TEST SUITE

Online Sales Portal (OSP), Group 22 (Animesh Jha, Nisarg Upadhyaya and Pranav Rajput)

Contents

NOTE

For the purpose of testing a sample database with the following data is used as and when required. References to this sample database are marked by **

- Managers:
  - admin (uniqueid: 0001)
- Buyers:
  - buyer1 (uniqueid: 1001, city: city1)
  - buyer2 (uniqueid: 1002, city: city1)
  - buyer3 (uniqueid: 1003, city: city2)
- Sellers:

- seller1 (uniqueid: 2001)
- seller2 (uniqueid: 2002)
- seller3 (uniqueid: 2003)

- Items:
  - item1 (uniqueid: 3001, isheavy: true, city: city1, category: cat1, seller: seller1)
  - item2 (uniqueid: 3002, isheavy: false, city: city2, category: cat2, seller: seller2)
- Category:
  - cat1(uniqueid: 4001)
  - cat2(uniqueid: 4002)
- BuyRequests:
  - req1(item: item1, buyer: buyer1, status: pending)
  - req2(item: item2, buyer: buyer2, status: rejected)
  - req3(item: item2, buyer: buyer3, status: approved)

CLASSES

1. Address Class

- Construction
  - Empty Fields

House Number	Street	Locality	City	State	Pin Code
“”	“”	“”	“”	“”	“”

OUTPUT - Exception raised for House Number, Locality, City and State for not conforming to minimum length criteria and Pin Code for being of invalid type.

  - Invalid Characters

House Number	Street	Locality	City	State	Pin Code
“@-402”	“?Lane”	“mira_bhayandar”	“new0delhi”	“dehradun123”	“876gh3”

OUTPUT - Exception raised for each of the 6 fields for being of invalid type.

  - Correct Parameters

House Number	Street	Locality	City	State	Pin Code

“A/306, Safal Parivesh”	“Aarohi Homes Lane”	“South Bopal”	“Ahmedabad”	“Gujarat”	“380058”

OUTPUT - Address saved successfully to database without any exceptions.

2. User Class

- Construction
  - Invalid Parameters

Name	Telephone Number	Email address
“”	“0123456789”	“random@random”

OUTPUT - Exception raised for Name for not conforming to minimum length criteria, and for telephone number and email address being invalid.

  - Correct Parameters

Name	Telephone Number	Email address
“Nisarg Upadhyaya”	“9033040695”	“nisarg1631@gmail.com”

OUTPUT - No exceptions raised. Unique id assigned to user. Database queried for users with that id. Only 1 user returned.

- Methods
  - Verify Password

Uses the *login* interface. Check the test suite for the same in interfaces section.

3. Manger Class

- Construction
  - Pass correct parameters for parent class and address (see correct parameters in address and user class), incorrect parameters for gender and date of birth

Gender	Date of Birth
“”	32/1/2001

OUTPUT - Exceptions raised for empty gender field and invalid date of birth.

- Pass all correct parameters

Gender	Date of Birth
"Male"	16/3/2001

OUTPUT - No exceptions raised.

- Methods
  - Audit, Negotiation

    Check the *audit,* negotiation interfaces for the same in interfaces section.
  - removeItem

    Uses the *removeItem* interface in Item class.
  - manageBuyer (or manageSeller)

    Buyer exists in system**

    Input: manageBuyer(buyer1)

    Output: buyer removed from database

    → query to database for buyer1 returns count 0

    → in case of seller query of database for items on sale by seller returns count 0

    → query to database for buy requests raise by removed buyer returns count 0

    Buyer doesn't exist in system

    Input: manageBuyer(buyer4)

    Output: prints "Buyer doesn't exist"
  - getInfo - Output: dictionary with details given during signup

4. Customer Class

- Construction
  - Pass correct parameters for parent class (see correct parameters in user class), incorrect parameters for city

    City - ""

    OUTPUT - Exceptions raised for empty city field.

    City - "New2Delhi"

    OUTPUT - Exceptions raised for incorrect field.
  - Pass all correct parameters

    City - "Ahmedabad"

    OUTPUT - No exceptions raised.

5. Buyer Class

- Constructor

- Constructor uses parent's constructor (No new parameters to be passed)
  Test using the test cases for customer class.

- Methods
  - generateBuyRequest ()

Buyer	Item	Price	output
buyer2	item1	2000	Successful (req4)
buyer3	item1	3000	Unsuccessful
buyer2	item1	-1000	Unsuccessful

Print requests list of buyer2**
OUTPUT - [req2, req4]

6. Seller Class

- Constructor
  Constructor uses parent's constructor (No new parameters to be passed). Test using the test cases for customer class.
- uploadItem
  Covered in **Items** class.
- generateBuyRequest (working covered in **BuyRequests** class) and print Seller's requests data member

Let there be item3 (uniqueid: 3003, isheavy: false, city: city1, category: cat3, seller: seller1), then Seller1's requests list:

BuyRequest	Buyer	Item	Price	Status
R1	Buyer1	item1	1200	Pending
R2	Buyer3	item2	3000	Pending
R3	Buyer1	item3	2000	Pending

- modifyRequestStatus
  Tested in **BuyRequest** class.
- approvePaymentStatus
  Tested in **BuyRequest** class
- removeItem
  Covered in **Items** class.

7. BuyRequest Class

- Constructor

Create a BuyRequest R1, R2 and R3 from a Buyer B to Seller S. While calling the constructor for a buy request, the program will check if there are enough items in the database, if not then the constructor would raise an exception.

BuyRequest	Buyer	Seller	Item	Buyer.City==Seller.City	Price	output
R1	Buyer1	Seller1	item1	True	2000	Created
R2	Buyer3	Seller2	item2	True	3000	Created
R3	Buyer3	Seller1	item1	False	1000	Not created

- Seller1 will call modifyRequest on R1 and that would change the price of R1

BuyRequest	Buyer	Seller	Item	Price	Status
R1	Buyer1	Seller1	item1	1200	Pending
R2	Buyer3	Seller2	item2	3000	Pending

- Buyer3 will call validateRequest on R2 after completing the transaction, this would notify Seller2 who will then call approvePaymentStatus after which R2's status would change to Approved

BuyRequest	Buyer	Seller	Item	Price	Status
R1	Buyer1	Seller1	item1	1200	Pending
R2	Buyer3	Seller2	item2	3000	Approved

- Buyer3 will call function negotiate of R2 and this should lead to an exception as approved buyrequests can't be negotiated on.

8. Item Class

- Construction
  - Invalid parameters

Name	Category	Price	Company	City	Photo	Info	Is Heavy
“”	cat4	-40	“”	“”	None	“”	None

OUTPUT - Exception raised for empty fields in Name, Company, City, Photo, Info and Is Heavy and for invalid parameters in Category and Price.

- Correct Parameters

Name	Category	Price	Company	City	Photo	Info	Is Heavy
“OnePlus TV”	cat2**	20000	“OnePlus”	“New Delhi”	tv.png	“6 months used smart android TV”	True
“Phillips Mixer”	cat2**	2000	“Phillips”	“Mumbai”	mixer.png	“Brand new 1kwatt mixer”	False

OUTPUT - Items added successfully to database with a unique id assigned to them.

- Methods
  - Add Items
    seller3** invokes add items for the aforementioned items (OnePlus TV and Phillips Mixer)
    Print all items on database**
    OUTPUT - [item1(seller: seller1), item2(seller: seller2), OnePlus TV (seller: seller3), Phillips Mixer (seller: seller3)]
    Print seller3 items list
    OUTPUT - [OnePlus TV (seller: seller3), Phillips Mixer (seller: seller3)]

  - Search Items
    Search(name=”TV”)
    OUTPUT - [OnePlus TV (seller: seller3)]
    Search (name=” Washing Machine”)

OUTPUT - [] (empty list)
Search(category=cat2)
OUTPUT - [item2(seller: seller2), OnePlus TV (seller: seller3), Phillips Mixer (seller: seller3)]

- Remove Items
  Remove(uniqueid=3002)
  Print all items on database**
  OUTPUT - [item1(seller: seller1), OnePlus TV (seller: seller3), Phillips Mixer (seller: seller3)]
  Print seller2 items list
  OUTPUT - [] (empty list)

9. Category Class

- Construction

  - Pass incorrect parameters
    name - ""
    OUTPUT - Exceptions raised for empty name field.
  - Pass all correct parameters
    name - "category1"
    OUTPUT - No exceptions raised.
    name - "category2"
    OUTPUT - No exceptions raised.
    Check for unique id: category1.id! = category2.id
    OUTPUT - True
    Query all categories in database**
    OUTPUT - [cat1, cat2, category1, category2]
  - Try creating category with same name as existing
    name - "cat2"
    OUTPUT - Exception raised for trying to save category with same name as existing

INTERFACES

1. Login Interface

Let the passwords of the following users** be as stated (for testing purposes only)

- admin - "pass1"
- buyer1 - "pass2"
- seller1 - "pass3"

Login function: bool Login (uniqueid, password, type)

- Testing manager

Unique ID	Password	Type	Output
0001	pass1	Manager	True
0001	pass2 (incorrect)	Manager	False
1001 (incorrect)	pass1	Manager	False
1001 (incorrect)	pass2 (incorrect)	Manager	False
0001	pass1	Buyer (incorrect)	False

- Testing buyer

Unique ID	Password	Type	Output
1001	pass2	Buyer	True
1001	pass3 (incorrect)	Buyer	False
2001 (incorrect)	pass2	Buyer	False
2001 (incorrect)	pass3 (incorrect)	Buyer	False
1001	pass2	Seller (incorrect)	False

- Testing seller

Unique ID	Password	Type	Output
2001	pass3	Seller	True
2001	pass0 (incorrect)	Seller	False
0001 (incorrect)	pass3	Seller	False
0001 (incorrect)	pass0 (incorrect)	Seller	False

2001	pass3	Manager (incorrect)	False

2. Signup Interface

Signup of the following users with all correct fields and email address is tested on an initially empty database:

Email ID	Type	Output
nisarg1631@gmail.com	Manager	(True, "Signup successful")
jha.animesh01@gmail.com	Manager	(True, "Signup successful")
nisarg1631@gmail.com	Manager	(False, "Manager with given email already exists")
nisarg1631@gmail.com	Buyer	(True, "Signup successful")
jha.animesh01@gmail.com	Buyer	(True, "Signup successful")
nisarg1631@gmail.com	Buyer	(False, "Buyer with given email already exists")
nisarg1631@gmail.com	Seller	(True, "Signup successful")
jha.animesh01@gmail.com	Seller	(True, "Signup successful")
nisarg1631@gmail.com	Seller	(False, "Seller with given email already exists")

3. Audit Interface

Let the database contain

- Items:
  - item1 (uniqueid: 3001, isheavy: true, city: city1, category: cat1, seller: seller1)
  - item2 (uniqueid: 3002, isheavy: false, city: city2, category: cat2, seller: seller2)
  - item3 (uniqueid: 3003, isheavy: true, city: city3, category: cat1, seller: seller1)
- Categories
  - cat1
  - cat2
- Pending Buy Requests
  - item1 (by buyer1, pending)

- item1 (by buyer2, pending)
- item3 (by buyer3, pending)

If cat1 is deleted

Item	Item Id	Category
item1	3001	UNK
item2	3002	cat2
item3	3003	UNK

If item1 is deleted

Item	Item Id	Category
item2	3002	cat2
item3	3003	UNK

Pending Buy Requests:
item3 (by buyer3, pending)

4. Negotiation Interface

Tests covered in the **BuyRequest** class test suite.