CSC 750 Service-Oriented Computing
Name: Chinmay Terse
Unity ID: cterse

The following is the protocol created for one buyer agent:

```
protocol P3Protocol {
	roles Buyer, Merchant, Shipper
	parameters out requestId key, out item, out orderId, out address, out buyerName, out
amount, out paymentId, out cancelId, out shipmentId, out refundId
	private out trackingId

	Buyer -> Merchant: RequestQuote[out requestId, out item, out buyerName]
	Buyer -> Merchant: Order[in requestId, in amount, out orderId, in item, out address, in
buyerName]
	Buyer -> Merchant: Pay[in requestId, in orderId, in amount, out paymentId, in item, in
address, in buyerName]
	Buyer -> Merchant: CancelOrder[in requestId, in orderId, nil shipmentId, out cancelId]

	Merchant -> Buyer: SendQuote[in requestId, in item, out amount, in buyerName]
	Merchant -> Shipper: SendItemToShip[in requestId, in orderId, in item, in address, in
buyerName, in paymentId]
	Merchant -> Buyer: SendShippedNotification[in requestId, in orderId, nil cancelId, in
paymentId]
	Merchant -> Buyer: RefundBuyerOnCancel[in requestId, in cancelId, in orderId, in amount,
out refundId]
	Merchant -> Buyer: SendDeliveryReminder[in requestId, in orderId, in shipmentId]
	Merchant -> Buyer: SendDefectiveNotification[in requestId, in orderId, in shipmentId, in
item]

	Shipper -> Buyer: Ship[in requestId, in orderId, out shipmentId, in item]
	Shipper -> Merchant: ProvideTracking[in requestId, in orderId, in shipmentId, out
trackingId]
	Shipper -> Merchant: ConfirmDelivery[in requestId, in orderId, in shipmentId]
	Shipper -> Merchant: ReportItem[in requestId, in orderId, in shipmentId, in item]
}
```

The following is the same protocol as above, this time modified for two buyer agents, Buyer and BuyerB:

```
protocol P3Protocol {
      roles Buyer, Merchant, Shipper, BuyerB
      parameters out defectiveRefund, out requestId key, out item, out orderId, out address,
out buyerName, out amount, out paymentId, out cancelId, out shipmentId, out refundId
      private out trackingId

      Buyer -> Merchant: RequestQuote[out requestId, out item, out buyerName]
      Buyer -> Merchant: Order[in requestId, in amount, out orderId, in item, out address, in
buyerName]
      Buyer -> Merchant: Pay[in requestId, in orderId, in amount, out paymentId, in item, in
address, in buyerName]
      Buyer -> Merchant: CancelOrder[in requestId, in orderId, nil shipmentId, out cancelId]

      Merchant -> Buyer: SendQuote[in requestId, in item, out amount, in buyerName]
      Merchant -> Shipper: SendItemToShip[in requestId, in orderId, in item, in address, in
buyerName, in paymentId]
      Merchant -> Buyer: SendShippedNotification[in requestId, in orderId, nil cancelId, in
paymentId]
      Merchant -> Buyer: RefundBuyerOnCancel[in requestId, in cancelId, in orderId, in amount,
out refundId]
      Merchant -> Buyer: SendDeliveryReminder[in requestId, in orderId, in shipmentId]
      Merchant -> Buyer: SendDefectiveNotification[in requestId, in orderId, in shipmentId, in
item]
      Merchant -> Buyer: SendDefectiveRefund[in requestId, in orderId, in shipmentId, out
defectiveRefund]

      Shipper -> Merchant: ProvideTracking[in requestId, in orderId, out shipmentId, out
trackingId]
      Shipper -> Merchant: ConfirmDelivery[in requestId, in orderId, in shipmentId]
      Shipper -> Merchant: ReportItem[in requestId, in orderId, in shipmentId, in item]



      BuyerB -> Merchant: RequestQuote[out requestId, out item, out buyerName]
      BuyerB -> Merchant: Order[in requestId, in amount, out orderId, in item, out address, in
buyerName]
      BuyerB -> Merchant: Pay[in requestId, in orderId, in amount, out paymentId, in item, in
address, in buyerName]
      BuyerB -> Merchant: CancelOrder[in requestId, in orderId, nil shipmentId, out cancelId]
      Merchant -> BuyerB: SendQuote[in requestId, in item, out amount, in buyerName]
      Merchant -> BuyerB: SendShippedNotification[in requestId, in orderId, nil cancelId, in
paymentId]
      Merchant -> BuyerB: RefundBuyerOnCancel[in requestId, in cancelId, in orderId, in amount,
out refundId]
```

```
      Merchant -> BuyerB: SendDeliveryReminder[in requestId, in orderId, in shipmentId]
      Merchant -> BuyerB: SendDefectiveNotification[in requestId, in orderId, in shipmentId, in
item]
```

```
      Merchant -> BuyerB: SendDefectiveRefund[in requestId, in orderId, in shipmentId, out
defectiveRefund]
}
```

The above protocol files can be found in the zip folder as p3protocol.json and p3protocol2.json, respectively. Both versions are safe as well as live.

Following are the enactments that can be demonstrated using the code:

1. **Happy Path:** Completion of the entire order process. A quote record is added to the Quotes table, which is then used by the Buyers to initiate an Order and Pay for the Order. The merchant, upon receipt of payment, forwards the order details to the shipper, and the shipper ships the order to the buyer and sends a tracking number to the merchant. Upon successful delivery, it sends a notification to the merchant, who in turn sends a delivery complete reminder to the buyer, hence completing the transaction.
2. **Defective Item found on Shipping:** The initial part is the same as above, the only difference being that the shipper sends a defective item notification instead of completing delivery. The merchant in turn sends another notification to the buyer that the item is defective, and initiates a defective item notification for the buyer.
3. **Buyer B Cancels Order:** Buyer B cancels order after payment. This cancel message can reach before or after the merchant has sent the order to the shipper, and depending on this the cancel request will be chosen to be respected by the merchant. If cancel message arrives before shipment, the order is cancelled and no further steps are taken. Else, the cancel request is ignored.

How to view enactments:

Deploy the system.

Enactment 1: Run the following POST query on the deployed system:

```
curl -X POST -H 'Content-Type: application/json' <DOMAIN>/buyer/quote -d
'{"requestId":2, "item":"test item", "buyerName": "buyerA"}'
```

NOTE: Request ID should be an even number for the delivery to be successful.

Enactment 2: Run the following POST query on the deployed system:

```
curl -X POST -H 'Content-Type: application/json' <DOMAIN>/buyer/quote -d
'{"requestId":1, "item":"test item", "buyerName": "buyerA"}'
```

NOTE: Request ID should be an odd number for the item to be identified as defective.

Enactment 3: Run the following POST query on the deployed system:

```
curl -X POST -H 'Content-Type: application/json' <DOMAIN>/buyer/quote -d
'{"requestId":2, "item":"test item", "buyerName": "buyerB"}'
```

NOTE: Request ID should be an even number and buyerName should be “buyerB” for the buyer to cancel the order.

The logs for the above enactments are present in files named en1.csv, en2.csv and en3.csv, respectively.