TK1 WS 2015/16

2nd Programming Exercise - Web Service

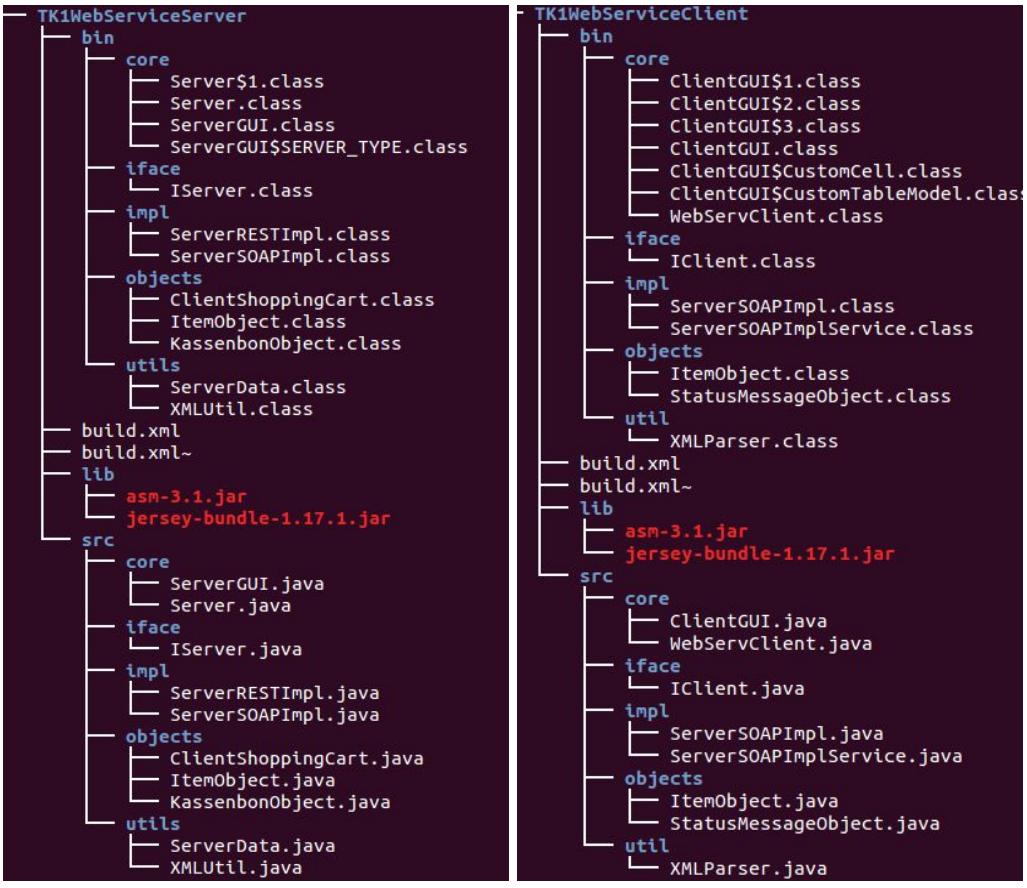
GROUP MEMBERS

1. Jason Christian
2. Dimas Prawita
3. Subhadeep Manna
4. Parvez Ahmad

DEVELOPMENT ENVIRONMENT

IDE	**:**	Eclipse Juno
Java version	**:**	JDK 1.7.0_85
		OpenJDK Runtime Environment (IcedTea 2.6.1)
Libraries	**:**	jersey-bundle-1.17.1.jar
		asm-3.1.jar (dependency for Jersey)

PROJECT DIRECTORY STRUCTURE



Each project has its own ant script (build.xml). Brief explanation about the directories:

1. TK1WebServiceServer: Eclipse project for the web service's server
    a. **core** contains main class and the GUI class
    b. **iface** contains the interface
    c. **impl** contains the implementation classes of the interface
    d. **objects** contains custom Java objects
    e. **utils** contains several classes with static function that holds the server's data and XML translator from Java objects to xml string
2. TK1WebServiceClient: Eclipse project for the web service's client
    a. **core** contains main class, GUI class, and support classes for custom swing components
    b. **iface** contains the interface implemented by the main class
    c. **impl** contains the .java stub files generated from the server using *wsimport*
    d. **objects** contains custom Java objects
    e. **utils** contains several classes with static function that parse XML string from server to client-side custom Java objects

HOW TO RUN

1. Open the server's project folder and type *ant* in the terminal to run the build.xml. it will compile and show the control panel for the server
2. Open the client's project folder and type *ant* in the terminal to run the build.xml. It will compile, generate stubs, and execute 2 clients: 1 client using SOAP-RPC protocol and another using RESTful protocol
3. The SOAP service uses port 8090 and can be accessed from this URL: `http://localhost:8090/ws/tk1wsshoppingcart?wsdl`
   The RESTful service uses port 8080 and can be accessed from this URL: `http://localhost:8080/tk1wsshoppingcart/application.wadl`

IMPORTANT CLASSES

1. Server side
    a. **Server.class** is the main class that gets executed and will run 2 different service: SOAP-RPC service which corresponds to **ServerSOAPImpl.class** and RESTful service which corresponds to **ServerRESTImpl.class**
    b. **ServerGUI.class** provides control panel for activate/deactivating the service and showing statuses

    c. **IServer.class** contains the abstract function that needs to be implemented by **ServerSOAPImpl.class** and **ServerRESTImpl.class**
    d. **ServerData.class** contains static methods and global variables (e.g. List of items, client's cart, etc.). Important methods include XML builder (**XMLUtil.class**) to encode custom Java objects to XML strings
2. Client side
    a. **WebServClient.class** is the main class that accepts 1 parameter: 'soap' or 'rest' which determines which service the client will use.
    b. **ClientGUI.class** provides GUI for client's shopping cart. It also holds several private classes for custom Swing components
    c. **IClient.class** contains the abstract function that needs to be implemented by **WebServClient.class**
    d. **XMLParser.class** contains static methods to parse XML strings returned by the server to client's custom Java objects.
    e. Classes in **impl** folder are the stubs generated by the command *wsimport* (used for client using SOAP-RPC service type)

XML STRUCTURE

All of the server's return value uses XML, except function `login` and `logout` which use plain text. The structure of the XML is as follow

1. For the list of item(s)

```
<data>
      <item>
            <id>..</id>
            <name>..</name>
            <price>..</price>
            <amount>..</amount>
      </item>
      ..
</data>
```

2. For status message (status code > 0 means successful operation, code < 0 means failed)

```
<data>
      <status>..</status>
      <msg>..</msg>
</data>
```

EVENT EXPLANATIONS

SOAP-RPC methods | RESTful service URL

1. When user logs in

`IServer:login()`

`localhost:8080/tk1wsshoppingcart/server/login`

a. It sends a login request to the server
b. Server increments CLIENT_COUNT, which then serves as the client's ID
c. Server creates an empty cart for the client
d. Server returns the client's ID back to the client
e. client automatically requests the list of item

2. When user requests list of items

   `IServer:getItems()`

   `localhost:8080/tk1wsshoppingcart/server/getItems`

   a. It sends the request to the server
   b. Server retrieves the list of items and subtracts it with the amount of items taken in all clients' cart to get the updated item's list.
   c. Server then transforms the array of **ItemObject** into XML string and then returned to the client
   d. Client parse the XML and put it into the table in the GUI

3. When user adds an item to the cart

   `IServer:addToCart(clientId,itemId,amount)`

   `localhost:8080/tk1wsshoppingcart/server/addToCart/{clientId}/{itemId}/{amount}`

   a. It sends the request to the server with parameter: [clientId, itemId, amount]
   b. If the amount is 0, server assumes that the user wants to remove the item from his/her cart. The server will access his cart, remove the item, and sends back "successful remove" message back to the client
   c. If not, the server will traverse to all clients' cart and gather the number of the same item which resides in other client's carts. It's then subtracted with the actual amount in the server
   d. If the subtracted amount is lesser than the requested amount, the server will send back "not enough" message to the client
   e. If the subtracted amount is greater than the requested amount, the server will add the requested amount to the user's cart and send back "success" message to the client
   f. The client will then request for the updated list of items and the content of his/her cart

4. When user requests the content of his/her cart

   `IServer:getClientCart(clientId)`

   `localhost:8080/tk1wsshoppingcart/server/getClientCart/{clientId}`

   a. It sends the request to the server with parameter: [clientId]
   b. The server gets the cart by the clientId, transforms it into XML string and is returned back to the client
   c. Client receives the XML string, parse it, and applies it to the GUI

5. When user checks out

   `IServer:checkOutCart(clientId)`

   `localhost:8080/tk1wsshoppingcart/server/checkoutCart/{clientId}`

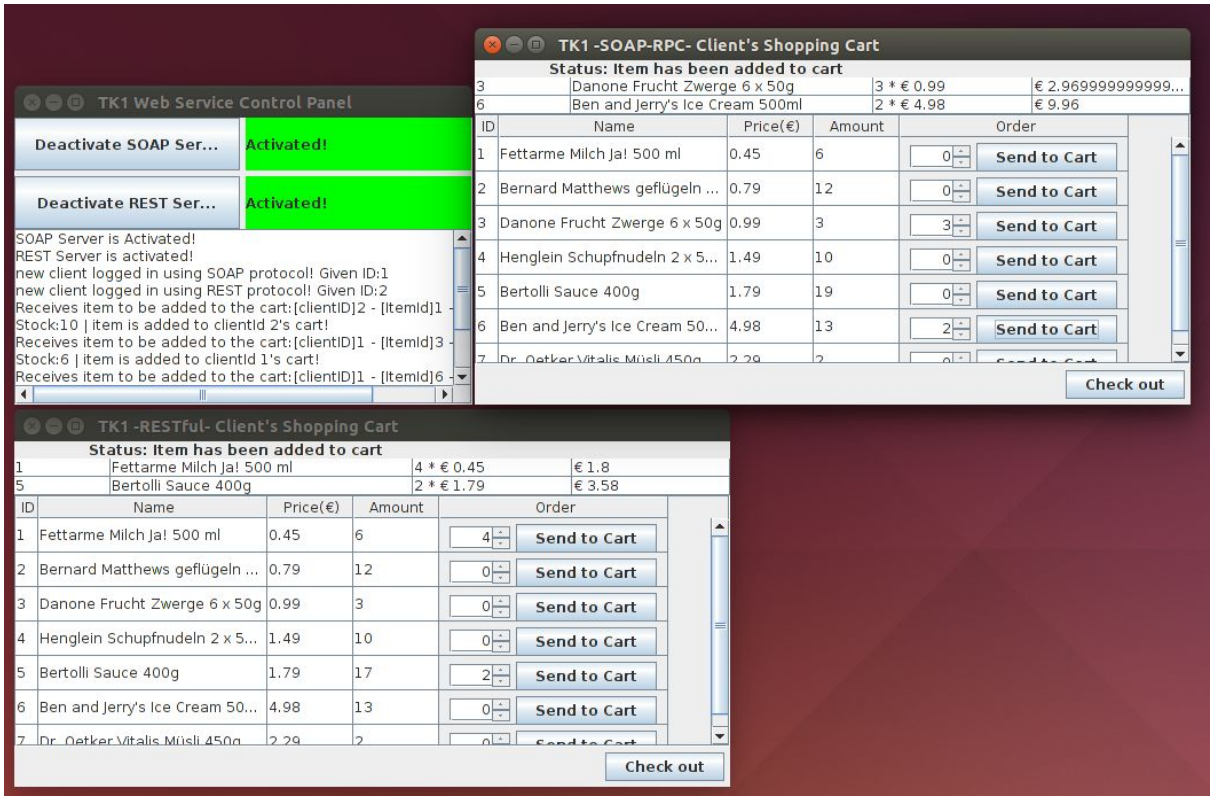
   a. It sends the request to the server with parameter: [clientId]
   b. The server gets the request, retrieves the cart by client's ID, and checks the content
   c. If it's empty, send back "cannot check out empty cart" message.
   d. If it's not, it gathers all the item, subtract the amount from the actual item's list, calculate total price, and send back "success" message to the client
   e. Client receives the response, and build the message dialog
6. When user logs out

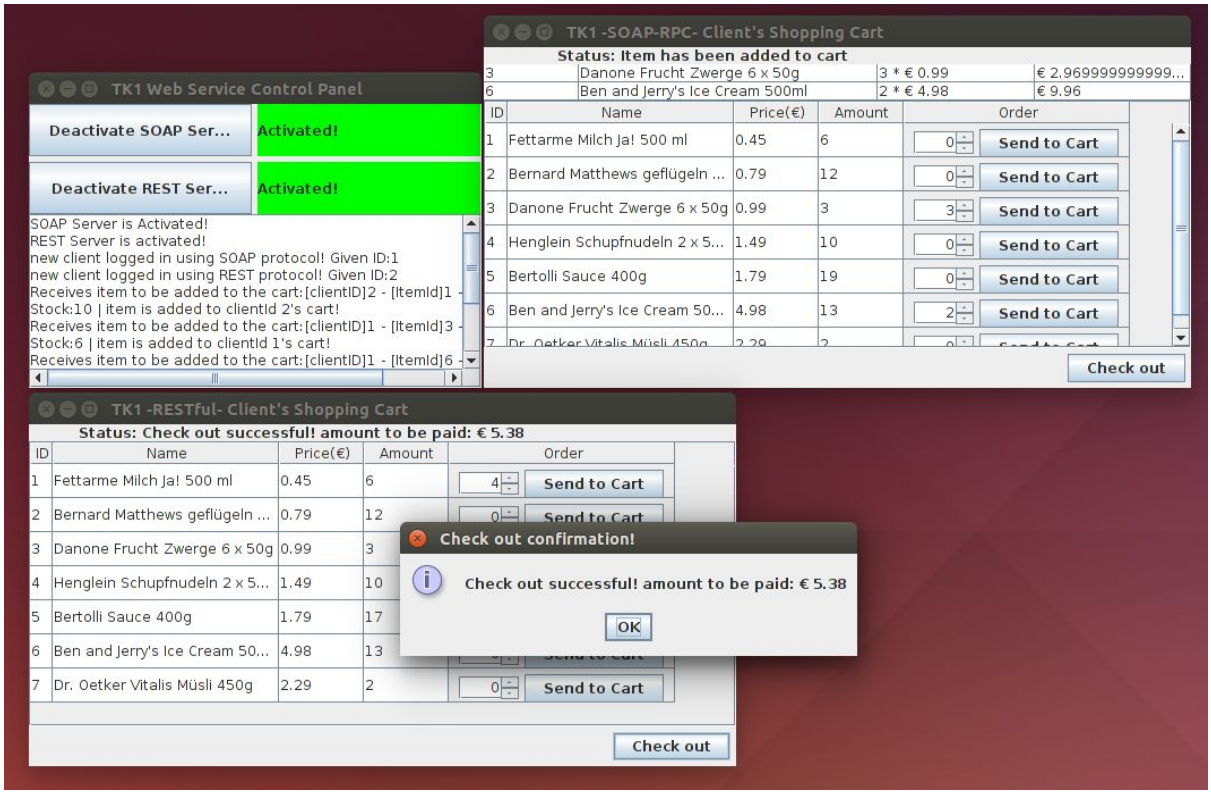
   `IServer:logout(clientId)`

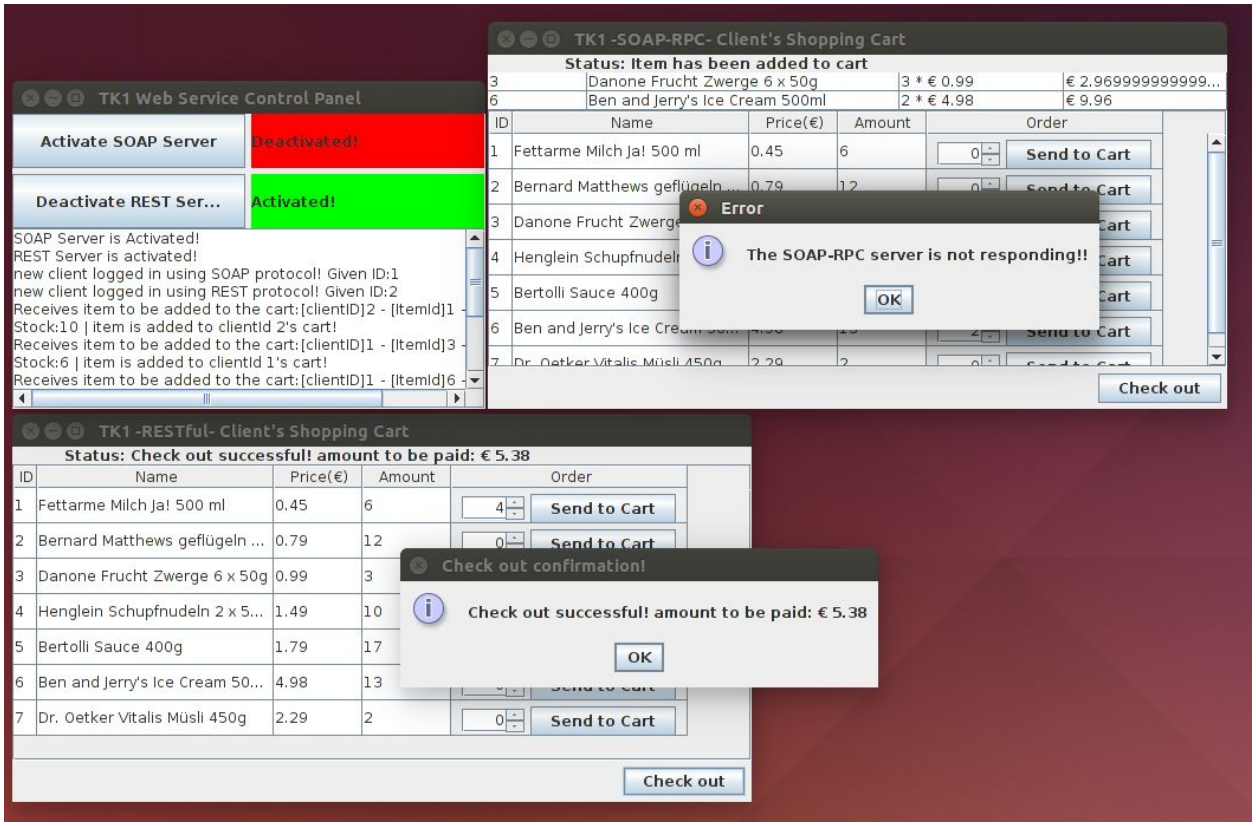
   `localhost:8080/tk1wsshoppingcart/server/logout/{clientId}`

   a. It sends the request to the server with parameter: [clientId]
   b. The server removes all items in the client's cart and the cart itself.
   c. Server returns "1" status code. (Needs to be done because both SOAP-RPC and RESTful service won't accept void-type returns. The status code can easily be ignored by the client).

SCREENSHOTS




TK1 Web Service Control Panel
Deactivate SOAP Ser...
Activated!
Deactivate REST Ser...
Activated!
SOAP Server is Activated!
REST Server is activated!
new client logged in using SOAP protocol! Given ID:1
new client logged in using REST protocol! Given ID:2
Receives item to be added to the cart:[clientID]2 - [ItemId]1 -
Stock:10 | item is added to clientId 2's cart!
Receives item to be added to the cart:[clientID]1 - [ItemId]3 -
Stock:6 | item is added to clientId 1's cart!
Receives item to be added to the cart:[clientID]1 - [ItemId]6 -
TK1 -SOAP-RPC- Client's Shopping Cart
Status: Item has been added to cart
3 Danone Frucht Zwerge 6 x 50g 3 * € 0.99 € 2.969999999999...
6 Ben and Jerry's Ice Cream 500ml 2 * € 4.98 € 9.96
ID Name Price(€) Amount Order
1 Fettarme Milch Ja! 500 ml 0.45 6 Send to Cart
2 Bernard Matthews geflügeln ... 0.79 12 Send to Cart
3 Danone Frucht Zwerge 6 x 50g 0.99 3 Send to Cart
4 Henglein Schupfnudeln 2 x 5... 1.49 10 Send to Cart
5 Bertolli Sauce 400g 1.79 19 Send to Cart
6 Ben and Jerry's Ice Cream 50... 4.98 13 Send to Cart
Check out
TK1 -RESTful- Client's Shopping Cart
Status: Check out successful! amount to be paid: € 5.38
ID Name Price(€) Amount Order
1 Fettarme Milch Ja! 500 ml 0.45 6 Send to Cart
2 Bernard Matthews geflügeln ... 0.79 12 Send to Cart
3 Danone Frucht Zwerge 6 x 50g 0.99 3
4 Henglein Schupfnudeln 2 x 5... 1.49 10
5 Bertolli Sauce 400g 1.79 17
6 Ben and Jerry's Ice Cream 50... 4.98 13
7 Dr. Oetker Vitalis Müsli 450g 2.29 2 Send to Cart
Check out
Check out confirmation!
Check out successful! amount to be paid: € 5.38
OK


TK1 Web Service Control Panel
Activate SOAP Server
Deactivated!
Deactivate REST Ser...
Activated!
SOAP Server is Activated!
REST Server is activated!
new client logged in using SOAP protocol! Given ID:1
new client logged in using REST protocol! Given ID:2
Receives item to be added to the cart:[clientID]2 - [ItemId]1 -
Stock:10 | item is added to clientId 2's cart!
Receives item to be added to the cart:[clientID]1 - [ItemId]3 -
Stock:6 | item is added to clientId 1's cart!
Receives item to be added to the cart:[clientID]1 - [ItemId]6 -
TK1 -SOAP-RPC- Client's Shopping Cart
Status: Item has been added to cart
3 Danone Frucht Zwerge 6 x 50g 3 * € 0.99 € 2.969999999999...
6 Ben and Jerry's Ice Cream 500ml 2 * € 4.98 € 9.96
ID Name Price(€) Amount Order
1 Fettarme Milch Ja! 500 ml 0.45 6 Send to Cart
2 Bernard Matthews geflügeln ... 0.79 12
3 Danone Frucht Zwerge
4 Henglein Schupfnudeln
5 Bertolli Sauce 400g
6 Ben and Jerry's Ice Cream
Error
The SOAP-RPC server is not responding!!
OK
Check out
TK1 -RESTful- Client's Shopping Cart
Status: Check out successful! amount to be paid: € 5.38
ID Name Price(€) Amount Order
1 Fettarme Milch Ja! 500 ml 0.45 6 Send to Cart
2 Bernard Matthews geflügeln ... 0.79 12 Send to Cart
3 Danone Frucht Zwerge 6 x 50g 0.99 3
4 Henglein Schupfnudeln 2 x 5... 1.49 10
5 Bertolli Sauce 400g 1.79 17
6 Ben and Jerry's Ice Cream 50... 4.98 13
7 Dr. Oetker Vitalis Müsli 450g 2.29 2 Send to Cart
Check out
Check out confirmation!
Check out successful! amount to be paid: € 5.38
OK

NOTES

1. Data such as client's cart and list of items sold are kept in a static variable and hard-coded inside the server's code **(ServerData.java)**. If the server is restarted, all data will be lost
2. Client's ID is generated from the visitor count (variable **CLIENT_COUNT** in **ServerData.java**) which increases every time a client is logged in
3. When user adds a certain item to his/her cart, the server DOES NOT subtract the amount from its original list of item. Instead, when user adds something to the cart, the server iterates through all other clients' cart and subtract the amount with the actual amount in the actual list of item, then compares it with the client's requested amount. ONLY when the user successfully checks out his/her cart, will the server subtract the amount from the actual list
4. Users automatically log in to the server the moment the client application is executed.
5. List of items in the user's table will ONLY be updated when the user logs in, add something to the cart, or checks out the items.