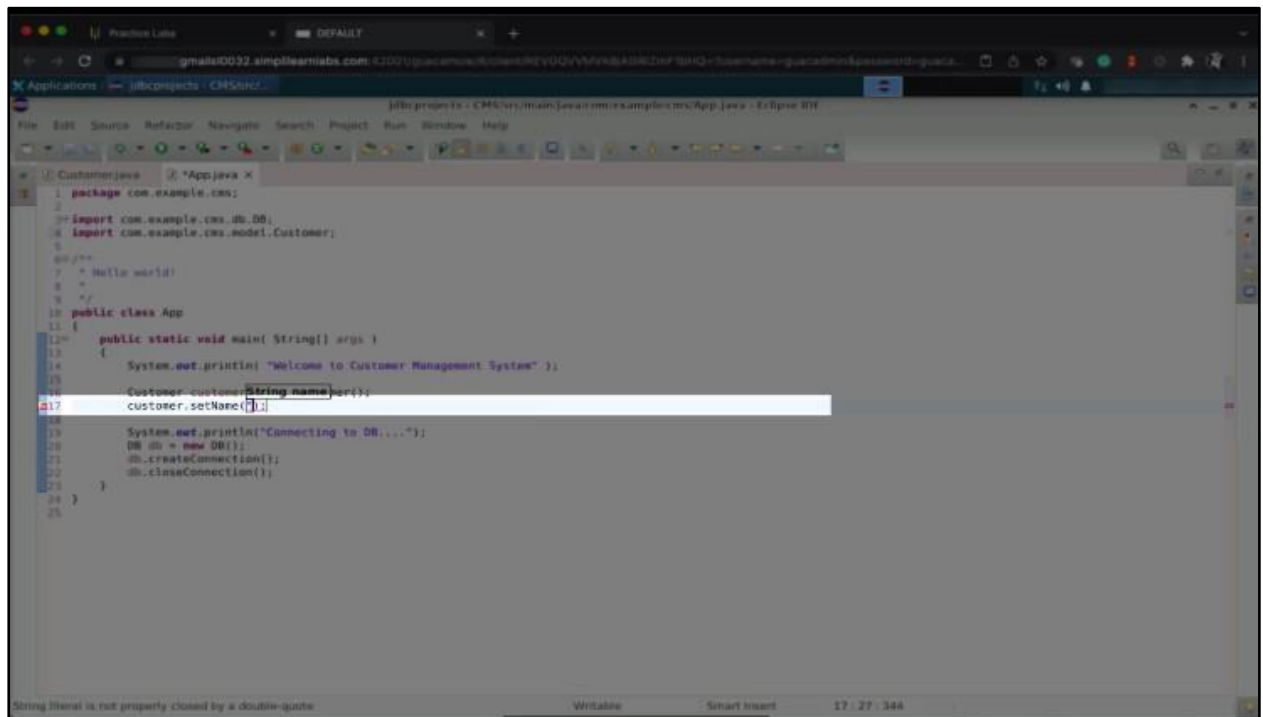


Step 2: Write a customer object in App.java

2.1 Define a **Customer** object in the **App.java** file



2.2 Set the name for the **Customer** object as shown in the screenshot below:

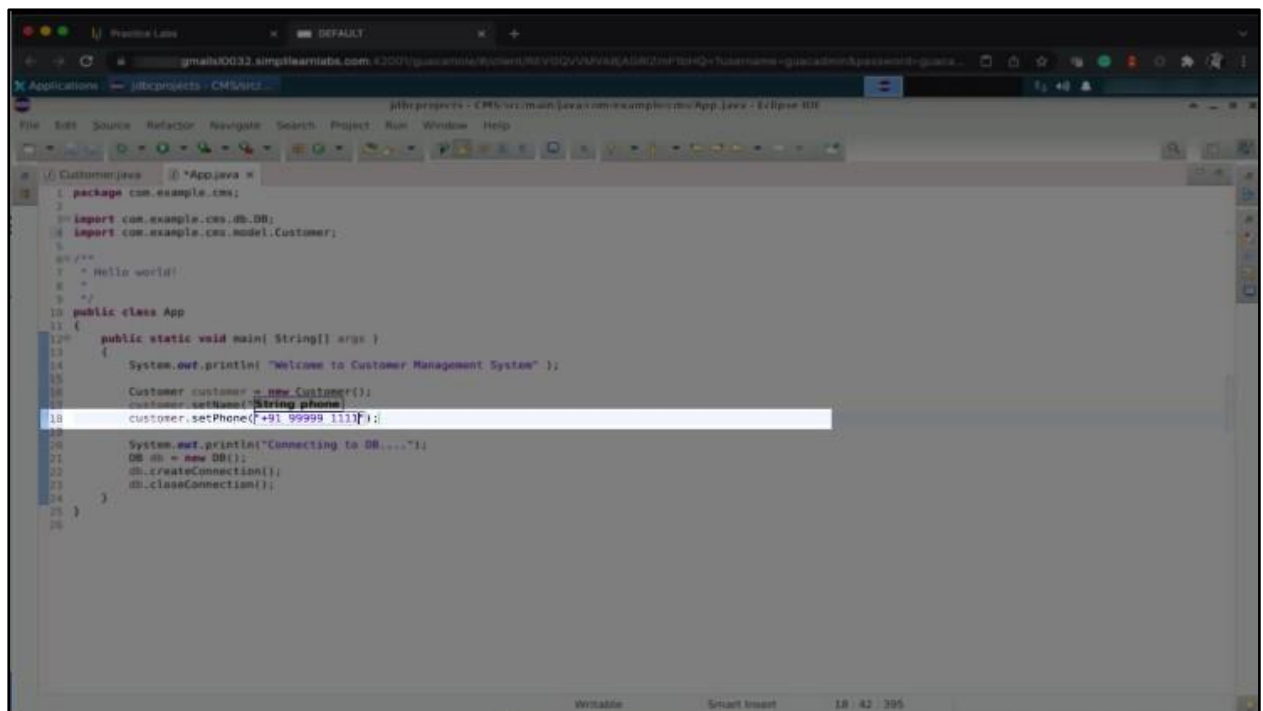


```

1 package com.example.cms;
2
3 import com.example.cms.db.DB;
4 import com.example.cms.model.Customer;
5
6 /**
7  * Hello world!
8  *
9  */
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println( "Welcome to Customer Management System" );
15
16         Customer customer = new Customer();
17         customer.setName('');
18
19         System.out.println("Connecting to DB...");
20         DB db = new DB();
21         db.createConnection();
22         db.closeConnection();
23     }
24 }
25

```

2.3 Create a method called **setPhone** with details as shown in the below screenshot:

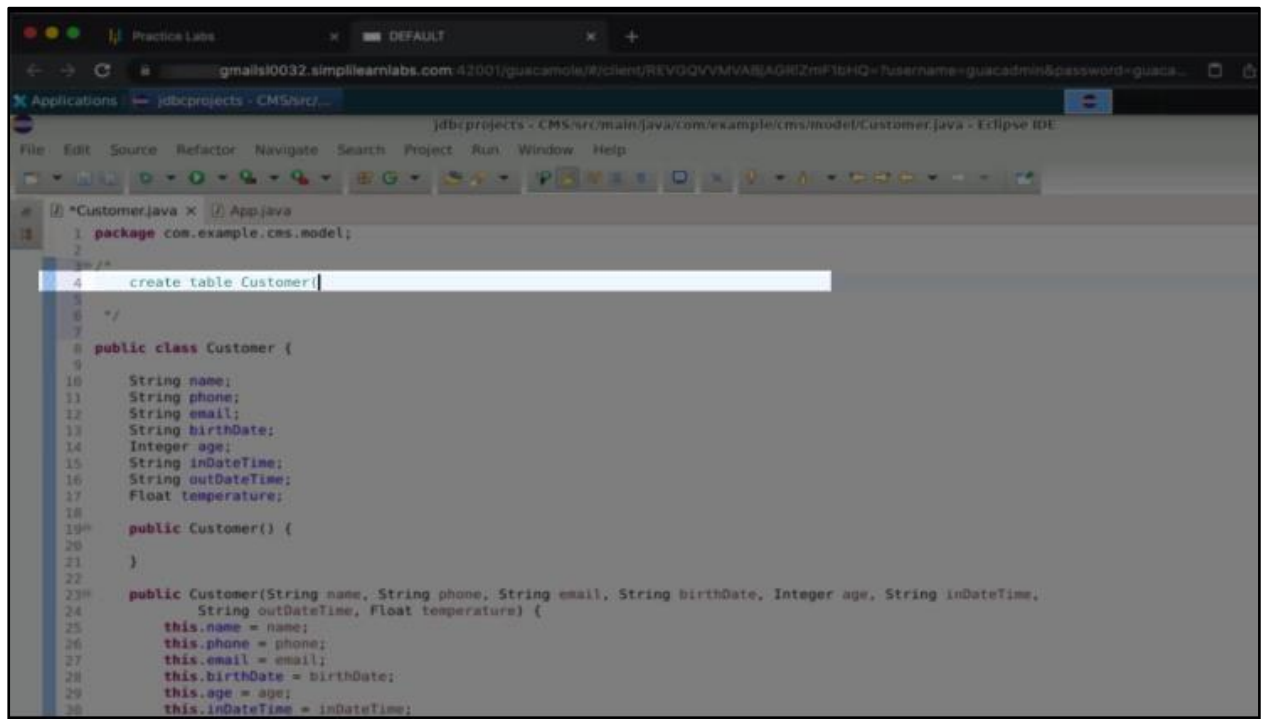


```

1 package com.example.cms;
2
3 import com.example.cms.db.DB;
4 import com.example.cms.model.Customer;
5
6 /**
7  * Hello world!
8  *
9  */
10 public class App
11 {
12     public static void main( String[] args )
13     {
14         System.out.println( "Welcome to Customer Management System" );
15
16         Customer customer = new Customer();
17         customer.setName('');
18         customer.setPhone('+91 99999 1111');
19
20         System.out.println("Connecting to DB...");
21         DB db = new DB();
22         db.createConnection();
23         db.closeConnection();
24     }
25 }
26

```

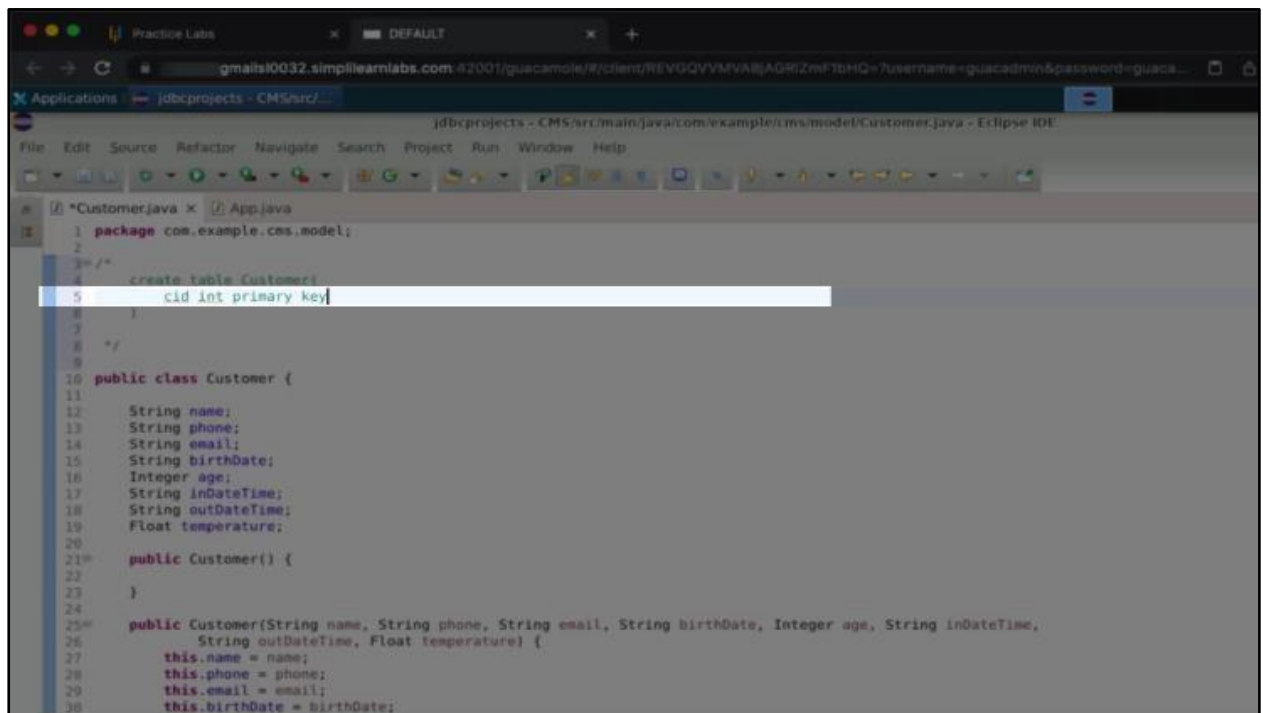

2.4 Use the **Customer** class and type **create table Customer** for object-relational mapping



```

1 package com.example.cms.model;
2
3 /**
4  * create table Customer
5  */
6
7
8 public class Customer {
9
10     String name;
11     String phone;
12     String email;
13     String birthDate;
14     Integer age;
15     String inDateTime;
16     String outDateTime;
17     Float temperature;
18
19     public Customer() {
20     }
21
22
23     public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
24                     String outDateTime, Float temperature) {
25         this.name = name;
26         this.phone = phone;
27         this.email = email;
28         this.birthDate = birthDate;
29         this.age = age;
30         this.inDateTime = inDateTime;
  
```

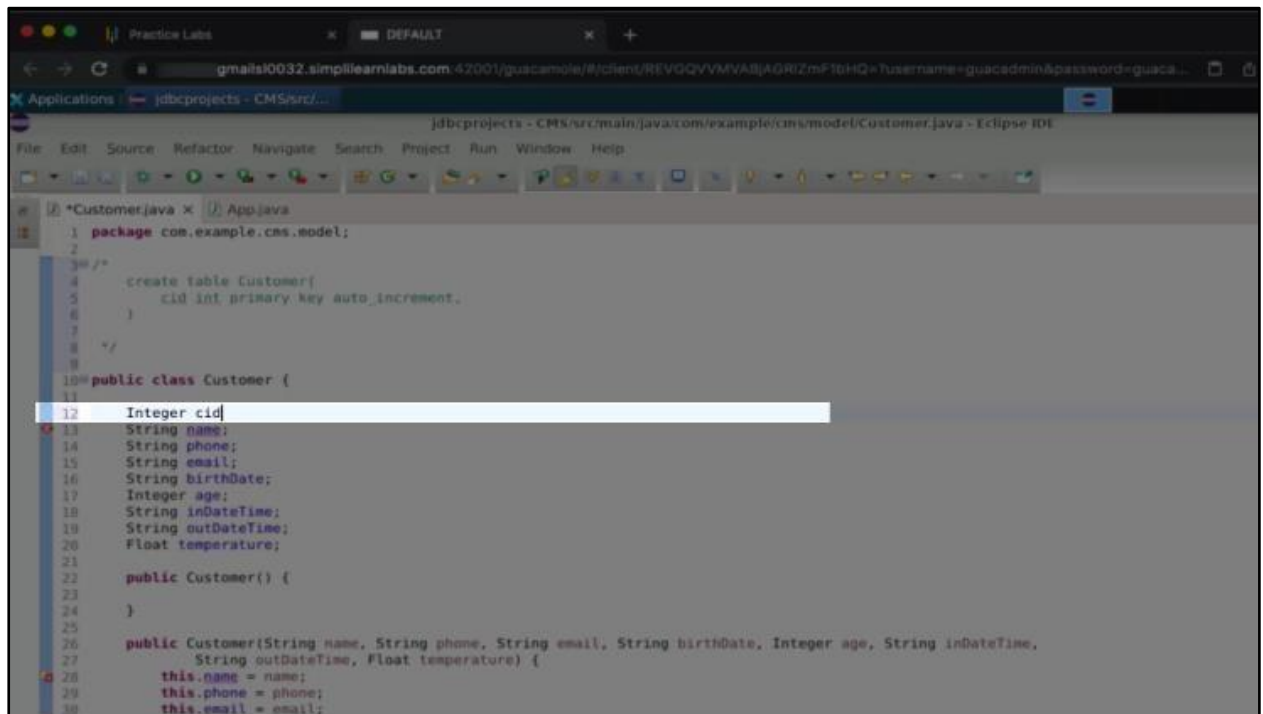
2.5 Add a primary key as **cid** for a unique identifier as shown below:



```

1 package com.example.cms.model;
2
3 /**
4  * create table Customer
5  * cid int primary key
6  */
7
8
9 public class Customer {
10
11     String name;
12     String phone;
13     String email;
14     String birthDate;
15     Integer age;
16     String inDateTime;
17     String outDateTime;
18     Float temperature;
19
20     public Customer() {
21     }
22
23
24     public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
25                     String outDateTime, Float temperature) {
26         this.name = name;
27         this.phone = phone;
28         this.email = email;
29         this.birthDate = birthDate;
  
```

2.6 Add an integer attribute **cid** as shown in the below screenshot:



```

package com.example.cms.model;

/**
 * create table Customer{
 *   cid int primary key auto increment,
 * }
 */

public class Customer {

    Integer cid
    String name;
    String phone;
    String email;
    String birthDate;
    Integer age;
    String inDateTime;
    String outDateTime;
    Float temperature;

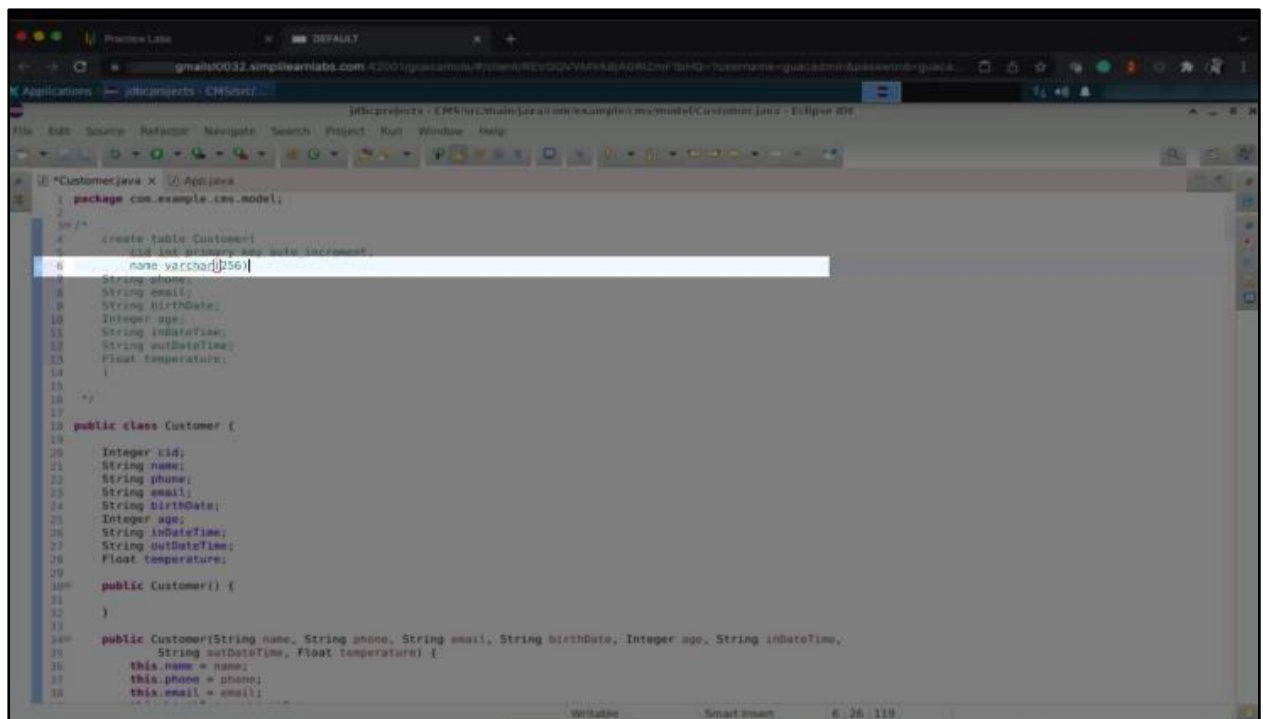
    public Customer() {

    }

    public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
        String outDateTime, Float temperature) {
        this.name = name;
        this.phone = phone;
        this.email = email;
    }
}

```

2.7 Specify the **name** attribute with the datatype **varchar(256)** as shown in the below screenshot:



```

package com.example.cms.model;

/**
 * create table Customer{
 *   cid int primary key auto increment,
 *   name varchar(256)
 * }
 */

public class Customer {

    Integer cid;
    String name;
    String phone;
    String email;
    String birthDate;
    Integer age;
    String inDateTime;
    String outDateTime;
    Float temperature;

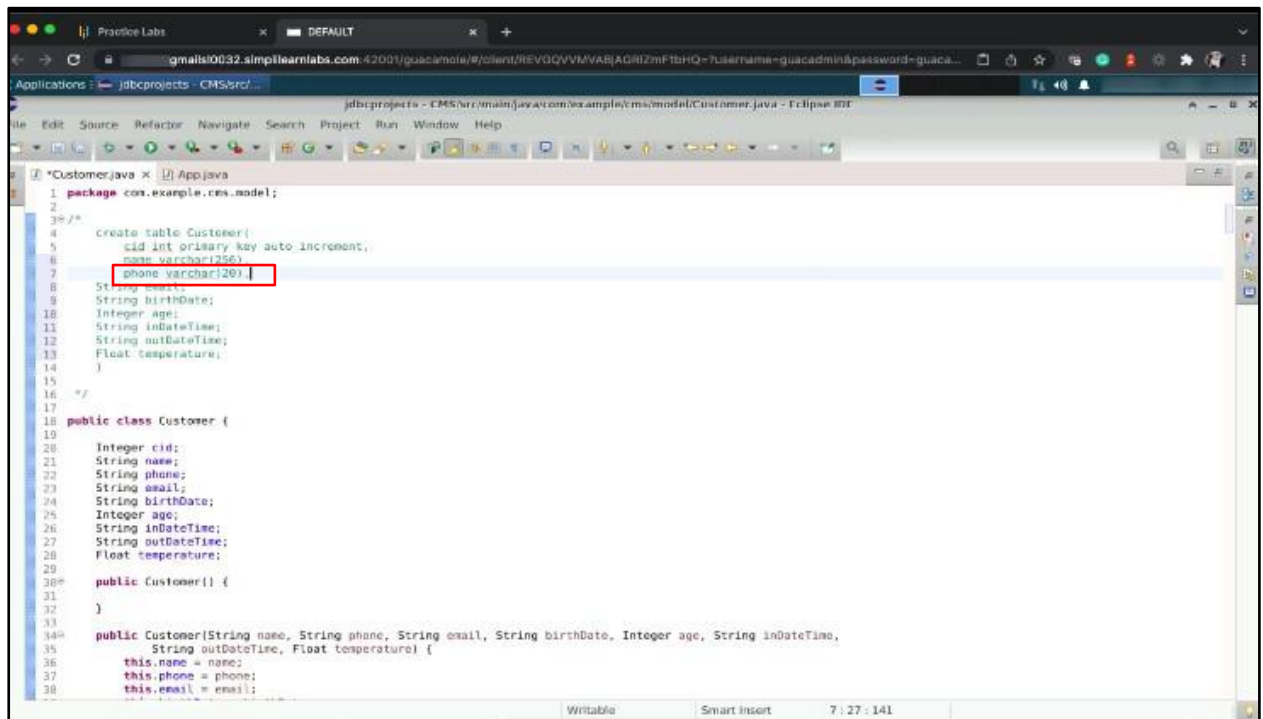
    public Customer() {

    }

    public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
        String outDateTime, Float temperature) {
        this.name = name;
        this.phone = phone;
        this.email = email;
    }
}

```

2.8 Add the **phone** attribute with the datatype **varchar(20)** as shown in the below screenshot:

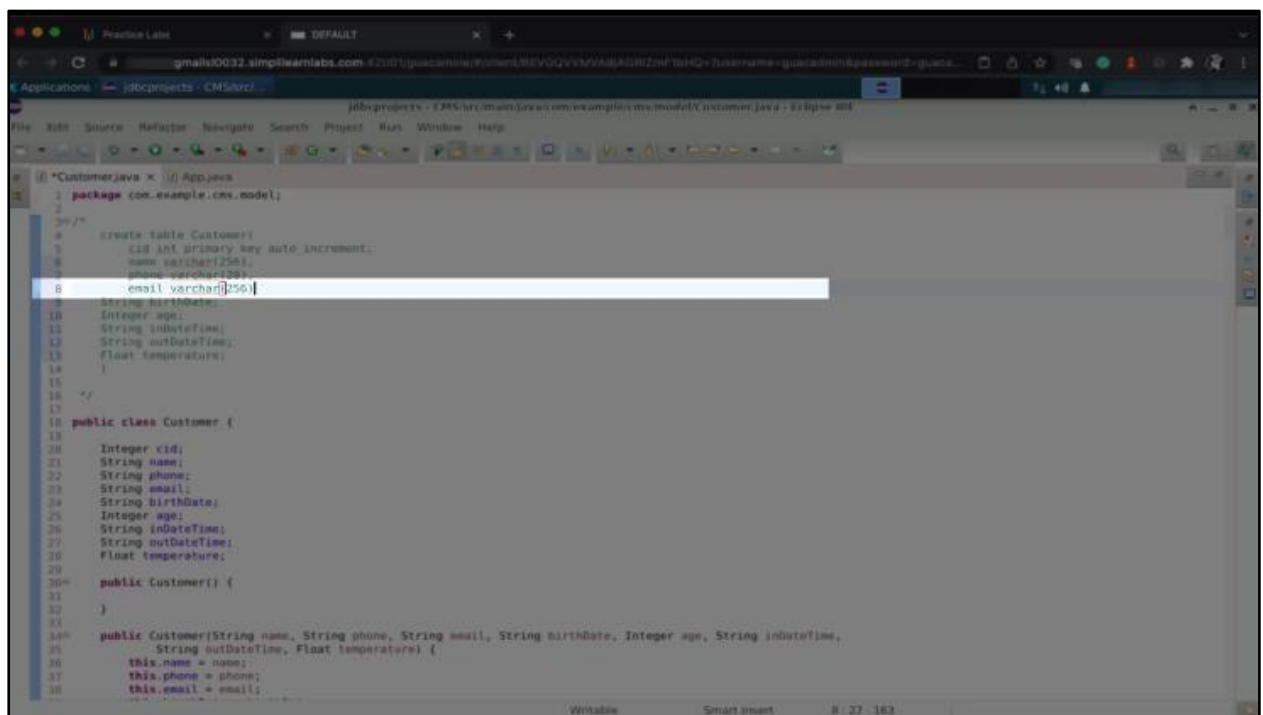


```

1 package com.example.cms.model;
2
3 /**
4  * create table Customer(
5  *   cid int primary key auto increment,
6  *   name varchar(256),
7  *   phone varchar(20),
8  *   String email;
9  *   String birthDate;
10  *   Integer age;
11  *   String inDateTime;
12  *   String outDateTime;
13  *   Float temperature;
14  * )
15  */
16
17 public class Customer {
18
19     Integer cid;
20     String name;
21     String phone;
22     String email;
23     String birthDate;
24     Integer age;
25     String inDateTime;
26     String outDateTime;
27     Float temperature;
28
29     public Customer() {
30
31     }
32
33     public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
34         String outDateTime, Float temperature) {
35         this.name = name;
36         this.phone = phone;
37         this.email = email;
38     }

```

2.9 Add the **email** attribute with the datatype **varchar(256)** as shown in the below screenshot:



```

1 package com.example.cms.model;
2
3 /**
4  * create table Customer(
5  *   cid int primary key auto increment,
6  *   name varchar(256),
7  *   phone varchar(20),
8  *   email varchar(256),
9  *   String birthDate;
10  *   Integer age;
11  *   String inDateTime;
12  *   String outDateTime;
13  *   Float temperature;
14  * )
15  */
16
17 public class Customer {
18
19     Integer cid;
20     String name;
21     String phone;
22     String email;
23     String birthDate;
24     Integer age;
25     String inDateTime;
26     String outDateTime;
27     Float temperature;
28
29     public Customer() {
30
31     }
32
33     public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
34         String outDateTime, Float temperature) {
35         this.name = name;
36         this.phone = phone;
37         this.email = email;
38     }

```

2.10 Set the **birthday** attribute with the datatype **date** as shown in the screenshot below:

```

1 package com.example.cms.model;
2
3 /**
4  * create table Customer(
5  *   cid int primary key auto increment,
6  *   name varchar(256),
7  *   phone varchar(20),
8  *   email varchar(256),
9  *   birthday date,
10 *   age int,
11 *   inDateTime datetime,
12 *   outDateTime datetime,
13 *   temperature float
14 * )
15 */
16
17 public class Customer {
18     Integer cid;
19     String name;
20     String phone;
21     String email;
22     String birthday;
23     Integer age;
24     String inDateTime;
25     String outDateTime;
26     Float temperature;
27
28     public Customer() {
29     }
30
31     public Customer(String name, String phone, String email, String birthday, Integer age, String inDateTime,
32         String outDateTime, Float temperature) {
33         this.name = name;
34         this.phone = phone;
35         this.email = email;
36     }
37 }

```

2.11 Add a floating-point attribute for **temperature** and a datetime attribute for **outDateTime** as shown in the screenshot below:

```

1 package com.example.cms.model;
2
3 /**
4  * create table Customer(
5  *   cid int primary key auto increment,
6  *   name varchar(256),
7  *   phone varchar(20),
8  *   email varchar(256),
9  *   birthday date,
10 *   age int,
11 *   inDateTime datetime,
12 *   outDateTime datetime,
13 *   temperature float
14 * )
15 */
16
17 public class Customer {
18     Integer cid;
19     String name;
20     String phone;
21     String email;
22     String birthday;
23     Integer age;
24     String inDateTime;
25     String outDateTime;
26     Float temperature;
27
28     public Customer() {
29     }
30
31     public Customer(String name, String phone, String email, String birthday, Integer age, String inDateTime,
32         String outDateTime, Float temperature) {
33         this.name = name;
34         this.phone = phone;
35         this.email = email;
36     }
37 }

```

2.12 Copy the entire **Customer** table query

```

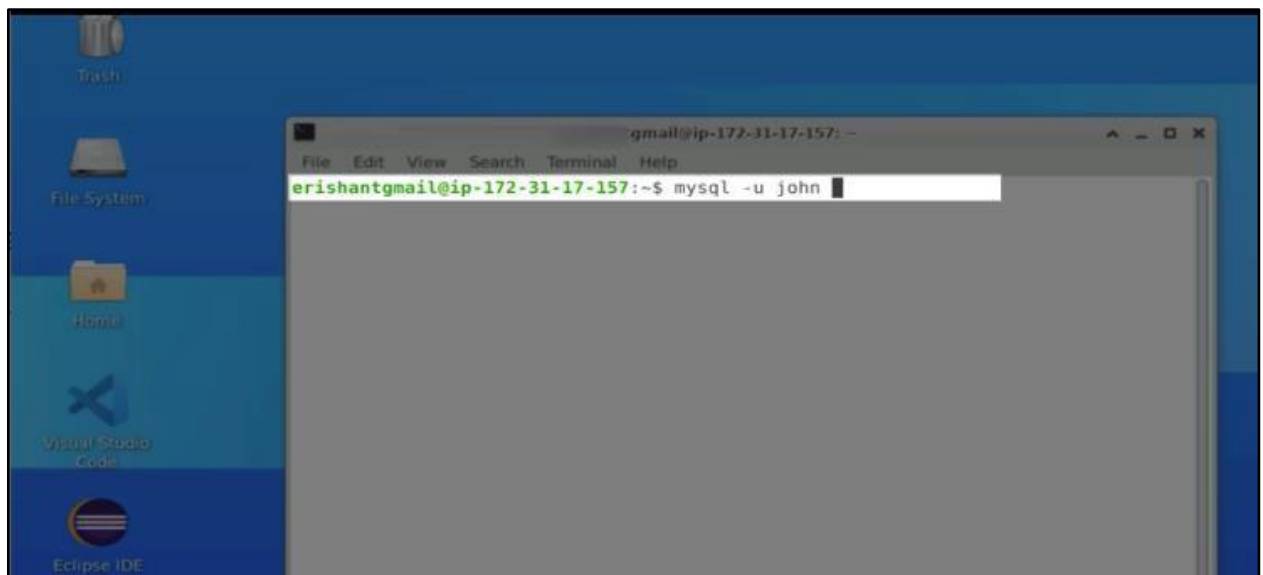
1 package com.example.cms.model;
2
3 /*
4  * create table Customer(
5  *   cid int primary key auto increment,
6  *   name varchar(256),
7  *   phone varchar(20),
8  *   email varchar(256),
9  *   birthDate date,
10 *   age int,
11 *   inDateTime datetime,
12 *   outDateTime datetime,
13 *   temperature float
14 * )
15 */
16
17 public class Customer {
18
19     Integer cid;
20     String name;
21     String phone;
22     String email;
23     String birthDate;
24     Integer age;
25     String inDateTime;
26     String outDateTime;
27     Float temperature;
28
29     public Customer() {
30
31     }
32
33     public Customer(String name, String phone, String email, String birthDate, Integer age, String inDateTime,
34                       String outDateTime, Float temperature) {
35         this.name = name;
36         this.phone = phone;
37         this.email = email;
38     }
39 }

```

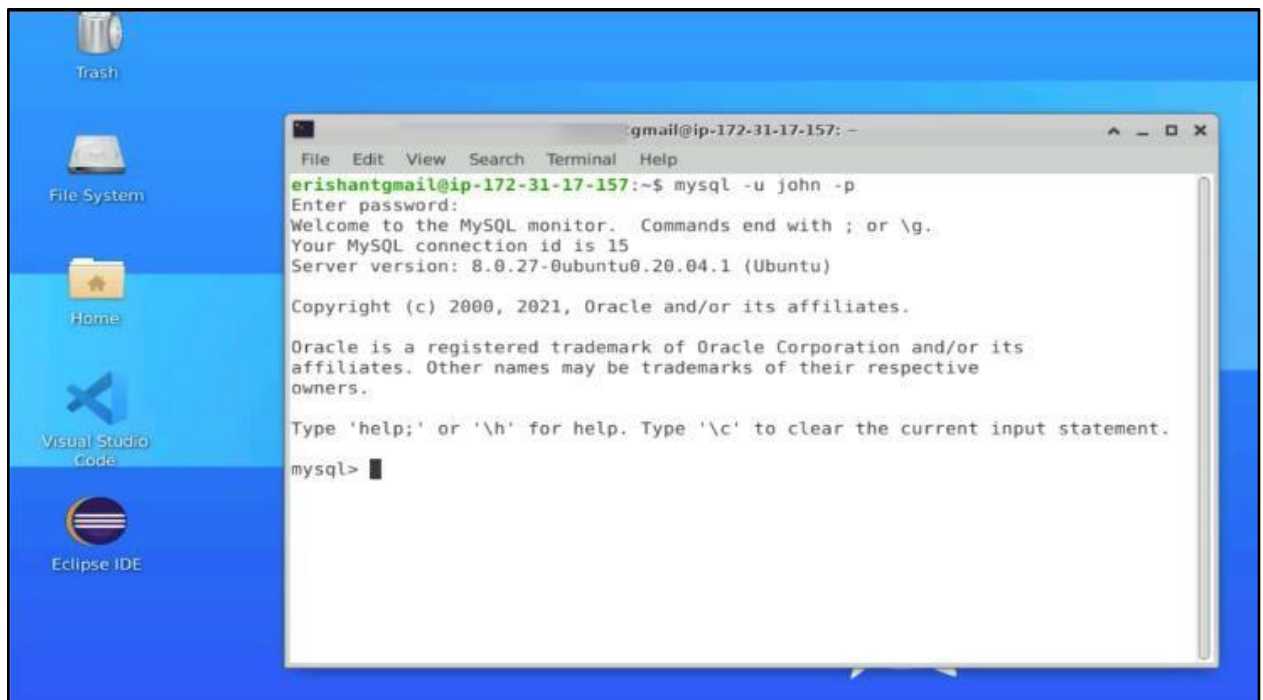
The word 'cid' is not correctly spelled

Step 3: Open the terminal window

3.1 Open the terminal window and log in to MySQL using the command **mysql -u john -p**

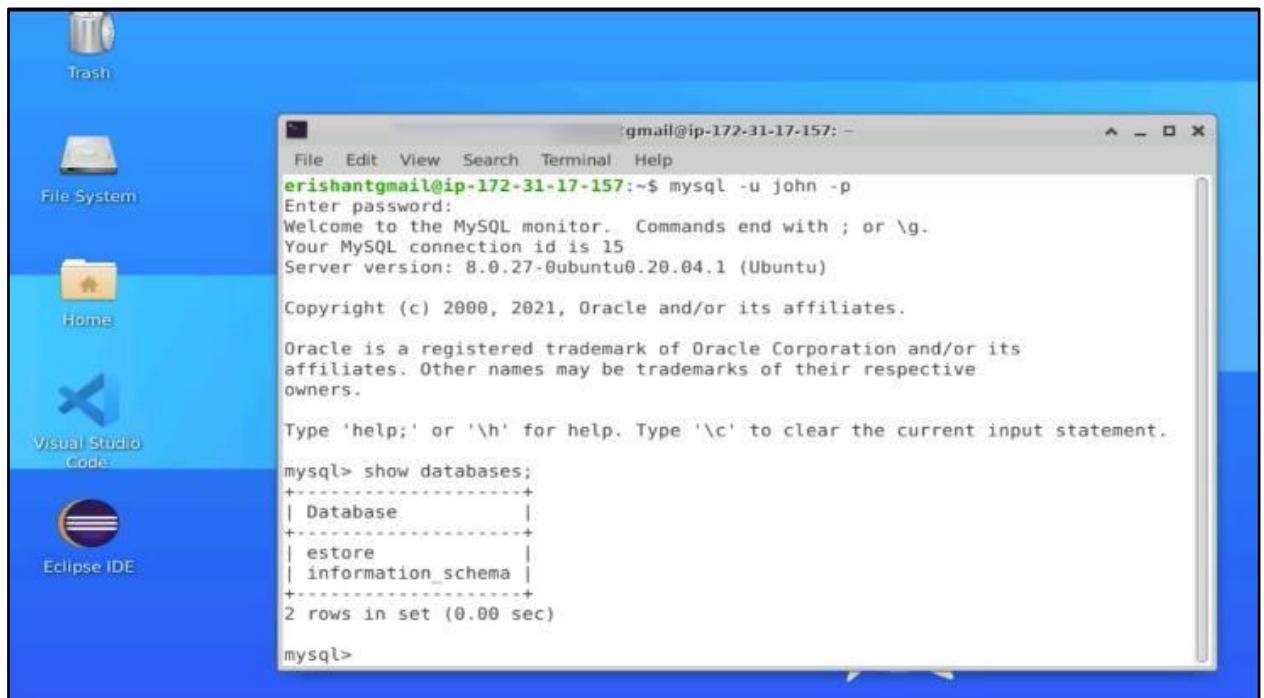


3.2 Enter the password



Note: A user named **John** has already been created for the database.

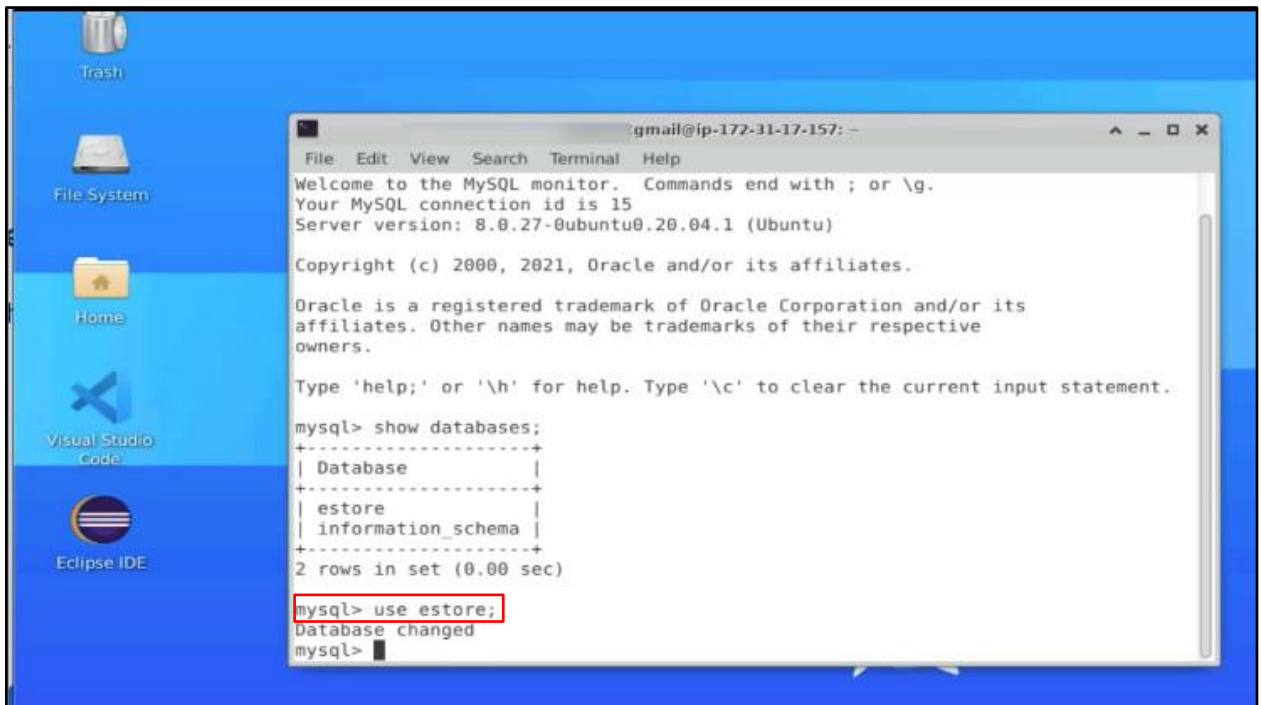
3.3 Type the **show databases;** command to check the available database names as shown in the screenshot below:



The screenshot shows a terminal window titled 'gmail@ip-172-31-17-157: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The user has executed the command `mysql -u john -p`. The terminal output shows the MySQL prompt, a password prompt, and a welcome message. The user then enters `show databases;`, which returns a table of databases: 'estore' and 'information_schema'. The terminal also shows the MySQL version and copyright information.

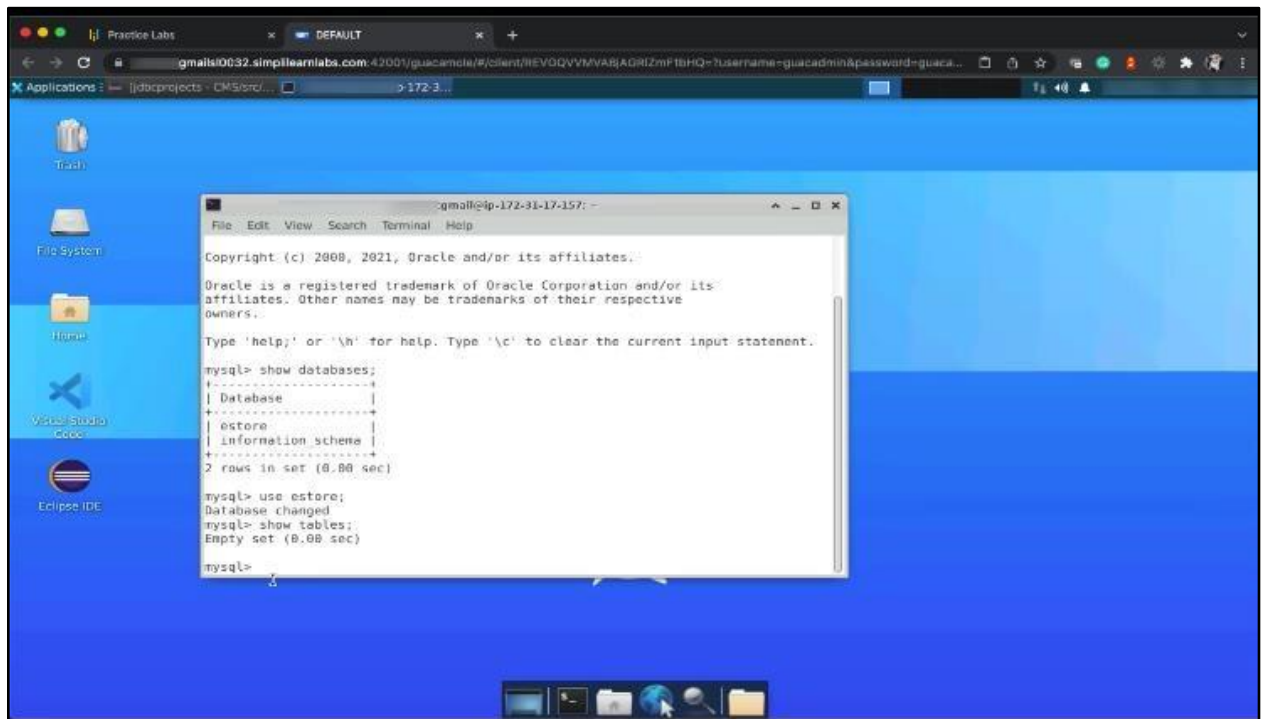
```
gmail@ip-172-31-17-157: ~  
File Edit View Search Terminal Help  
erishantgmail@ip-172-31-17-157:~$ mysql -u john -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 15  
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2021, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| estore   |  
| information_schema |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql>
```

3.4 Enter **use estore;** to work on the estore databases shown in the screenshot below:

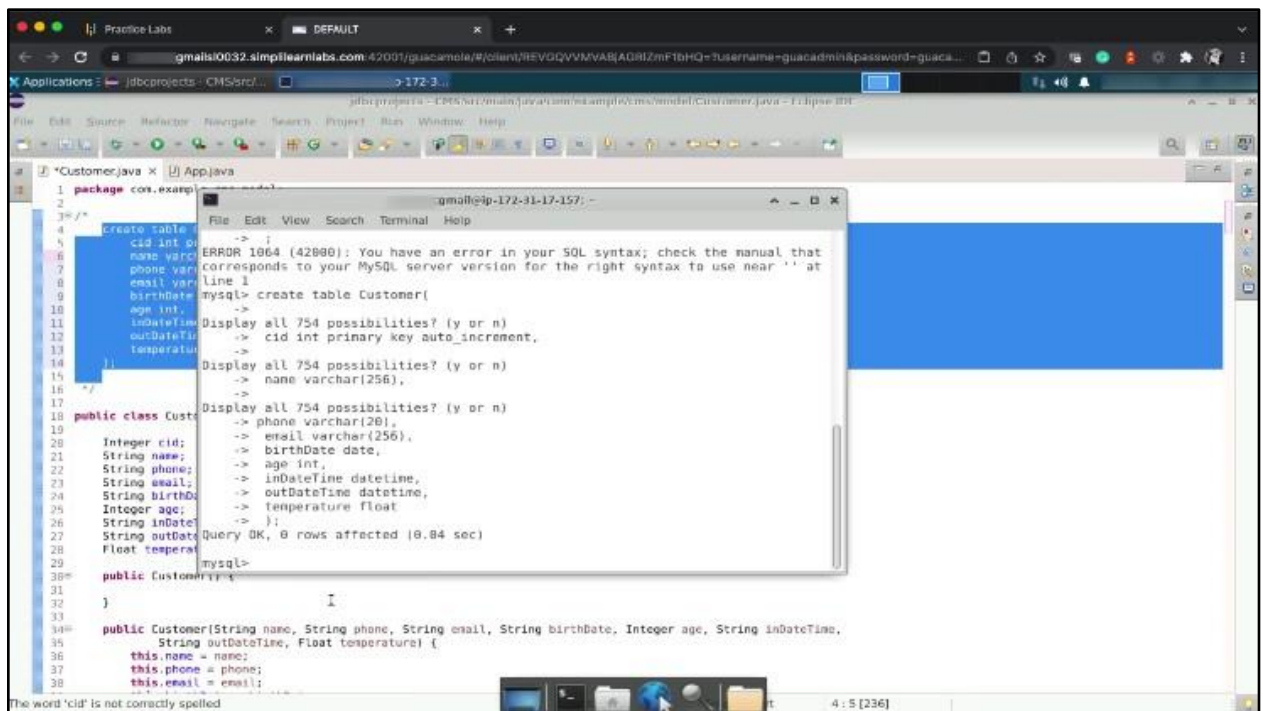


```
gmail@ip-172-31-17-157: ~  
File Edit View Search Terminal Help  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 15  
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2021, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| estore   |  
| information_schema |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql> use estore;  
Database changed  
mysql>
```

3.5 Enter **show tables;** to check tables available in the estore database



3.6 Paste the **Customer** table copied in step 2.12



3.7 Run the **show tables;** command to see the changes

```

package com.example.demo;

import java.sql.*;

public class Customer {

    Integer cid;
    String name;
    String phone;
    String email;
    String birthDate;
    Integer age;
    String inDateTime;
    String outDateTime;
    Float temperature;

    public Customer() {
        // ...
    }
}

```

```

mysql> show tables;
+-----+
| Tables_in_estore |
+-----+
| Customer          |
+-----+
1 row in set (0.00 sec)

```

3.8 Enter the **describe Customer;** command to view the customer table details

```

mysql> describe Customer;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cid   | int  | NO   | PRI | NULL    | auto_increment |
| name  | varchar(256) | YES |     | NULL    |               |
| phone | varchar(20) | YES |     | NULL    |               |
| email | varchar(256) | YES |     | NULL    |               |
| birthDate | date | YES |     | NULL    |               |
| age   | int  | YES |     | NULL    |               |
| inDateTime | datetime | YES |     | NULL    |               |
| outDateTime | datetime | YES |     | NULL    |               |
| temperature | float | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

```

3.9 Enter **select * from Customer;** to check the data in the table

```

package com.example.cms.model;

import java.util.Date;

public class Customer {
    Integer cid;
    String name;
    String phone;
    String email;
    String birthDate;
    Integer age;
    String inDateTime;
    String outDateTime;
    Float temperature;
}

```

```

mysql> describe Customer;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cid   | int  | NO   | PRI | NULL    | auto_increment |
| name  | varchar(256) | YES |     | NULL    |               |
| phone | varchar(20) | YES |     | NULL    |               |
| email | varchar(256) | YES |     | NULL    |               |
| birthDate | date | YES |     | NULL    |               |
| age   | int  | YES |     | NULL    |               |
| inDateTime | datetime | YES |     | NULL    |               |
| outDateTime | datetime | YES |     | NULL    |               |
| temperature | float | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

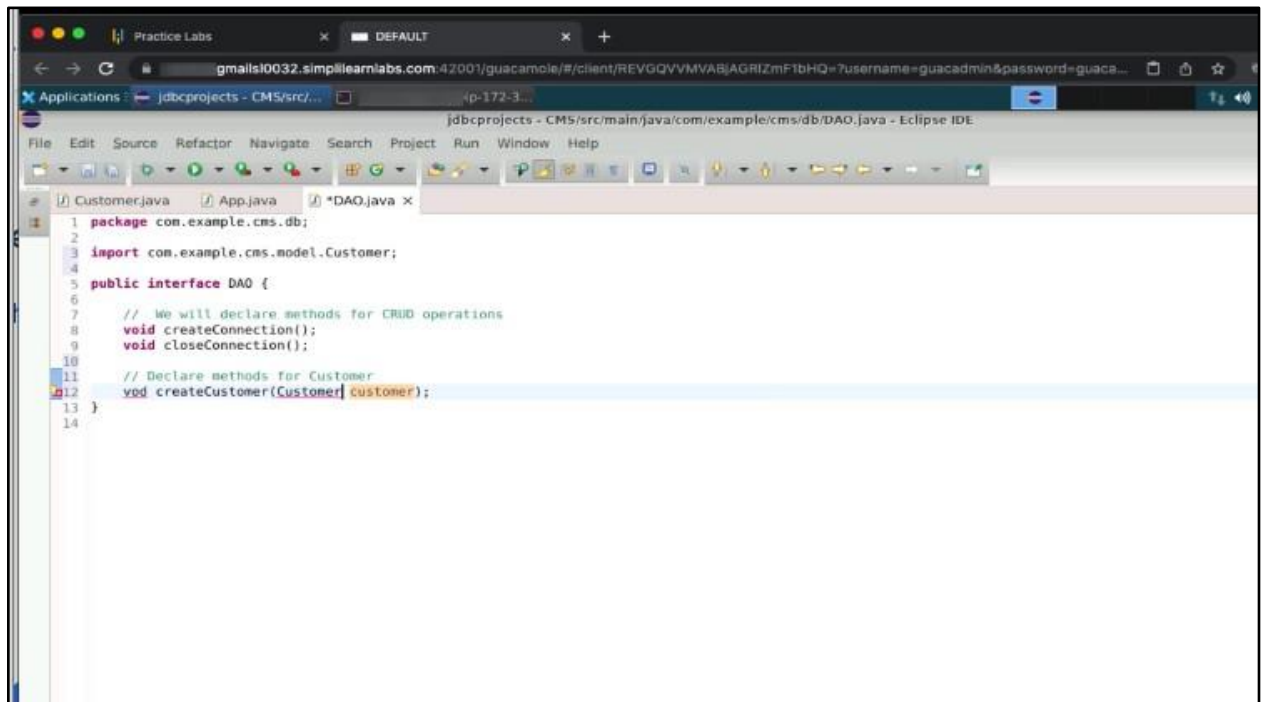
mysql> select * from Customer;
Empty set (0.00 sec)

mysql>

```

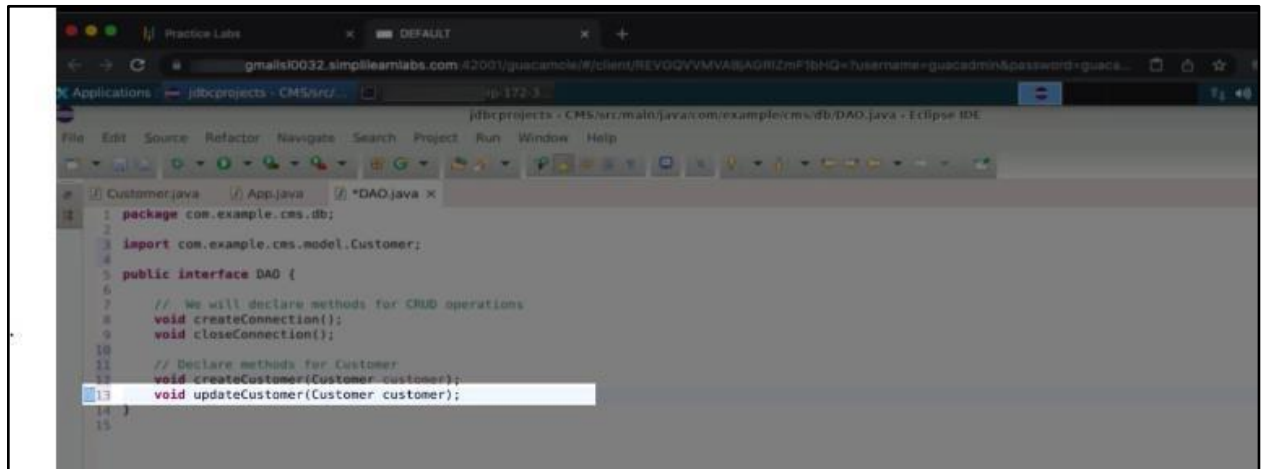
Step 4: Declare a method for a customer in the DAO interface

4.1 Declare methods for a customer in the DAO interface. The first one is **createCustomer()** that takes a single customer as input.



```
1 package com.example.cms.db;
2
3 import com.example.cms.model.Customer;
4
5 public interface DAO {
6
7     // We will declare methods for CRUD operations
8     void createConnection();
9     void closeConnection();
10
11     // Declare methods for Customer
12     void createCustomer(Customer customer);
13 }
14
```


4.2 Additionally, implement **updateCustomer()** which also takes a customer as input for making updates

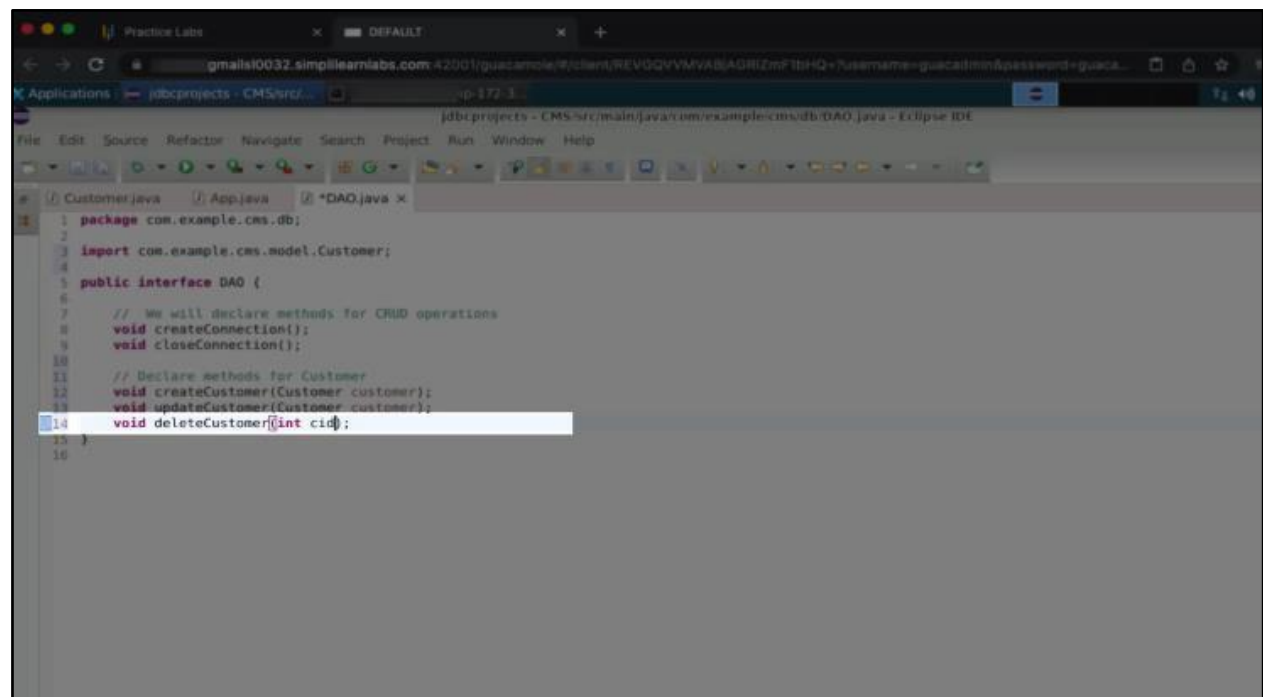


```

1 package com.example.cms.db;
2
3 import com.example.cms.model.Customer;
4
5 public interface DAO {
6
7     // We will declare methods for CRUD operations
8     void createConnection();
9     void closeConnection();
10
11     // Declare methods for Customer
12     void createCustomer(Customer customer);
13     void updateCustomer(Customer customer);
14 }
15

```

4.3 Create **deleteCustomer()**, which takes a single input **cid**

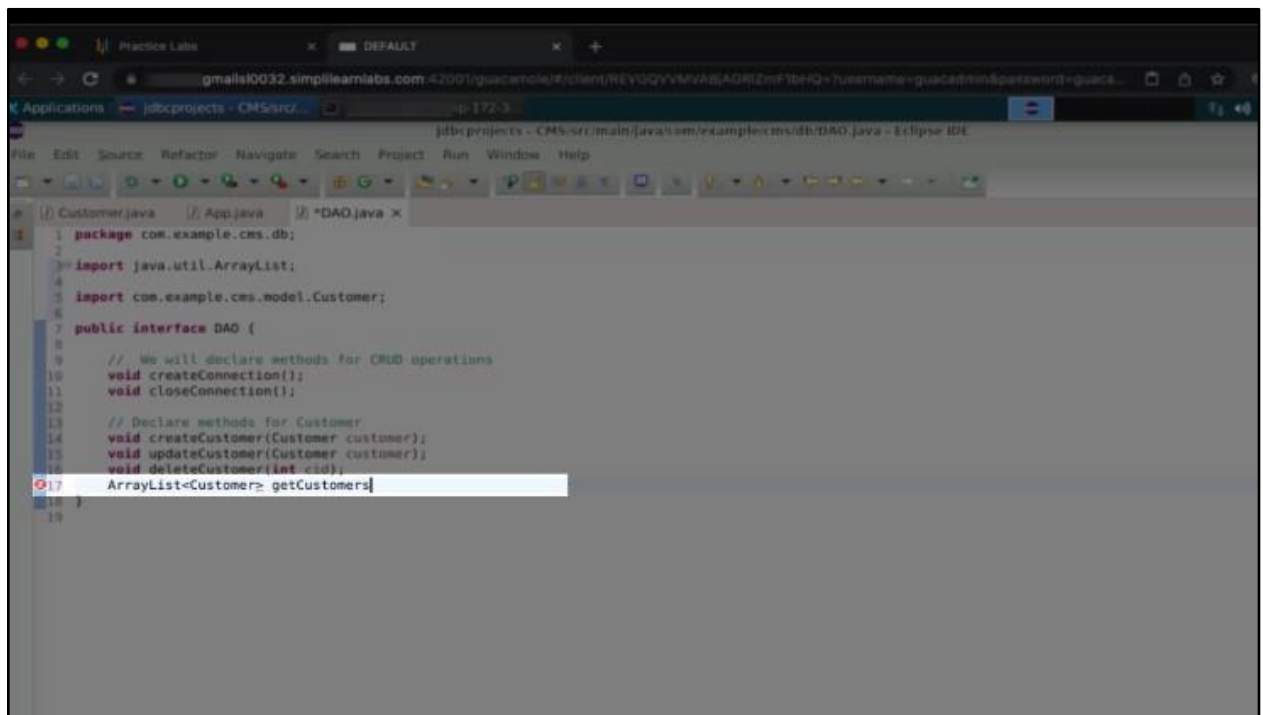


```

1 package com.example.cms.db;
2
3 import com.example.cms.model.Customer;
4
5 public interface DAO {
6
7     // We will declare methods for CRUD operations
8     void createConnection();
9     void closeConnection();
10
11     // Declare methods for Customer
12     void createCustomer(Customer customer);
13     void updateCustomer(Customer customer);
14     void deleteCustomer(int cid);
15 }
16

```

4.4 Implement the **getAllCustomers()** method to retrieve the list of customers

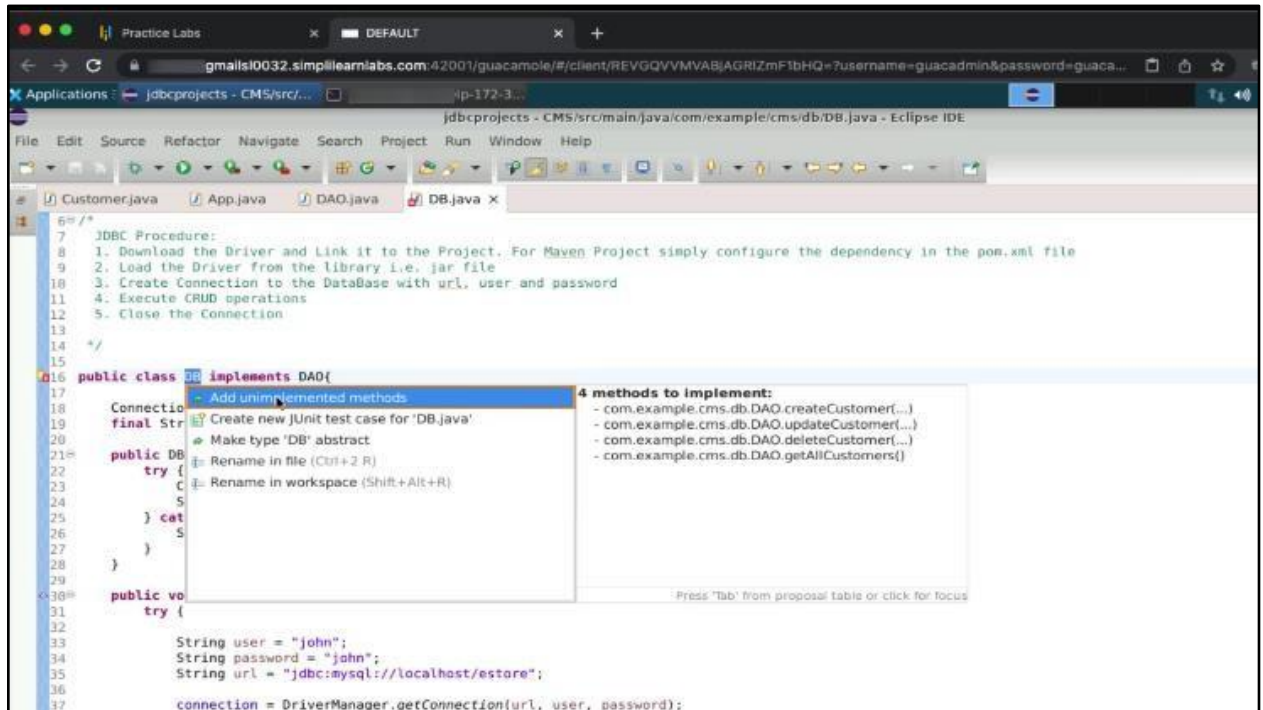


```

1 package com.example.cms.db;
2
3 import java.util.ArrayList;
4
5 import com.example.cms.model.Customer;
6
7 public interface DAO {
8
9     // We will declare methods for CRUD operations
10    void createConnection();
11    void closeConnection();
12
13    // Declare methods for Customer
14    void createCustomer(Customer customer);
15    void updateCustomer(Customer customer);
16    void deleteCustomer(int cid);
17    ArrayList<Customer> getAllCustomers();
18 }
19

```

4.5 Navigate to the **DB.java** file where you can see the error, and click on the error to add the unimplemented methods



By following these steps, you have successfully created a model named **Customer** and established a corresponding table in the database using ORM. Additionally, we have implemented methods following the DAO design pattern to perform CRUD operations on the data.