JAVA SE PROGRAMMING LANGUAGE

Lab Guides

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, 06/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
»	01/Oct/2018	Create new	Draft		
»	01/Jun/2019	Update template	Fsoft template	DieuNT1	VinhNV

Contents

CODE:	JPL.L.L201
TYPE:	LONG
LOC:	350
DURATION:	90 MINUTES

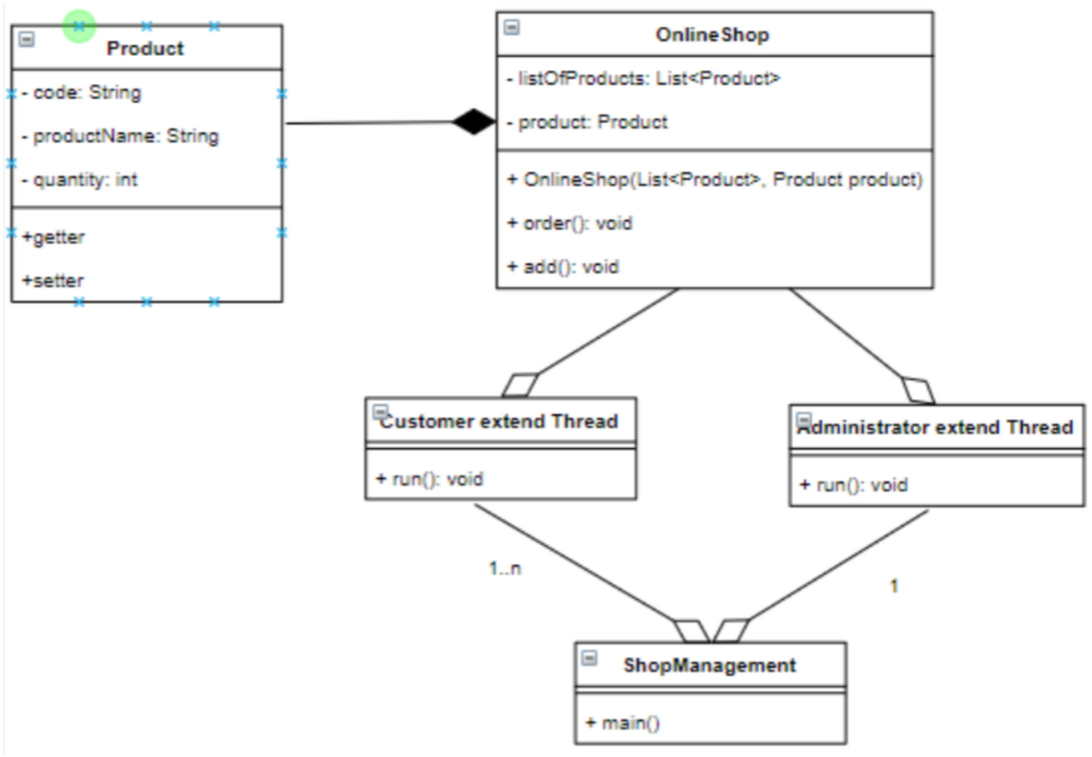
Unit 14: Concurrency

Objectives:

» Understand how to use create threads and manipulate threads with its API.

Lab Specifications:

For the hierarchy below, the trainee will create java classes that will implement this class diagram. Your classes should be able to show relationship between the entities.



Create a class called **Product** with the following information:

» Three private instance variables: code(String), productName(String), quantity(int).

» One constructor to initialize the code, product name, quantity with the given values. Also include **getter** and **setter** method, overiding **equal()** method.

And, a class called **OnlineShop** provides 2 functions: *order* and *add* method.

» *Order()* method allows the customer can order a specific product.

» *Add()* method allows the system admin can add a specific product to shop.

» Finally, two classes named **Customer** and **Administrator** inherited Thread class, overriding run() method. These classes will provide instances of customer and system admin.

Business Rules

» While the customer purchases a product, the product is locked (synchronized).

» If the product does not exist, then wait().

» Admin adds a product and then notifyAll().

Functional Requirements

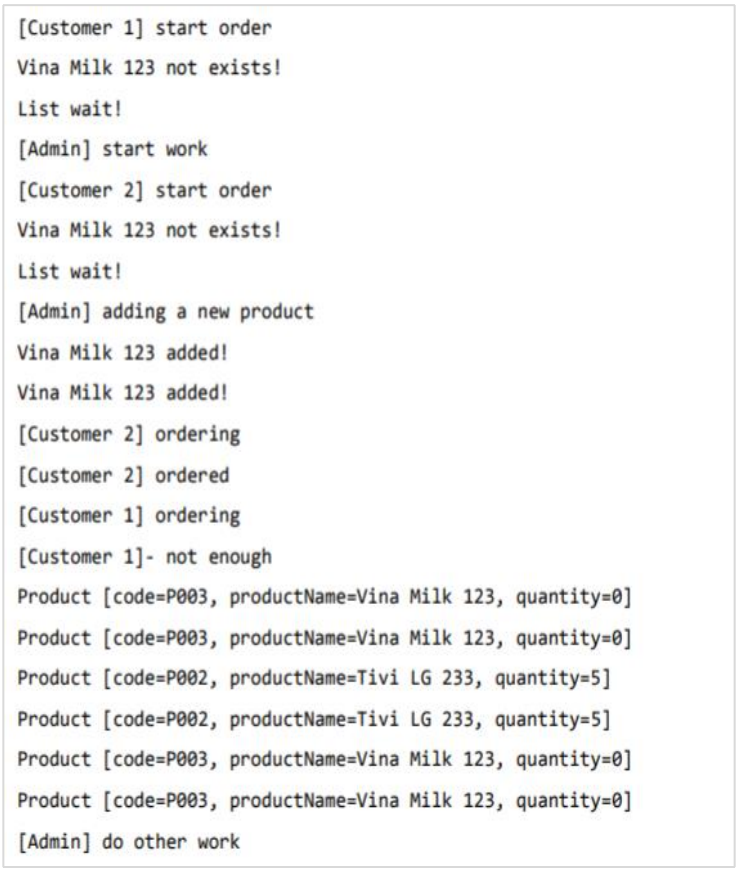
Let's create a class named **ShopManagement** contains main() method to simulate an online shop.

a. Create some instances of **Customer** that make the order a specific **Product** using order() method.

b. An instance of **Administrator** makes the add a specific **Product** to **Shop** using add() method.

c. In this app, users (Customer and Admin) is concurrently using the OnlineShop so that you should apply multi-threadings to solve this problem.

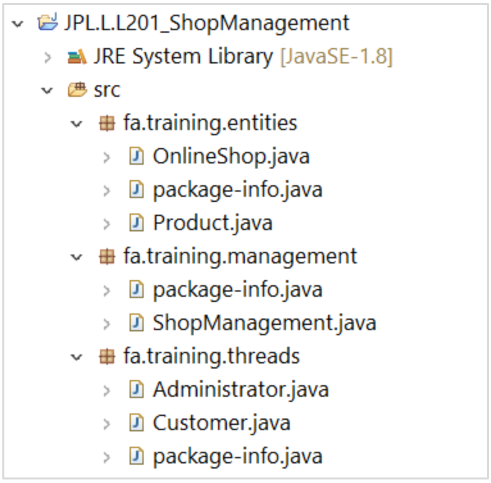
Screen Design



```
[Customer 1] start order
Vina Milk 123 not exists!
List wait!
[Admin] start work
[Customer 2] start order
Vina Milk 123 not exists!
List wait!
[Admin] adding a new product
Vina Milk 123 added!
Vina Milk 123 added!
[Customer 2] ordering
[Customer 2] ordered
[Customer 1] ordering
[Customer 1]- not enough
Product [code=P003, productName=Vina Milk 123, quantity=0]
Product [code=P003, productName=Vina Milk 123, quantity=0]
Product [code=P002, productName=Tivi LG 233, quantity=5]
Product [code=P002, productName=Tivi LG 233, quantity=5]
Product [code=P003, productName=Vina Milk 123, quantity=0]
Product [code=P003, productName=Vina Milk 123, quantity=0]
[Admin] do other work
```

Guidelines:

» Step1. Create a new project named **JPL.L.L201_ShopManagement**.



```
JPL.L.L201_ShopManagement
  JRE System Library [JavaSE-1.8]
  src
    fa.training.entities
      OnlineShop.java
      package-info.java
      Product.java
    fa.training.management
      package-info.java
      ShopManagement.java
    fa.training.threads
      Administrator.java
      Customer.java
      package-info.java
```

» Step2: Create package *fa.training.entities* that contains classes named **Product** and **OnlineShop** class as follows:

Product class:

```java
package fa.training.entities;
public class Product {
private String code;
private String productName;
private int quantity;

public Product(String code, String productName, int quantity) {
        super();
        this.code = code;
        this.productName = productName;
        this.quantity = quantity;
}
public Product() {
        super();
}

public String getCode() {
        return code;
}
public void setCode(String code) {
        this.code = code;
}

public String getProductName() {
        return productName;
}

public void setProductName(String productName) {
        this.productName = productName;
}

public int getQuantity() {
        return quantity;
}

public void setQuantity(int quantity) {
        this.quantity = quantity;
}

@Override
public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((code == null) ? 0 : code.hashCode());
        result = prime * result
                        + ((productName == null) ? 0 : productName.hashCode());
        result = prime * result + quantity;
        return result;
}

@Override
public boolean equals(Object obj) {
        if (this == obj)
                return true;
        if (obj == null)
                return false;
        if (getClass() != obj.getClass())
                return false;
        Product other = (Product) obj;
        if (code == null) {
                if (other.code != null)
                        return false;
        } else if (!code.equals(other.code))
                return false;
        if (productName == null) {
                if (other.productName != null)
                        return false;
        } else if (!productName.equals(other.productName))
                return false;
        if (quantity != other.quantity)
                return false;
        return true;
}

@Override
public String toString() {
        return "Product [code=" + code + ", productName=" + productName +
                 ", quantity=" + quantity + "]";
}

}
```

OnlineShop class:

```java
package fa.training.entities;
import java.util.List;
public class OnlineShop {
private List<Product> listOfProducts;
private Product product;

public OnlineShop(List<Product> listOfProducts, Product product) {
        super();
        this.listOfProducts = listOfProducts;
        this.product = product;
}

public void order() {
        System.out.println(Thread.currentThread().getName() + " start order");
        synchronized (listOfProducts) {
                if (!listOfProducts.contains(product)) {
                try {
                        System.out.println(product.getProductName() + " not exists!");
                        System.out.println("List wait!");
                        listOfProducts.wait();
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                System.out.println(product.getProductName() + " added!");
        }
}

/*
 Customer lock a select product
*/
        synchronized (product) {
                System.out.println(Thread.currentThread().getName() + " ordering");
                int index = 0, amount = 2;
                Product orderProduct = null;
                for (int i = 0; i < listOfProducts.size(); i++) {
                        orderProduct = listOfProducts.get(i);
                        if (orderProduct.equals(product)) {
                                if (orderProduct.getQuantity() >= amount) {
                                        index = i;
                                        orderProduct.setQuantity(orderProduct.getQuantity()
                                                                                        - amount);
                                        System.out.println(Thread.currentThread().getName() +
                                                                                 " ordered");
                                        break;
                                } else {
                                        System.out.println(Thread.currentThread().getName() +
                                                                                 "- not enough");
                                }
                        }
                }
                listOfProducts.set(index, orderProduct);
        }
        for(Product product: listOfProducts) {
                System.out.println(product);
        }
}

/**
Add a new product
*/
public void add() {
        System.out.println("[Admin] start work");
        synchronized (listOfProducts) {
                System.out.println("[Admin] adding a new product");
                listOfProducts.add(product);
                listOfProducts.notifyAll();
        }
        System.out.println("[Admin] do other work");
}

}
```

» Step3: Create package *fa.training.threads* that contains classes named **Customer** and **Administrator** class as follows:

Customer class:

```
package fa.training.threads;
import java.util.List;
import fa.training.entities.OnlineShop;
import fa.training.entities.Product;
public class Customer extends Thread {

private List<Product> listOfProducts;
private Product product;

public Customer(List<Product> listOfProducts, Product product) {
        super();
        this.listOfProducts = listOfProducts;
        this.product = product;
}




@Override
```

```java
public void run() {
        OnlineShop onlineShop= new OnlineShop(listOfProducts, product);
        onlineShop.order();
}

}
```

Administrator class:

```java
package fa.training.threads;
import java.util.List;
import fa.training.entities.OnlineShop;
import fa.training.entities.Product;

public class Administrator extends Thread {
        private List<Product> listOfProducts;
        private Product product;

public Administrator(List<Product> listOfProducts, Product product) {
        super();
        this.listOfProducts = listOfProducts;
        this.product = product;
}

@Override
public void run() {
        OnlineShop onlineShop= new OnlineShop(listOfProducts, product);
        onlineShop.add();
}
}
```

» Step4: Create package fa.training.management that contains a class named **ShopManagement** as follows:

ShopManagement class:

```java
package fa.training.management;
import java.util.ArrayList;
import java.util.List;
import fa.training.entities.Product;
import fa.training.threads.Administrator;
import fa.training.threads.Customer;

public class ShopManagement {

public static void main(String[] args) {
        List<Product> listOfProducts = new ArrayList<>();
        Product p1 = new Product("P001", "Laptop Dell 123", 3);
        Product p2 = new Product("P002", "Tivi LG 233", 5);
        Product p3 = new Product("P003", "Vina Milk 123", 2);

        listOfProducts.add(p1);
        listOfProducts.add(p2);

        Customer customer1 = new Customer(listOfProducts, p3);
        customer1.setName("[Customer 1]");
        Customer customer2 = new Customer(listOfProducts, p3);
        customer2.setName("[Customer 2]");

        Administrator administrator = new Administrator(listOfProducts, p3);

        customer1.setPriority(Thread.MAX_PRIORITY);
        customer2.setPriority(Thread.MAX_PRIORITY);
        administrator.setPriority(Thread.MIN_PRIORITY);

        customer1.start();
        customer2.start();
        administrator.start();
}

}
```

----oOo-----

THE END