Emipro Technologies Pvt. Ltd.

2nd Floor, “Shyamantak”, Everest Park, Near Hotel Regenta RPJ, Nana Mava, Kalawad Road, Rajkot 360005, Gujarat, India

CIN: U72200GJ2014PTC081583

Instructions

- Create separate file for each programs
- Write appropriate comments for each program in methods, classes
- Write appropriate variable names, method names

Program - 1

- Write a program that contains a ‘manufacturing’ class which allows product manufacturing.
- It purchases raw material from the market and produces a product.
- One should manage the raw material product to produce the actual product.
- One should manage the ratio of raw material to produce the actual product.
    - Example - 2 wheels(raw material) are required to produce 1 bicycle(actual product)
- Take input from user raw material, raw material ratio qty, actual product
- While producing the actual product, if the system doesn’t have enough of its stock of raw material, it should show a warning message as “Not enough raw material available to produce the product, please do the purchase”.
- Program should be menu driven allowing to
    - Purchase Raw Material Product
    - Manufacture Actual Product
    - Show Raw Material Quantity
    - Show Actual Product Quantity
    - Exit
- Following attributes and method function should be part of the class
    - Appropriate constructor to set the values for raw material product, actual product, raw material ratio qty value
    - produce() - taking no qty to be produced for actual product as argument
    - display_raw_material_stock()
    - display_final_product_stock()
    - purchase_raw_material() - taking no of qty of raw material to be purchased

Program - 2

- Write a program which extends Program - 1
- Add appropriate class and perform inheritance
- Add the facility of scrapping the raw material product and actual product
- Add appropriate option in the menu for
    - scrapping the raw material product
    - scrapping the actual product
- Add appropriate methods to scrap the raw material product and actual product

Program - 3

- Write a program which is having a class InventoryManagement.
- System should manage the quantity of purchase and sales processes.
- If the system is not having enough quantity during sales, the system should show a warning message “Not enough product quantities to sell”.
- Write appropriate constructor, methods to manage the sales and purchase processes.
- There should be a provision to view the available quantities.
- This should be a menu driven program, which allows to
    - Purchase Product
    - Sale Product
    - View Available Product Quantities
    - Exit

- Whenever sales take place, the system should process the quantities and then should show the available quantities.

Program - 4

- Write a program which is having a class InventoryManagement.
- System should manage the quantity of purchase and sales processes.
- Users should be able to purchase the product with different prices in the same object of Inventory Management - Use a dictionary, maintain a numerical index and store product price and product quantity against it.
- Whenever the product is sold, it will start deducting the product qty whichever entry is done first in the dictionary, so qty will be deducted as FIFO (First In First Out).
- Once the qty from each purchase is used make sure, it is not used again, so update the dictionary accordingly.
- Program should be able to handle the valuation.
- Formula for valuation - sum of price * qty / sum of total qty.
- Write appropriate constructor, methods to manage the sales and purchase processes.

- There should be a provision to view the available quantities.
- This should be a menu driven program, which allows to
    - Purchase Product
    - Sale Product
    - View Available Product Quantities
    - Exit

- Whenever sales take place, the system should process the quantities and then should show the available quantities.

Program - 5

Write a program to read xml file.
Read parent tag and child tag inside that parent tag.
Read the attribute of that child tag and print the value of that attribute.
Use xml.dom.minidom library
Example - from xml.dom.minidom import parse
Copy the below xml text and save it as .xml file
Use parse(path of xml file) to read xml file from path

```
<odoo>
<template id="_assets_backend_helpers" inherit_id="web._assets_backend_helpers">
        <xpath expr="//link" position="before">
                <link rel="stylesheet" type="text/scss"
        href="/account/static/src/scss/variables.scss"/>
        </xpath>
</template>
<template id="assets_backend" name="account assets" inherit_id="web.assets_backend">
        <xpath expr="." position="inside">
                <link rel="stylesheet"
                href="/account/static/src/css/account_bank_and_cash.css"/>
                <link rel="stylesheet" href="/account/static/src/css/account.css"/>
                <link rel="stylesheet" type="text/scss"
                href="/account/static/src/scss/account_reconciliation.scss"/>
                <link rel="stylesheet" type="text/scss"
                href="/account/static/src/scss/account_journal_dashboard.scss"/>
                <link rel="stylesheet" type="text/scss"
                href="/account/static/src/scss/account_dashboard.scss"/>
                <link rel="stylesheet"
                href="/account/static/src/scss/section_and_note_backend.scss"/>
                <script type="text/javascript"
                src="/account/static/src/js/reconciliation/reconciliation_action.js"/>
                <script type="text/javascript"
                src="/account/static/src/js/reconciliation/reconciliation_model.js"/>
```

```
                <script type="text/javascript"
                src="/account/static/src/js/reconciliation/reconciliation_renderer.js"/>
                <script type="text/javascript"
                src="/account/static/src/js/reconciliation/tour_reconciliation.js"/>
                <script type="text/javascript"
                src="/account/static/src/js/account_payment_field.js"/>
                <script type="text/javascript" src="/account/static/src/js/bank_statement.js"/>
                <script type="text/javascript"
                src="/account/static/src/js/section_and_note_fields_backend.js"/>
                <script type="text/javascript" src="/account/static/src/js/tour.js"/>
        </xpath>
</template>
<template id="assets_frontend" name="account assets" inherit_id="web.assets_frontend">
        <xpath expr="." position="inside">
                <script type="text/javascript"
        src="/account/static/src/js/account_portal_sidebar.js"/>
        </xpath>
</template>
<template id="qunit_suite" name="account_reconciliation tests"
inherit_id="web.qunit_suite">
        <xpath expr="//script[contains(@src, '/web/static/tests/views/kanban_tests.js')]"
        position="after">
                <script type="text/javascript"
                src="/account/static/tests/reconciliation_tests.js"/>
                <script type="text/javascript"
                src="/account/static/tests/account_payment_field_tests.js"/>
        </xpath>
</template>
</odoo>
```

Program - 6

Create a program that read a .csv file and print its data in dictionary
Data is about Sales Order
The dictionary should have SO number as key and its values as dictionaries

{SO001 :
{‘customer’ : {‘name’:<customer name>,‘address 1’ : <value of address1>,’address 2’: <value of address 2>, ‘city’ :<city>, ‘country’: <country>},
’Products’:[
{‘sku’:<sku>,’price’:<price>,’qty’:<qty>},
{‘sku’:<sku>,’price’:<price>}

```
                    ]
                }
            }
```

There should be a provision in the program where the code of the country is converted to the country's full name and only the full name of the country should be stored in a dictionary.
Use import csv library for managing csv task

Order No	Customer	SKU	Qty	Price	Address1	Address2	Zipcode	City	Country
SO001	Mohit Patel	STARCD173501	1	25	Address0002	Address20001	67806	Amritsar	USA
SO001	Mohit Patel	STARCD173502	2	30	Address0002	Address20001	67806	Amritsar	USA
SO003	Ketan Desai	MALTAF104511	7	45	Address0011	Address20030	67806	Sydney	AU
SO001	Mohit Patel	STARCD173503	5	87	Address0002	Address20001	28777	Amritsar	USA
SO004	Ankit Batra	STARCD173501	6	88	Address0005	Address20012	22589	Hamburg	ES
SO002	Rajesh	7PLUSF173201	5	89	Address0006	Address20007	67806	Dörrmoschel	DE
SO002	Rajesh	7PLUSF173201	6	90	Address0006	Address20007	67806	Dörrmoschel	DE
SO006	Mihir Tanna	7PLUSF173201	4	91	Address0008	Address20016	1720	New York	UK
SO002	Rajesh	7PLUSF173201	3	93	Address0006	Address20007	67806	Dörrmoschel	DE
SO003	Ketan Desai	MALTAF104512	1	94	Address0011	Address20030	67806	Sydney	AU

SO004	Ankit Batra	MALTAF104511	4	95	Address0005	Address20012	22589	Hamburg	ES
SO003	Ketan Desai	MALTAF104513	2	97	Address0011	Address20030	67806	Sydney	AU
SO005	Prashant Singh	MALTAF104511	3	98	Address0015	Address20015	22589	Hamburg Sülldorf	IT
SO006	Mihir Tanna	ASTN173502	6	99	Address0008	Address20016	1720	New York	UK
SO002	Rajesh	ASTN173502	4	100	Address0006	Address20007	67806	Dörrmoschel	DE
SO004	Ankit Batra	ASTN173502	5	102	Address0005	Address20012	22589	Hamburg	ES
SO006	Mihir Tanna	MSCE183104	7	104	Address0008	Address20016	1720	New York	UK
SO004	Ankit Batra	MSCE183104	7	105	Address0005	Address20012	22589	Hamburg	ES
SO006	Mihir Tanna	MSCE183105	5	106	Address0008	Address20016	1720	New York	UK
SO006	Mihir Tanna	MSCE183109	6	108	Address0008	Address20016	1720	New York	UK
SO010	Ajit Kapoor	STARCF182801	2	112	Address0029	Address20035	28777	Peru	IT
SO003	Ketan Desai	STARCF182801	4	113	Address0011		67806	Sydney	AU
SO003	Ketan Desai	7PLUSF173202	5	114	Address0011		67806	Sydney	AU
SO010	Ajit Kapoor	7PLUSF173202	4	118	Address0029	Address20035	28778	Peru	IT
SO006	Mihir Tanna	STARCE172901	2	134	Address0008	Address20016	1720	New York	UK
SO003	Ketan Desai	STARC	4	136	Address	Address2	67806	Sydney	AU

		E172901			0011	0030			
SO006	Mihir Tanna	STARCE172902	7	137	Address 0008	Address2 0016	1720	New York	UK
SO003	Ketan Desai	STARCE172902	1	138	Address 0011	Address2 0030	67806	Sydney	AU
SO002	Rajesh	7PLUSF174501	5	141	Address 0006	Address2 0007	67806	Dörrmoschel	DE
SO002	Rajesh	MALTAE150301	6	144	Address 0006	Address2 0007	67806	Dörrmoschel	DE
SO002	Rajesh	MALTAE150301	2	146	Address 0006	Address2 0007	67806	Dörrmoschel	DE
SO001	Mohit Patel	STARCD173504	4	45	Address 0002	Address2 0001	1720	Amritsar	USA

Program - 7

Create a program that write a .csv file
From Program 6 read the .csv file and create a dictionary and after that write the values from that dictionary to .csv file.
Read the dictionary and write in the “.csv” file
Following will be the headers

Order No	Customer	SKU	Qty	Price	Address1	Address2	Zipcode	City	Country

Program - 8

Create a program that write a .csv file
From Program 6 read the .csv file and create a dictionary and after that write the values from that dictionary to .xls file.
Read the dictionary and write in the “.xls” file
Following will be the headers

Order No	Customer	SKU	Qty	Price	Address1	Address2	Zipcode	City	Country

Using following import

from openpyxl import Workbook

Program - 9

Write a program which will handle the Sales Transaction class.
This will be a menu driven program.
It should be able to store the product details such as product name, product unit price, product cost, product_type(stockable, consumable, service), stock - separate dictionary

- Generate the unique sku automatically as follows PRD1, PRD2, PRD3...*PRDn*
- Use sku as the key
- There will be a "manage product" method.

There should be a separate method which will do the preparation of a dictionary of the product and return the prepared values which will be used to add a new product to the dictionary.
The create method of product should return the newly created index(which is sku in our case)
There should be provision to increase stock of the product

There should be provision to enter the Customer details (unique customer code, customer name, customer email, customer phone) -
{cust_1:{‘name’:<value>,’email’:<value>,’phone’:<value>}}
{cust_1:{‘address1’:<value>,’address2’:<value>,‘city’:<value>,’state’:<value>,’country’:<value>,’zipcode’:<value>}}
There should be provision to enter the Address belonging to specific customer (customer code, Address1, Address2, City, Zipcode, State, Country)
The customer code in customer address must be from the customer details, otherwise warning is to be raised.

There should be Sales process where sales order is generated
SO order no should be automatically generated as SO1, SO2,....SOn
SO should contain multiple products from product which are already created
It should show the list of products and one can select from the list, similar to menu driven, so that one can enter more than one product to the sales order. If the same product is selected add the quantity, instead of creating new dict. One should enter the sku which will inform the program which product is to be added to the order. If the given SKU is not found in the product list, a warning is to be raised.
This inner dictionary of orderline should contain list of dictionary of product containing - product sku, product qty, product unit price, subtotal

{‘SO1’ : {
‘customer’ : ‘cust_1’,
‘order_lines’ : [
{‘product_sku’ : <value>,

‘unit_price’:<value>,
‘quantity’:<value>,
‘subtotal’:<value>,
‘state’:<value>},

{‘product_sku’ : <value>,
‘unit_price’:<value>,
‘quantity’:<value>,
‘subtotal’:<value>,
‘state’:<value>},

{‘product_sku’ : <value>,
‘unit_price’:<value>,
‘quantity’:<value>,
‘subtotal’:<value>,
‘state’:<value>}
,
,
, upto - n

],
‘order_date’:<value>,
‘state’:<value>,
‘order_total_amount’:<value>}

There should be provision to add customer to sales order, only customer code should be added to the sale order
If wrong customer code is entered, ask to re-enter or exit
Sales Order dictionary will contain all the orders with SO1 as key and value will be
-customer detail(customer code)
-all the products in that sales order is to set as dictionary
-total of the order
-state of the order - possible values will be ['draft','confirm',’done’,cancel']
-default state will be 'draft'
-Order date
There should be a provision to confirm order
There should be a provision to cancel the order
There should be a provision to mark the order as Done
There should be a provision to set the order to draft state(which are cancelled once)

Only confirmed orders can be marked as ‘done’.
Confirmed order can be cancelled.
Done order cannot be changed to cancelled or confirm or draft

Once the order is cancelled it cannot be reverted to confirm state directly, it should be first set to draft and then to confirm state again.

Order should contain the details of the customer - Customer code only, rest of the details should be fetched using the customer code and used in printing only
Print the specific sales order as follows

Order No : <SO number>
Order Status : <Order status>
Customer : <cust_code>, <customer name>
<phone>
<email>

Order Date : <order date>

<address 1>
<address 2>
<city>, <state>
<zipcode>
<country>

Product Name	Product Price	Product Quantity	Subtotal
Xyz	88	7	2342.33
Abc	22	3	546.56

Order Total - 2334908.00

Edit:
Try to create a method named “prepare_order_lines” which only returns a prepared order line dictionary, one at a time. One dictionary at a time will be returned from it. So if someone is entering 3-4 products in an order, it will be called that many times and one dictionary per product will be returned. You can add more methods to support “prepare_order_lines” if you want to. “prepared_order_lines” will be called from the “prepare_sales_order” method. But if you have some other logic then you can do it. Currently product code, product unit price, product qty, subtotal were added in each order line, add “state” to it with default value as “draft”.
Currently there is one menu to change the state of the order, instead convert it to following menus and separate methods for each process.

- Confirm Order
- Cancel Order
- Set To Draft
- Set to Done

When order is confirmed, change the state of each order line to “Confirm” and then change the state of order to “Confirm”.

Program - 10

[[2,4,5],[4,6,8],[3,6,9]]
Considering the above list, we need to have a sum of all the values of all the inner lists.
- Find the sum of only even numbers
- Find the sum of all the numbers from all the inner lists

Don’t use a loop.

Program - 11

```
emp_dict =  {
        101:{   'name': 'Anupriya Roy',
                'depart_id':1,
                'attendances':[{'date':1, 'hours':[3.5,4.5]},{'date':2, 'hours':[3.2,4.5]},{'date':3,
'hours':[3.2,4.6]},
                        {'date':4, 'hours':[3.0,4.5]},{'date':5, 'hours':[2.5,4.5]},{'date':6,
'hours':[1.5,4.5]},
                        {'date':7, 'hours':[2,3]},{'date':8, 'hours':[0,4.5]},{'date':9, 'hours':[2,3.5]},
                        {'date':10, 'hours':[4,3.5]}],
                'leaves':[{'date':7, 'no_of_hours':1.5},{'date':7, 'no_of_hours':1.5},{'date':8,
'no_of_hours':3}]
            },
        102:
        {   'name': 'Kadambari Sharma',
            'depart_id':1,
            'attendances':[{'date':1, 'hours': [0,4.5]},{'date':2, 'hours':[3.2,0]},{'date':3,
'hours':[3.2,4.6]},
                        {'date':4, 'hours':[1,4.5]},{'date':5, 'hours':[2.5,2]},{'date':6, 'hours':[1.5,1]},
                        {'date':7, 'hours':[2,4]},{'date':8, 'hours':[1,4.5]},{'date':9, 'hours':[2,2]},
                        {'date':10, 'hours':[2,3.5]}],
            'leaves':[{'date':1, 'no_of_hours':3.5},{'date':2, 'no_of_hours':2},{'date':2,
'no_of_hours':2}]
        },
        103:
        {   'name': 'Abhishek Verma',
            'depart_id':1,
            'attendances':[{'date':3, 'hours':[3.2,4.6]},{'date':4, 'hours':[1,4.5]},{'date':5,
'hours':[2.5,2]},
                    {'date':6, 'hours':[1.5,1]},{'date':7, 'hours':[2,4]},{'date':8, 'hours':[1,4.5]},
                    {'date':9, 'hours':[2,2]},{'date':10, 'hours':[2,3.5]}
            ],
            'leaves':[{'date':1, 'no_of_hours':3},{'date':2, 'no_of_hours':2},{'date':2,
'no_of_hours':3}]
```

```
        }
}
```

Find the following outputs - **Don’t use a loop. The only loop that will be used is the main dictionary loop of emp_dict**

1.
Employee's total attendance and leave hours :

```
[
        {'employee_id': 101, ‘employee_name’:’Anupriya Roy’,'total_attendance_hours': 66.5,
        'total_leave_days': 6.0},
        {'employee_id': 102,,’employee_name’:'Kadambari Sharma',
        'total_attendance_hours': 49.0, 'total_leave`_days': 7.5},
        {'employee_id': 103,’employee_name’:'Abhishek Verma', 'total_attendance_hours':
        41.3, 'total_leave_days': 8}
]
```

2. Consider 8hrs of work duty, get the following output
Employee's day wise record in which they have not 7 hours attendance :

```
 [
        {101: {'date': [6, 7, 8, 9], 'total_hrs': [6.0, 5, 4.5, 5.5], 'remaining_hrs': [2.0, 3, 3.5,
        2.5]}},
        {102: {'date': [1, 2, 4, 5, 6, 7, 8, 9, 10], 'total_hrs': [4.5, 3.2, 5.5, 4.5, 2.5, 6, 5.5, 4,
        5.5], 'remaining_hrs': [3.5, 4.8, 2.5, 3.5, 5.5, 2, 2.5, 4, 2.5]}},
        {103: {'date': [4, 5, 6, 7, 8, 9, 10], 'total_hrs': [5.5, 4.5, 2.5, 6, 5.5, 4, 5.5],
        'remaining_hrs': [2.5, 3.5, 5.5, 2, 2.5, 4, 2.5]}}
]
```

Program - 12

This program is an extension to Program-9.
Read about “Delivery Order” in reference to “Sales Order”. You can search on Google for this.
When the order is confirmed, create the delivery order(which will be a dictionary similar to sales order, product or customer, take this dictionary in sale order class). The key will be automatically generated as follows : DO1, DO2, DO3...DOn
Each delivery order should be like as follows:

```
{DO1 : {‘sales_order’ : SO1,
                ‘stock_moves’:[{‘product_code’:<value>,
‘product_qty’:<value>,’state’:<value>}]
        ‘state’:<value>}}
```

Values for stock_moves will be copied from order lines of related sales order
Values for “state” will be - draft(default value), done, cancelled.
Remove the menu of “Set to Done” order.
Add menus

- Validate Delivery Order
- Cancel Delivery Order

The changes from Program-9 in this program will be that the quantity of product will be deducted from the master dictionary of products when the Delivery order is validated, not before that. That means not during order in draft or confirmed state.
Whenever Delivery order is validated, set the stock moves state to “done”, then mark the delivery order as done and then set the state of the related sales order as “Done”.
Whenever Delivery order is cancelled, set the stock moves state to “cancel” and then set the state of delivery order as “Cancel”. ~~Also, revert the quantities to the product dictionary.~~ Don’t change the state of sales order automatically.w

- If the current order state is confirmed and wants to cancel the order - Is there a delivery order created for this order, if yes is it in a cancel state then cancel the order. If it is not in a cancel state, then the system should not cancel the order and show the warning to cancel the delivery order first. If the delivery order is cancelled, then the system should allow you to cancel the order. If the delivery order is not created, then it should allow you to cancel the order. If the delivery order is in “done” state, then the order cannot be cancelled.
- If the current order state is confirmed and wants to set it to draft - Cannot directly set back to draft, first cancel the order and then bring it back to the draft state.
- If the current order state is cancelled and then changed to draft and then want to confirm it - Check if there is already any delivery order created for that sales order and is cancelled or not. As per the flow the delivery order must be cancelled, so now when the state is changed to confirm, it should create a new Delivery Order. In normal cases if it is getting confirmed for the first time, it will create the first Delivery Order, in case of cancel -> draft -> confirm it will keep on creating new delivery orders.