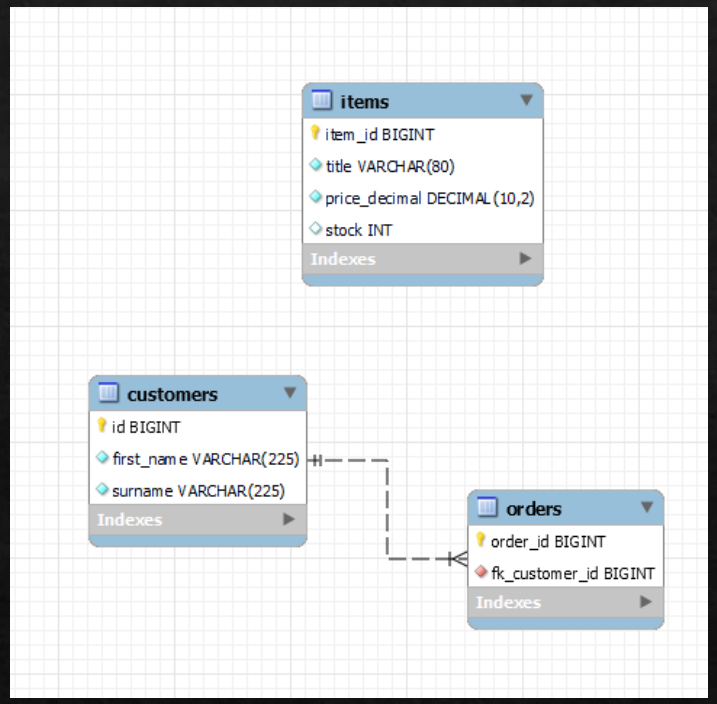
```
Welcome to the Inventory Management System!
Which entity would you like to use?
CUSTOMER: Information about customers
ITEM: Individual Items
ORDER: Purchases of items
ORDER_ITEM: Adding an item(s) to an order
STOP: To close the application
```

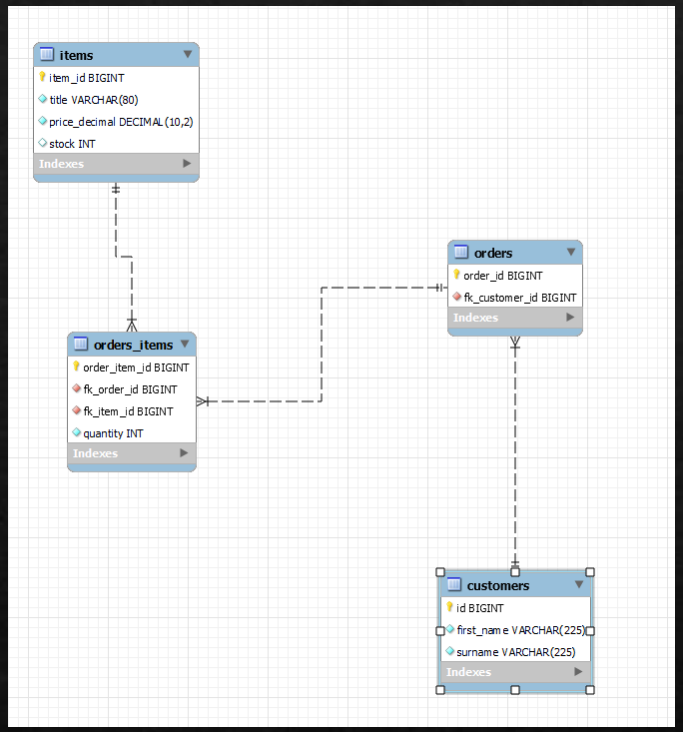
IMS Project

Rebecca Swinton

How did you approach the spec?

- Thinking about what tables had to be made
- What attributes each table had to have in order to describe what each table was doing and what was going to be stored in it
- How they would relate to one another (foreign keys)
- Orders_item table being the ‘middle man’ for relating items back to orders and customers





As many customers can make many orders and many items can have many orders

To avoid a many to many relationship, a relational database is created to separate the chaos. The customers table contains a list of customers, the orders table contains a list of orders, which can be linked back to customers, an items table contains the list of products and the order_items table contains a list of order items which is linked to both the item and order tables

```sql
create database ims;
use ims;
create table customers (
id bigint primary key auto_increment not null,
first_name varchar(225) not null,
surname varchar(225) not null
);
create table items (
item_id bigint primary key auto_increment not null,
`title` varchar(80) not null,
`price_decimal` decimal(10,2) not null,
`stock` int
);
create table orders (
order_id bigint primary key auto_increment not null,
fk_customer_id bigint not null,
foreign key (fk_customer_id) references customers(id)
);
create table orders_items (
order_item_id bigint primary key auto_increment not null,
fk_order_id bigint not null,
fk_item_id bigint not null,
quantity int not null,
foreign key (fk_order_id) references orders(order_id),
foreign key (fk_item_id) references items(item_id)
);
```

IMS database created in mysql, to show the separate tables and how they relate to each other with the foreign keys being the middle man across the database.

Consultant journey – What technologies have you learnt so far?

- Version Control System : Git, ensuring that I am making branches to work on separate parts of code, making commits and pushing those to GitHub
- Source Code Management – GitHub, I have learnt how to fork repositories, create pull requests and merge branches
- Scrum Board – Jira, track project management and planning. I have learnt how to create sprints based on the epic, user stories and tasks of developing an app
- Database Management System – SQL, to create the IMS database with the 4 relational tables and connecting those to Java
- Backend Programming Language – Java, I am able to follow coding best practices (SOLID principles), making sure each class have methods/functions that only relate to that class .

Consultant journey – What technologies have you learnt so far?

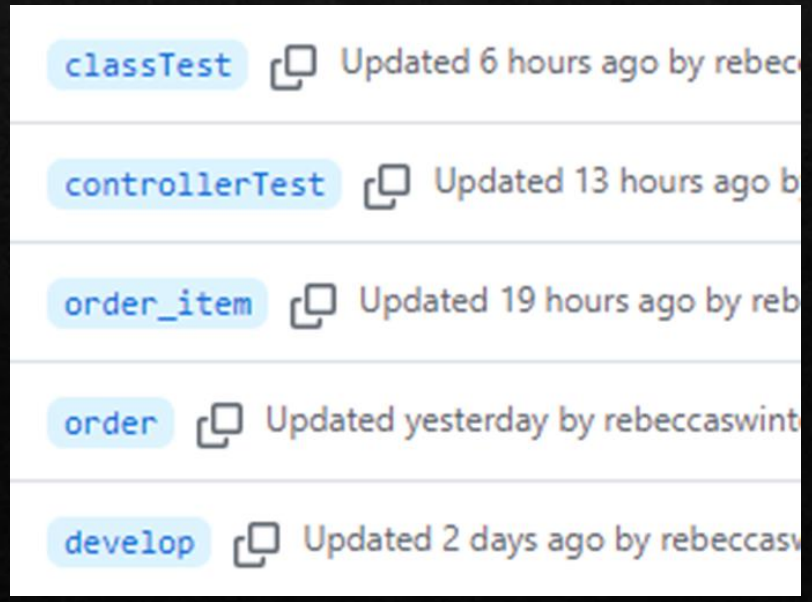
Build Tools – Maven, creating maven projects and using them to create maintainbale models for projects. Adding dependencies to pom.xl files to make sure objects are configured properly.

Unit Testing – Junit and Mockito, checking that every piece of code has functionality and is working how it is supposed to.

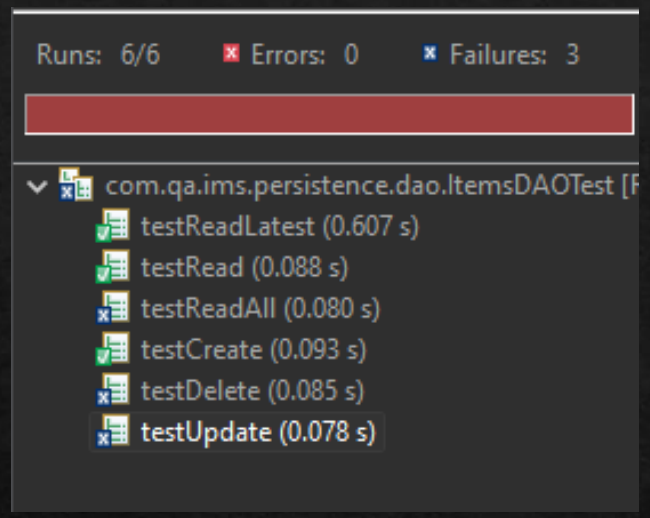
Using these tools helped me understand the scope of the project and how they're to be implemented for the overall result.

CI: How did you approach version control?

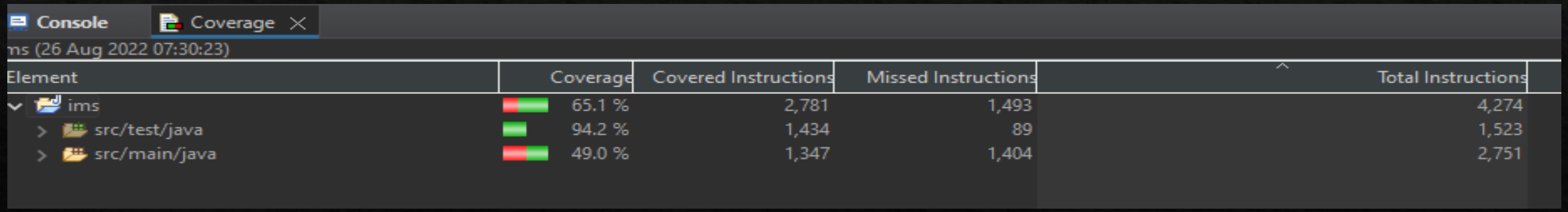
- Following branching best practices for working on different parts of code to ensure code modifications are being tracked and protected.
- Making sure to commit changes often, to show my work has been updated frequently and is traceable.



TESTING



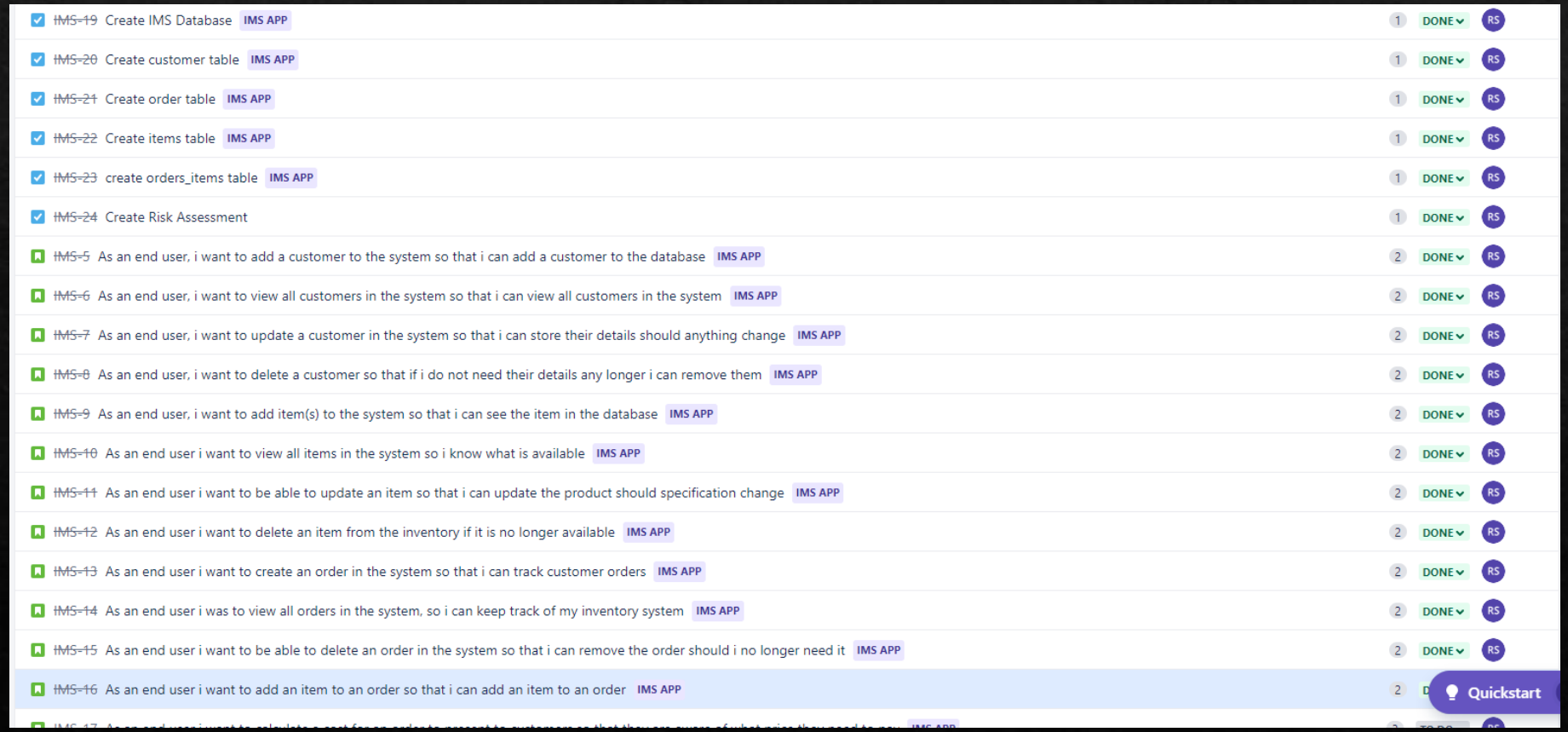
- Haven't completed testing
- Currently at 49% coverage
- Controller classes were tested for CRUD functionality
- DAO classes still need to be tested



User Stories:

- IMS-9 As an end user, i want to add item(s) to the system so that i can see the item in the database IMS APP
- IMS-10 As an end user i want to view all items in the system so i know what is available IMS APP
- IMS-11 As an end user i want to be able to update an item so that i can update the product should specification change IMS APP
- IMS-12 As an end user i want to delete an item from the inventory if it is no longer available IMS APP

Retrospective



- Using MoSCoW principles I prioritised the basic functionality of the app to make sure it could be used and understood
- All user stories except calculating a cost were completed

Improvements

- Adding a method to calculate the cost of an order

```java
package com.qa.ims.persistence.domain;

import java.util.Objects;

public class Items {
private Long id;
private String title;
private double price_decimal;
private int stock;


public Items(Long id, String title, double price_decimal, int stock) {
    super();
    this.id = id;
    this.title = title;
    this.price_decimal = price_decimal;
    this.stock = stock;
}


public Items(String title, double price_decimal, int stock) {
    super();
    this.title = title;
    this.price_decimal = price_decimal;
    this.stock = stock;
}


public Long getId() {
    return id;
}


public void setId(Long id) {
    this.id = id;
```

```java
package com.qa.ims.persistence.dao;

import java.sql.Connection;

public class ItemsDAO implements Dao<Items> {

    public static final Logger LOGGER = LogManager.getLogger();


    @Override
    public Items modelFromResultSet(ResultSet resultSet) throws SQLException {
        Long id = resultSet.getLong("item_id");
        String title = resultSet.getString("title");
        double price_decimal = resultSet.getDouble("price_decimal");
        int stock = resultSet.getInt("stock");
        return new Items (id, title, price_decimal, stock);
    }


    // reading all items and returns as a list
@Override
    public List<Items> readAll() {
        try(Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM items");){
            List<Items> items = new ArrayList<Items>();
            while (resultSet.next()) {
                items.add(modelFromResultSet(resultSet));
            }
            return items;
        }
        catch(SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<Items>();
    }
```

```java
    public Items readLatest() {
        try(Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM items");){
            resultSet.next();
            return modelFromResultSet(resultSet);
            } catch (Exception e) {
                LOGGER.debug(e);
                LOGGER.error(e.getMessage());
            }
        return null;
    }
// takes in item object, id is ignored
    @Override
    public Items create(Items items) {
        try(Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("INSERT INTO items(title, price_decimal, stock) VALUES(?, ?,?)");){
            statement.setString(1, items.getTitle());
            statement.setDouble(2, items.getPrice_decimal());
            statement.setInt(3, items.getStock());
            statement.executeUpdate();
            return readLatest();
            } catch(Exception e) {
                LOGGER.debug(e);
                LOGGER.error(e.getMessage());
            }
        return null;
    }
// read items by id
    @Override
    public Items read(Long id) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("SELECT * FROM items WHERE item_id = ?" );){
            statement.setLong(1, id);
            try (ResultSet resultSet = statement.executeQuery();) {
                resultSet.next();
                return modelFromResultSet(resultSet);
            }
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return null;
```

```java
        }


    @Override
    public Items update(Items items) {
        try (Connection connection = DBUtils.getInstance().getConnection();
    PreparedStatement statement = connection
            .prepareStatement("UPDATE items SET title = ?, price_decimal = ?, stock = ? WHERE item_id =?");){
            statement.setString(1, items.getTitle());
            statement.setDouble(2, items.getPrice_decimal());
            statement.setInt(3, items.getStock());
            statement.setLong(4, items.getId());
            statement.executeUpdate();
            return read(items.getId());
            }catch (Exception e) {
                LOGGER.debug(e);
                LOGGER.error(e.getMessage());
            }
        return null;
    }
// delete item by id
    @Override
    public int delete(long id) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection.prepareStatement("DELETE FROM items WHERE item_id = ?");) {
            statement.setLong(1, id);
            return statement.executeUpdate();
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return 0;
    }
```

```
    public static final Logger LOGGER = LogManager.getLogger();

    @Override
    public Orders modelFromResultSet(ResultSet resultSet) throws SQLException {
        Long id = resultSet.getLong("order_id");
        Long fk_customer_id = resultSet.getLong("fk_customer_id");

        return new Orders(id, fk_customer_id);
}
@Override
    public List<Orders> readAll() {
        try( Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM orders");){
            List<Orders> orders = new ArrayList<Orders>();
            while(resultSet.next()) {
                orders.add(modelFromResultSet(resultSet));
                }
            return orders;
        } catch (SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<Orders>();

    }

public Orders readLatest() {
    try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM orders");){
        resultSet.next();
        return modelFromResultSet(resultSet);
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}
// id is ignored
@Override
public Orders create(Orders orders) {
    try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("INSERT INTO ORDERS (fk_customer_id) VALUES(?)");){
        statement.setLong(1, orders.getFk_customer_id());
```

```
public Orders create(Orders orders) {
    try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("INSERT INTO ORDERS (fk_customer_id) VALUES(?)");){
        statement.setLong(1, orders.getFk_customer_id());
        return readLatest();
    }catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}
@Override
public Orders read(Long id) {
    try( Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("SELECT * FROM orders WHERE order_id = ?" );){
        statement.setLong(1, id);
        try( ResultSet resultSet = statement.executeQuery();) {
                resultSet.next();
                return modelFromResultSet(resultSet);
        }
} catch (Exception e) {
    LOGGER.debug(e);
    LOGGER.error(e.getMessage());
}
    return null;
    }
// takes in a order object, the id field will be used to
// update that order in the database
@Override
public Orders update(Orders orders) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("UPDATE orders fk_customer_id = ? WHERE order_id = ?");){
                    statement.setLong(1, orders.getFk_customer_id());
                    statement.setLong(2, orders.getFk_customer_id());
                    statement.executeUpdate();
                    return read(orders.getId());
                    }catch (Exception e) {
                        LOGGER.debug(e);
                        LOGGER.error(e.getMessage());
                    }
                return null;
            }

```

```
            }

@Override
public int delete(long id) {
    try(Connection connection = DBUtils.getInstance().getConnection();
    PreparedStatement statement = connection.prepareStatement("DELETE FROM orders WHERE order_id = ?");) {
statement.setLong(1, id);
return statement.executeUpdate();
} catch (Exception e) {
LOGGER.debug(e);
LOGGER.error(e.getMessage());
}
return 0;
}


}
```

```java
@Override
public List<Items> readAll() {
List<Items> items = itemsDAO.readAll();
for(Items item : items) {
    LOGGER.info(item);
}
return items;
}
// creates an item by taking in user input
@Override
public Items create() {
    LOGGER.info("Please enter a Title" );
    String title  = utils.getString();
    LOGGER.info("Please enter the price of " + title);
    double price_decimal = utils.getDouble();
    LOGGER.info("Please enter the stock quantity");
    int stock = utils.getInt();
    Items item = itemsDAO.create(new Items(title, price_decimal, stock));
    LOGGER.info("Item Created");
    return item;

}
// updates an item by taking in user input
@Override
public Items update() {
    LOGGER.info(" Please enter the id of the item you would like to update");
    Long id = utils.getLong();
    LOGGER.info("Please enter a title of the item");
    String title = utils.getString();
    LOGGER.info("Please enter the price of " + title);
    double price_decimal = utils.getDouble();
    LOGGER.info("What is the stock quanity?");
    int stock = utils.getInt();
    Items item = itemsDAO.update(new Items(id, title, price_decimal, stock));
    LOGGER.info("Item Updated");
    return item;
}

@Override
public int delete() {
    LOGGER.info("Please enter the id of the item you would like to remove");
    Long id = utils.getLong();
    LOGGER.info("Item Deleted!");
    return itemsDAO.delete(id);
}
```

```java
    public static final Logger LOGGER = LogManager.getLogger();

    private OrdersDAO ordersDAO;
    private Utils utils;


    public OrdersController(OrdersDAO ordersDAO, Utils utils) {
        super();
        this.ordersDAO = ordersDAO;
        this.utils = utils;
    }
// reads all orders to the logger
    @Override
    public List<Orders> readAll() {
        List<Orders> orders = ordersDAO.readAll();
        for (Orders order : orders) {
            LOGGER.info(order);
        }
        return orders;
    }

    @Override
    public Orders create() {
    LOGGER.info("Please enter customer id");
    Long fk_customer_id = utils.getLong();
    Orders orders = ordersDAO.create(new Orders(fk_customer_id));
    LOGGER.info("Order Created!");

    return orders;
    }

    @Override
    public Orders update() {
        LOGGER.info("Please enter the id for the order you would like to update");
        Long id = utils.getLong();
        LOGGER.info("Please enter the customer id");
        Long fk_customer_id = utils.getLong();
        Orders orders = ordersDAO.update(new Orders(id, fk_customer_id));
        return orders;
    }

    @Override
    public int delete() {
            LOGGER.info("Please enter the id of the customer you would like to delete");
            Long id = utils.getLong();
            LOGGER.info("Order Deleted");
```

```java
public class OrdersItems {
private Long id;
private Long fk_order_id;
private Long fk_item_id;
private int quantity;


public OrdersItems(Long id, Long fk_order_id, Long fk_item_id, int quantity) {
    super();
    this.id = id;
    this.fk_order_id = fk_order_id;
    this.fk_item_id = fk_item_id;
    this.quantity = quantity;

}
public OrdersItems(Long fk_order_id, Long fk_item_id, int quantity) {
    super();
    this.fk_order_id = fk_order_id;
    this.fk_item_id = fk_item_id;
    this.quantity = quantity;

}
public Long getId() {
    return id;
}
public void setId(Long id) {
    this.id = id;
}
public Long getFk_order_id() {
    return fk_order_id;
}
public void setFk_order_id(Long fk_order_id) {
    this.fk_order_id = fk_order_id;
}
public Long getFk_item_id() {
    return fk_item_id;
}
public void setFk_item_id(Long fk_item_id) {
    this.fk_item_id = fk_item_id;
}
public int getQuantity() {
    return quantity;
```

```java
public OrdersItemsController(OrdersItemsDAO ordersitemsDAO, Utils utils) {
    super();
    this.ordersitemsDAO = ordersitemsDAO;
    this.utils = utils;
}

@Override
public List<OrdersItems> readAll() {
    List<OrdersItems> ordersitems = ordersitemsDAO.readAll();
    for (OrdersItems ordersitem : ordersitems) {
        LOGGER.info(ordersitem);
    }
    return ordersitems;
}

@Override
public OrdersItems create() {
    LOGGER.info("Please enter the order ID" );
    Long fk_order_id = utils.getLong();
    LOGGER.info("Please enter the item ID");
    Long fk_item_id = utils.getLong();
    LOGGER.info("Please enter the quantity");
    int quantity = utils.getInt();
    OrdersItems ordersitems = ordersitemsDAO.create(new OrdersItems(fk_order_id, fk_item_id, quantity));
    LOGGER.info("Order-Item Created");
    return ordersitems;
}

@Override
public OrdersItems update() {

    LOGGER.info(" Please enter the id of the item you would like to update");
    Long id = utils.getLong();
    LOGGER.info("Please enter the corresponding order ID" );
    Long fk_order_id = utils.getLong();
    LOGGER.info("Please enter the corresponding item ID");
    Long fk_item_id = utils.getLong();
    LOGGER.info("Please enter the quantity");
    int quantity = utils.getInt();
    OrdersItems ordersitems = ordersitemsDAO.update(new OrdersItems(fk_order_id, fk_item_id, quantity));
    LOGGER.info("Order-Item Updated");
    return ordersitems;
```

```java
public class OrdersItemsDAO implements Dao<OrdersItems>{

    public static final Logger LOGGER = LogManager.getLogger();

    @Override
    public OrdersItems modelFromResultSet(ResultSet resultSet) throws SQLException {
        Long id = resultSet.getLong("order_item_id");
        Long fk_order_id = resultSet.getLong("fk_order_id");
        Long fk_item_id = resultSet.getLong("fk_item_id");
        int quantity = resultSet.getInt("quantity");

        return new OrdersItems(id, fk_order_id, fk_item_id, quantity);
    }

@Override
    public List<OrdersItems> readAll() {
        try( Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM orders_items");){
            List<OrdersItems> ordersItems = new ArrayList<OrdersItems>();
            while(resultSet.next()) {
                ordersItems.add(modelFromResultSet(resultSet));
                }
            return ordersItems;
        } catch (SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<OrdersItems>();
    }

public OrdersItems readLatest() {
    try(Connection connection = DBUtils.getInstance().getConnection();
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM orders_items ORDER BY order_item_id DESC LIMIT 1"); ){
        resultSet.next();
        return modelFromResultSet(resultSet);
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
    return null;
```

```java
// id ignored as it is auto increment
@Override
    public OrdersItems create(OrdersItems ordersItems) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("INSERT INTO orders_items(fk_order_id, fk_item_id, quantity) VALUES (?, ?, ?)");) {
        statement.setLong(1, ordersItems.getFk_order_id());
        statement.setLong(2, ordersItems.getFk_item_id());
        statement.setInt(3, ordersItems.getQuantity());
        statement.executeUpdate();
        return readLatest();
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
    }
// read by id
@Override
    public OrdersItems read(Long id) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection.prepareStatement("SELECT * FROM orders_items WHERE order_item_id = ?");) {
        statement.setLong(1, id);
        try (ResultSet resultSet = statement.executeQuery();) {
            resultSet.next();
            return modelFromResultSet(resultSet);
        }
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}


    @Override
    public OrdersItems update(OrdersItems ordersItems) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("UPDATE orders_items SET fk_order_id = ?, fk_item_id = ?, quantity = ? WHERE order_item_id = ?");) {
            statement.setLong(1, ordersItems.getFk_order_id());
            statement.setLong(2, ordersItems.getFk_item_id());
            statement.setInt(3, ordersItems.getQuantity());
            statement.executeUpdate();
            return read(ordersItems.getId());
```

```java
@Override
public OrdersItems update(OrdersItems ordersItems) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("UPDATE orders_items SET fk_order_id = ?, fk_item_id = ?, quantity = ? WHERE order_item_id = ?");) {
        statement.setLong(1, ordersItems.getFk_order_id());
        statement.setLong(2, ordersItems.getFk_item_id());
        statement.setInt(3, ordersItems.getQuantity());
        statement.executeUpdate();
        return read(ordersItems.getId());
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}

@Override
public int delete(long id) {
    try(Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection.prepareStatement("DELETE FROM orders_items WHERE order_item_id = ?");) {
        statement.setLong(1, id);
        return statement.executeUpdate();
        } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
        }
    return 0;
}
```

```java
    @Override
    public List<OrdersItems> readAll() {
        List<OrdersItems> ordersitems = ordersitemsDAO.readAll();
        for (OrdersItems ordersitem : ordersitems) {
            LOGGER.info(ordersitem);
        }
        return ordersitems;
    }

    @Override
    public OrdersItems create() {
        LOGGER.info("Please enter the order ID" );
        Long fk_order_id = utils.getLong();
        LOGGER.info("Please enter the item ID");
        Long fk_item_id = utils.getLong();
        LOGGER.info("Please enter the quantity");
        int quantity = utils.getInt();
        OrdersItems ordersitems = ordersitemsDAO.create(new OrdersItems(fk_order_id, fk_item_id, quantity));
        LOGGER.info("Order-Item Created");
        return ordersitems;
    }

    @Override
    public OrdersItems update() {

        LOGGER.info(" Please enter the id of the item you would like to update");
        Long id = utils.getLong();
        LOGGER.info("Please enter the corresponding order ID" );
        Long fk_order_id = utils.getLong();
        LOGGER.info("Please enter the corresponding item ID");
        Long fk_item_id = utils.getLong();
        LOGGER.info("Please enter the quantity");
        int quantity = utils.getInt();
        OrdersItems ordersitems = ordersitemsDAO.update(new OrdersItems(fk_order_id, fk_item_id, quantity));
        LOGGER.info("Order-Item Updated");
        return ordersitems;
}
    @Override
    public int delete() {
        LOGGER.info("Please enter the id of the order-item you would like to remove");
        Long id = utils.getLong();
        LOGGER.info("Order-Item Deleted!");
        return ordersitemsDAO.delete(id);
```

```
Welcome to the Inventory Management System!
Which entity would you like to use?
CUSTOMER: Information about customers
ITEM: Individual Items
ORDER: Purchases of items
ORDER_ITEM: Adding an item(s) to an order
STOP: To close the application
item
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
create
Please enter a Title
Hairbrush
Please enter the price of Hairbrush
1.99
Please enter the stock quantity
8
Item Created
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
read
Items: [id: 1, title: hairbrush, price: 1.09, stock: 10
```

```
RETURN: To return to domain selection
read
Items: [id: 1, title: hairbrush, price: 1.09, stock: 10
Items: [id: 2, title: Hairbrush, price: 1.99, stock: 8
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
update
 Please enter the id of the item you would like to update
1
Please enter a title of the item
Water Bottle
Please enter the price of Water Bottle
.99
What is the stock quanity?
12
Item Updated
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
read
Items: [id: 1, title: Water Bottle, price: 0.99, stock: 12
Items: [id: 2, title: Hairbrush, price: 1.99, stock: 8
```

```
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
delete
Please enter the id of the item you would like to remove
2
Item Deleted!
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
read
Items: [id: 1, title: Water Bottle, price: 0.99, stock: 12
What would you like to do with item:
CREATE: To save a new entity into the database
READ: To read an entity from the database
UPDATE: To change an entity already in the database
DELETE: To remove an entity from the database
RETURN: To return to domain selection
```

```java
package com.qa.ims.persistence.domain;

import java.util.Objects;


public class Orders {
private Long id;
private Long fk_customer_id;


public Orders(Long id, Long fk_customer_id) {
    super();
    this.id = id;
    this.fk_customer_id = fk_customer_id;

}


public Orders(Long fk_customer_id) {
    super();
    this.fk_customer_id = fk_customer_id;

}


public Long getId() {
    return id;
}


public void setId(Long id) {
    this.id = id;
}


public Long getFk_customer_id() {
    return fk_customer_id;
}


public void setFk_customer_id(Long fk_customer_id) {
    this.fk_customer_id = fk_customer_id;
}
```

```java
public Orders create(Orders orders) {
    try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("INSERT INTO ORDERS (fk_customer_id) VALUES(?)");){
        statement.setLong(1, orders.getFk_customer_id());
        return readLatest();
    }catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}
@Override
public Orders read(Long id) {
    try( Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("SELECT * FROM orders WHERE order_id = ?" );){
        statement.setLong(1, id);
        try( ResultSet resultSet = statement.executeQuery();) {
                resultSet.next();
                return modelFromResultSet(resultSet);
        }
} catch (Exception e) {
    LOGGER.debug(e);
    LOGGER.error(e.getMessage());
}
    return null;
    }
// takes in a order object, the id field will be used to
// update that order in the database
@Override
public Orders update(Orders orders) {
    try (Connection connection = DBUtils.getInstance().getConnection();
            PreparedStatement statement = connection
                    .prepareStatement("UPDATE orders fk_customer_id = ? WHERE order_id = ?");){
                    statement.setLong(1, orders.getFk_customer_id());
                    statement.setLong(2, orders.getFk_customer_id());
                    statement.executeUpdate();
                    return read(orders.getId());
                    }catch (Exception e) {
                        LOGGER.debug(e);
                        LOGGER.error(e.getMessage());
                    }
                return null;
            }
```

```java
package com.qa.ims.persistence.dao;

import java.sql.Connection;

public class OrdersDAO implements Dao<Orders> {

    public static final Logger LOGGER = LogManager.getLogger();

    @Override
    public Orders modelFromResultSet(ResultSet resultSet) throws SQLException {
        Long id = resultSet.getLong("order_id");
        Long fk_customer_id = resultSet.getLong("fk_customer_id");

        return new Orders(id, fk_customer_id);
}
@Override
    public List<Orders> readAll() {
        try( Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM orders");){
            List<Orders> orders = new ArrayList<Orders>();
            while(resultSet.next()) {
                orders.add(modelFromResultSet(resultSet));
                }
            return orders;
        } catch (SQLException e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return new ArrayList<Orders>();

    }

public Orders readLatest() {
    try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM orders");){
        resultSet.next();
        return modelFromResultSet(resultSet);
    } catch (Exception e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return null;
}
```

```java
    private OrdersDAO ordersDAO;
    private Utils utils;


    public OrdersController(OrdersDAO ordersDAO, Utils utils) {
        super();
        this.ordersDAO = ordersDAO;
        this.utils = utils;
    }
// reads all orders to the logger
    @Override
    public List<Orders> readAll() {
        List<Orders> orders = ordersDAO.readAll();
        for (Orders order : orders) {
            LOGGER.info(order);
        }
        return orders;
    }

    @Override
    public Orders create() {
    LOGGER.info("Please enter customer id");
    Long fk_customer_id = utils.getLong();
    Orders orders = ordersDAO.create(new Orders(fk_customer_id));
    LOGGER.info("Order Created!");

    return orders;
    }

    @Override
    public Orders update() {
        LOGGER.info("Please enter the id for the order you would like to update");
        Long id = utils.getLong();
        LOGGER.info("Please enter the customer id");
        Long fk_customer_id = utils.getLong();
        Orders orders = ordersDAO.update(new Orders(id, fk_customer_id));
        return orders;
    }

    @Override
    public int delete() {
            LOGGER.info("Please enter the id of the customer you would like to delete");
            Long id = utils.getLong();
            LOGGER.info("Order Deleted");
            return ordersDAO.delete(id);
    }
```

```java
    public int getInt() {
String input = null;
Integer intInput = null;
do {
    try {
        input = getString();
        intInput = Integer.parseInt(input);
        } catch (NumberFormatException nfe) {
            LOGGER.info("Not an integer, please try again.");
        }
    } while (intInput == null);
return intInput;
    }
```

```java
public class IMS {

    public static final Logger LOGGER = LogManager.getLogger();

    private final CustomerController customers;
    private final ItemsController items;
    private final OrdersController orders;
    private final OrdersItemsController ordersItems;
    private final Utils utils;

    public IMS() {
        this.utils = new Utils();
        final CustomerDAO custDAO = new CustomerDAO();
        this.customers = new CustomerController(custDAO, utils);
        final ItemsDAO itemsDAO = new ItemsDAO();
        this.items = new ItemsController(itemsDAO, utils);
        final OrdersDAO ordersDAO = new OrdersDAO();
        this.orders = new OrdersController(ordersDAO, utils);
        final OrdersItemsDAO orderItemsDAO = new OrdersItemsDAO();
        this.ordersItems = new OrdersItemsController(orderItemsDAO, utils);
    }
```

```java
        CrudController<?> active = null;
        switch (domain) {
        case CUSTOMER:
            active = this.customers;
            break;
        case ITEM:
            active = this.items;
            break;
        case ORDER:
            active = this.orders;
            break;
        case ORDER_ITEM:
            active = this.ordersItems;
            break;
            case STOP:
            return;
        default:
            break;
        }
```