Compte Rendu RMI

Lu Chenxin, Xu Jitao, Zhu Zhenghui

Cas d'utilisation:

Diagramme de sequence:

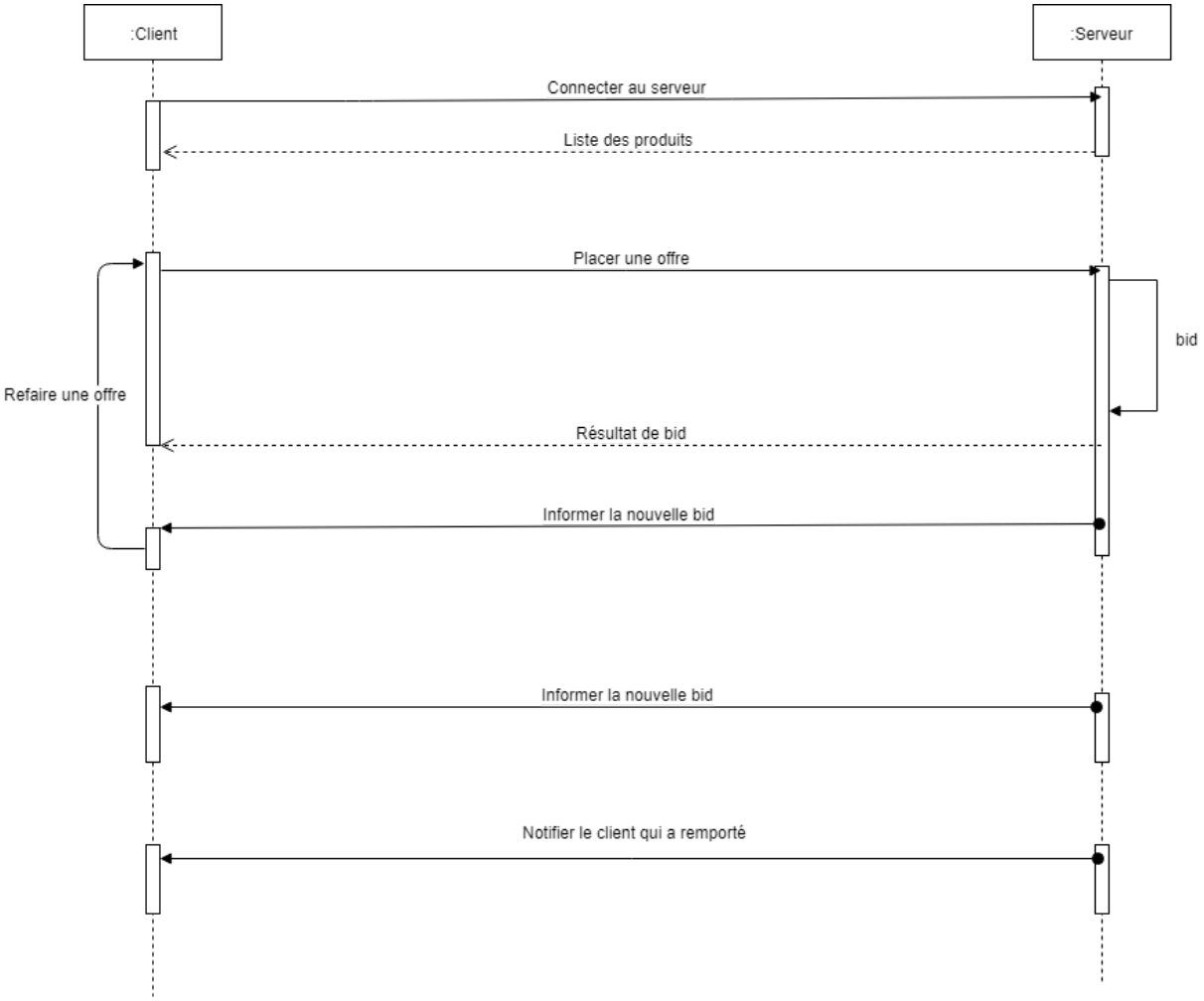
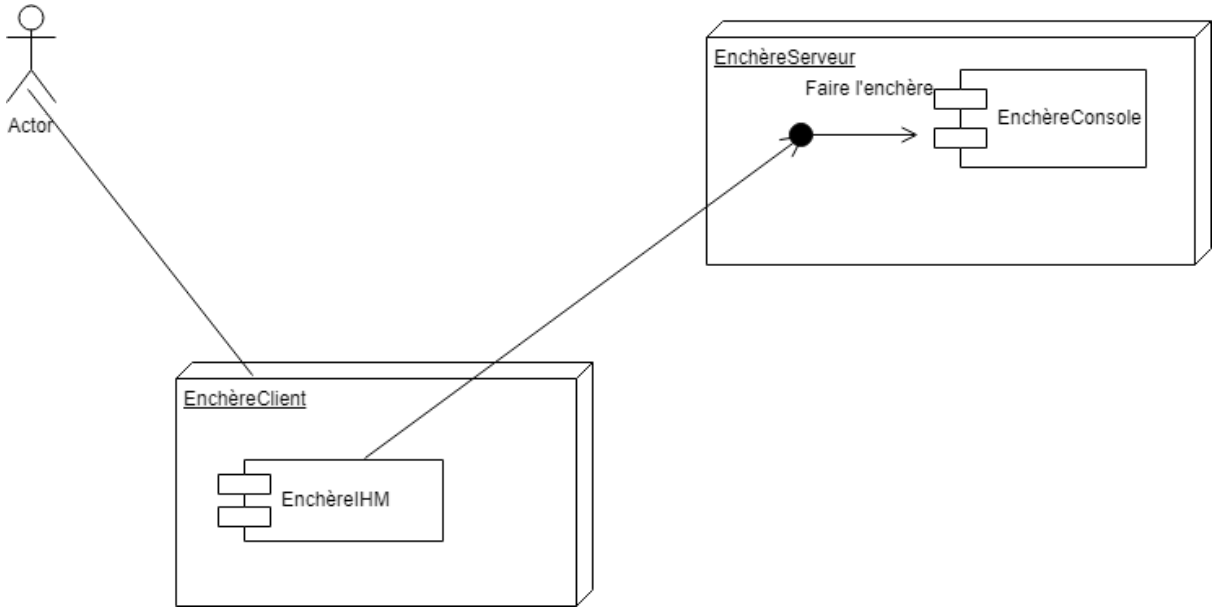


Diagramme de déploiement:



Code Serveur:

Serveur.java — src

```java
import java.rmi.registry.*;
import java.rmi.server.UnicastRemoteObject;
import java.util.*;

public class Serveur {
    public static void main(String args[]) {
        int port = 1099;
        if(args.length == 1)
            port = Integer.parseInt(args[0]);
        try {
            ArrayList<Product> productList = new ArrayList<>();
            Product stubIphone = (Product)UnicastRemoteObject.exportObject(new ProductImpl("IphoneX", 500, 900), 0);
            Product stubPixel2 = (Product)UnicastRemoteObject.exportObject(new ProductImpl("Pixel2", 400, 700), 0);
            productList.add(stubIphone);
            productList.add(stubPixel2);
            Registry registry = LocateRegistry.getRegistry(port);
            if(!Arrays.asList(registry.list()).contains("IphoneXCallback")) {
                registry.bind("IphoneXCallback", stubIphone);
            }
            else {
                registry.rebind("IphoneXCallback", stubIphone);
            }
            if(!Arrays.asList(registry.list()).contains("Pixel2Callback")) {
                registry.bind("Pixel2Callback", stubPixel2);
            }
            else {
                registry.rebind("Pixel2Callback", stubPixel2);
            }
            System.out.println("Service IphoneXCallback et Pixel2Callback lient au registre.");

            ArrayList<Notification> notificationList = new ArrayList<>();

            String[] callbackNameList = registry.list();
            int nbCallback = callbackNameList.length;
            for (int i = 0; i < callbackNameList.length; i++) {
                if (callbackNameList[i].startsWith("NotificationCallback")) {
                    notificationList.add((Notification)registry.lookup(callbackNameList[i]));
                }
            }

            for (Notification n : notificationList) {
                n.setProductList(productList);
                n.productList();
            }

            ArrayList<Integer> oldPrices = new ArrayList<>();
```

```
            ArrayList<Integer> oldPrices = new ArrayList<>();
            for(int i = 0; i < productList.size(); i++) {
                oldPrices.add(productList.get(i).getPrice());
            }
            while (true) {
                callbackNameList = registry.list();
                if (nbCallback < callbackNameList.length) {
                    System.out.println("New product added.");
                    int prodSize = productList.size();
                    int registrySize = callbackNameList.length;
                    int nb = registrySize - prodSize - 1;
                    if (callbackNameList[nb].startsWith("NotificationCallback")) {
                        notificationList.add((Notification)registry.lookup(callbackNameList[nb]));
                    }
                    nbCallback = callbackNameList.length;
                    notificationList.get(nb).setProductList(productList);
                    notificationList.get(nb).productList();
                }
            }
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}

```

Code Client:

Client.java

```java
import java.rmi.registry.*;
import java.rmi.server.UnicastRemoteObject;
import java.util.*;

public class Client {
    public static void main(String args[]) {
        String machine = "localhost";
        int port = 1099;
        Bidder stubBidder, bidder = new BidderImpl();
        Notification stubNotification, notification = new NotificationImpl();
        Scanner sc;
        if(args.length == 3) {
            machine = args[0];
            port = Integer.parseInt(args[1]);
        }
        else if(args.length == 2) {
            machine = args[0];
        }
        try {
            Registry registry = LocateRegistry.getRegistry(port);
            stubBidder = (Bidder)UnicastRemoteObject.exportObject(bidder, 0);
            stubNotification = (Notification)UnicastRemoteObject.exportObject(notification, 0);
            String notificationCallbackName = "NotificationCallback";
            int nbNotif = 1;
            while (true) {
                String temp = notificationCallbackName + nbNotif;
                if(!Arrays.asList(registry.list()).contains(temp)) {
                    registry.bind(temp, stubNotification);
                    break;
                }
                else {
                    nbNotif++;
                }
                System.out.println(temp);
            }
            System.out.println("Service NotificationCallback lie au registre.");

            while (true) {
                System.out.println("Saisir le nom de produit et le prix d'enchère, SVP. Exit pour ne pas faire enchère.");
                sc = new Scanner(System.in);
                String bid = sc.nextLine();
                if (bid.equalsIgnoreCase("exit")) {
                    UnicastRemoteObject.unexportObject(notification, true);
                    break;
                }
                String[] parts = bid.split(" ");
                String productName = parts[0];
                int bidPrice = Integer.parseInt(parts[1]);
                ArrayList<Product> products = notification.getProductList();
                for (Product p : products) {
                    if (p.getName().equalsIgnoreCase(productName)) {
                        p.bid(bidPrice, stubNotification);
                        break;
                    }
                }
            }
        }
        catch (Exception e) {
            System.out.println("Client exception: " + e);
        }
    }
}

```