Indira College of Commerce and Science Pune M.Sc. (CS)-II (Semester –III) 2019 Pattern CSUP235: SPPU

Practical Examination (SADP, ML, WF)

Roll No 19 Valentina

Q1 Write a Java Program to implement Iterator Pattern for Designing Menu like Breakfast, Lunch or Dinner Menu.

Program:

```
import java.util.Iterator;

public interface Menu {
        public Iterator<?> createIterator();

        String name;
        public String getName() {
                return name;
        }
}


public class MenuItem {
        String name;
        String description;
        boolean vegetarian;
        double price;

        public MenuItem(String name,
                        String description,
                        boolean vegetarian,
                        double price)
        {
                this.name = name;
                this.description = description;
                this.vegetarian = vegetarian;
                this.price = price;
        }

        public String getName() {
                return name;
        }

        public String getDescription() {
                return description;
        }

        public double getPrice() {
                return price;
        }

        public boolean isVegetarian() {
                return vegetarian;
```

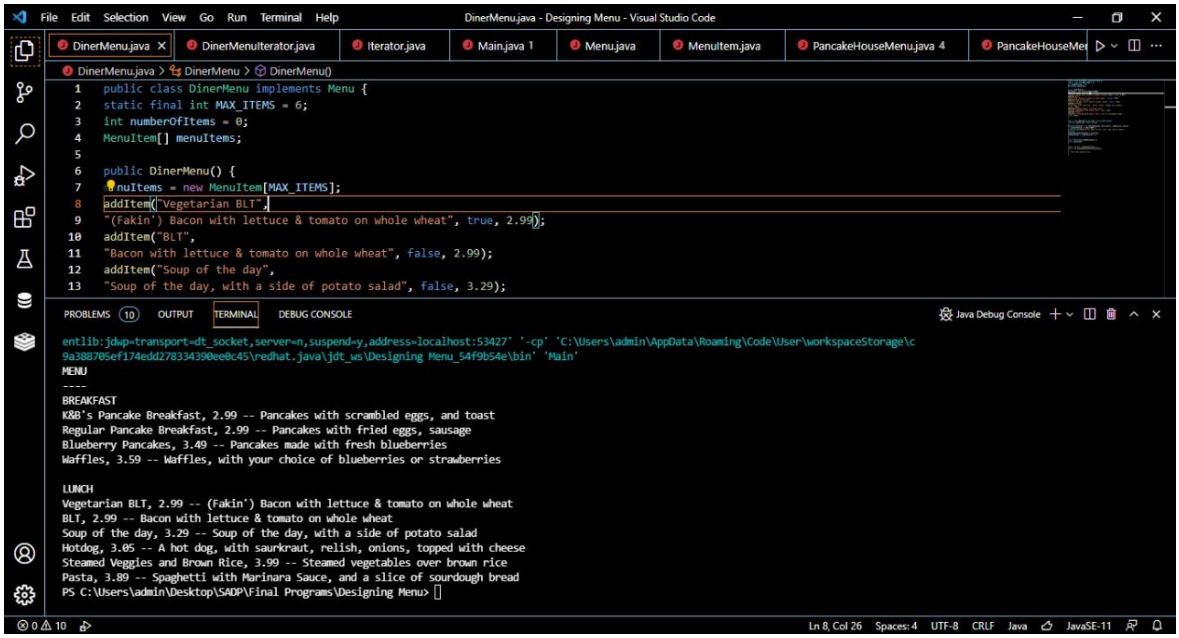
```
        }
}

public class PancakeHouseMenu implements Menu {
        ArrayList<MenuItem> menuItems;

        public PancakeHouseMenu() {
                name = "BREAKFAST";
                menuItems = new ArrayList<MenuItem>();

                addItem("K&B's Pancake Breakfast",
                        "Pancakes with scrambled eggs, and toast",
                        true,
                        2.99);

                addItem("Regular Pancake Breakfast",
                        "Pancakes with fried eggs, sausage",
                        false,
                        2.99);

                addItem("Blueberry Pancakes",
                        "Pancakes made with fresh blueberries, and blueberry
syrup",
                        true,
                        3.49);

                addItem("Waffles",
                        "Waffles, with your choice of blueberries or
strawberries",
                        true,
                        3.59);
        }

        public void addItem(String name, String description,
                            boolean vegetarian, double price)
        {
                MenuItem menuItem = new MenuItem(name, description, vegetarian,
price);
                menuItems.add(menuItem);
        }

        public ArrayList<MenuItem> getMenuItems() {
                return menuItems;
        }

        public Iterator<MenuItem> createIterator() {
                return menuItems.iterator();
        }


}

import java.util.Iterator;

public class DinerMenu implements Menu {
        static final int MAX_ITEMS = 6;
        int numberOfItems = 0;
        MenuItem[] menuItems;
```

```
        public DinerMenu() {
                name = "LUNCH";
                menuItems = new MenuItem[MAX_ITEMS];

                addItem("Vegetarian BLT",
                        "(Fakin') Bacon with lettuce & tomato on whole wheat",
true, 2.99);
                addItem("BLT",
                        "Bacon with lettuce & tomato on whole wheat", false,
2.99);
                addItem("Soup of the day",
                        "Soup of the day, with a side of potato salad", false,
3.29);
                addItem("Hotdog",
                        "A hot dog, with saurkraut, relish, onions, topped with
cheese",
                        false, 3.05);
                addItem("Steamed Veggies and Brown Rice",
                        "Steamed vegetables over brown rice", true, 3.99);
                addItem("Pasta",
                        "Spaghetti with Marinara Sauce, and a slice of sourdough
bread",
                        true, 3.89);
        }

        public void addItem(String name, String description,
                            boolean vegetarian, double price)
        {
                MenuItem menuItem = new MenuItem(name, description, vegetarian,
price);
                if (numberOfItems >= MAX_ITEMS) {
                        System.err.println("Sorry, menu is full!  Can't add item
to menu");
                } else {
                        menuItems[numberOfItems] = menuItem;
                        numberOfItems = numberOfItems + 1;
                }
        }

        public MenuItem[] getMenuItems() {
                return menuItems;
        }

        public Iterator<MenuItem> createIterator() {
                return new DinerMenuIterator(menuItems);
                //return new AlternatingDinerMenuIterator(menuItems);
        }

        public

        }
import java.util.Iterator;

public class DinerMenuIterator implements Iterator<MenuItem> {
        MenuItem[] list;
        int position = 0;

        public DinerMenuIterator(MenuItem[] list) {
                this.list = list;
```

```
        }

        public MenuItem next() {
                MenuItem menuItem = list[position];
                position = position + 1;
                return menuItem;
        }

        public boolean hasNext() {
                if (position >= list.length || list[position] == null) {
                        return false;
                } else {
                        return true;
                }
        }

        public void remove() {
                if (position <= 0) {
                        throw new IllegalStateException
                                ("You can't remove an item until you've done at
least one next()");
                }
                if (list[position-1] != null) {
                        for (int i = position-1; i < (list.length-1); i++) {
                                list[i] = list[i+1];
                        }
                        list[list.length-1] = null;
                }
        }
}


public class Waitress {
        ArrayList<Menu> menus;


        public Waitress(ArrayList<Menu> menus) {
                this.menus = menus;
        }

        public void printMenu() {
                Iterator<?> menuIterator = menus.iterator();

                System.out.print(MENU\n----\n);
                while(menuIterator.hasNext()) {
                        Menu menu = (Menu)menuIterator.next();
                        System.out.print("\n" + menu.getName() + "\n");
                        printMenu(menu.createIterator());
                }
        }

        void printMenu(Iterator<?> iterator) {
                while (iterator.hasNext()) {
                        MenuItem menuItem = (MenuItem)iterator.next();
                        System.out.print(menuItem.getName() + ", ");
                        System.out.print(menuItem.getPrice() + " -- ");
                        System.out.println(menuItem.getDescription());
                }
        }
```

```
}

public class MenuTestDrive {
        public static void main(String args[]) {
                PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
                DinerMenu dinerMenu = new DinerMenu();
                ArrayList<Menu> menus = new ArrayList<Menu>();
                menus.add(pancakeHouseMenu);
                menus.add(dinerMenu);
                Waitress waitress = new Waitress(menus);
                waitress.printMenu();

        }
}
```

Output:

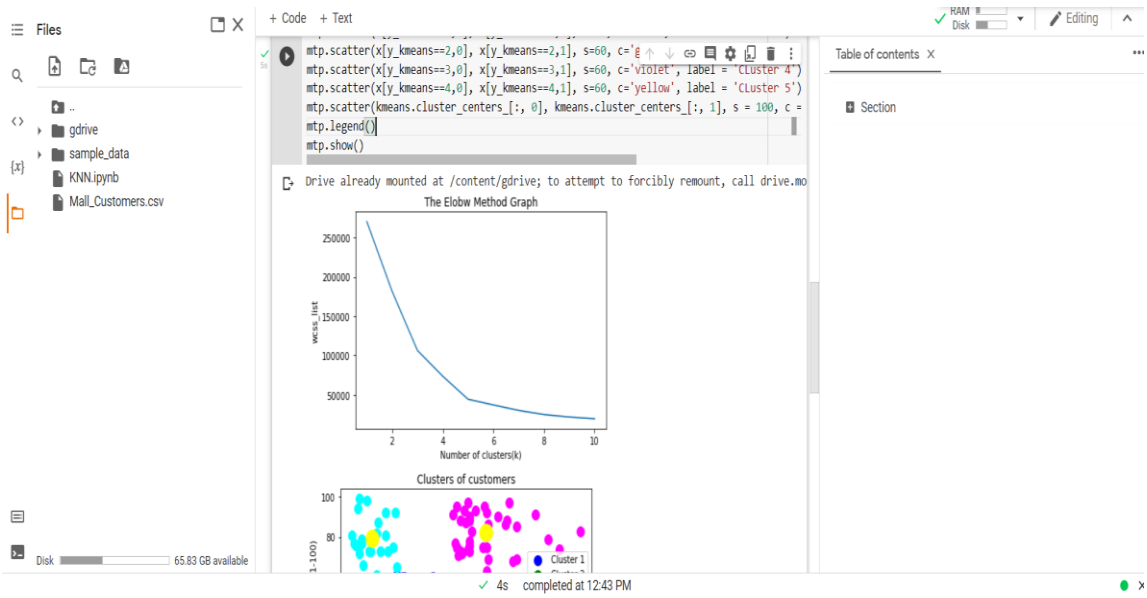


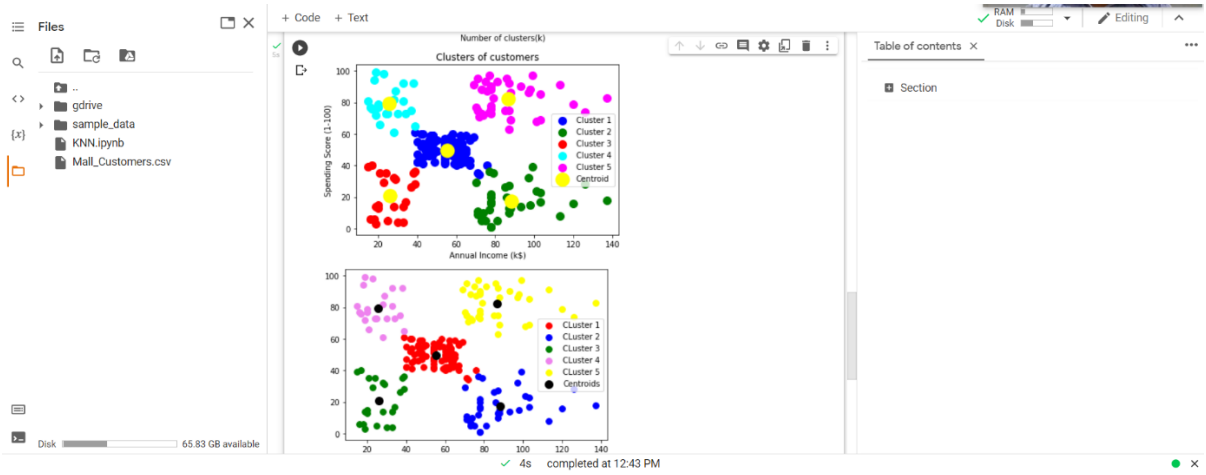
Q2.Write a python program to implement k-nearest Neighbors ML algorithm to build prediction model for given dataset.

```
# -*- coding: utf-8 -*-
"""K-means.ipynb

Automatically generated by Colaboratory.

Original file is located at
    /content/Mall_Customers.csv
```

```
"""

#https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-
learning

from google.colab import drive
drive.mount('/content/gdrive')

import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

data_set = pd.read_csv('/content/Mall_Customers.csv')

data_set.head(5)

x = data_set.iloc[:, [3, 4]].values
x[:5]

from sklearn.cluster import KMeans
wcss_list= []  #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()

kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
y_kmeans= kmeans.fit_predict(x)

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'b
lue', label = 'Cluster 1') #for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'g
reen', label = 'Cluster 2') #for second cluster
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 're
d', label = 'Cluster 3') #for third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'c
yan', label = 'Cluster 4') #for fourth cluster
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'm
agenta', label = 'Cluster 5') #for fifth cluster
```

```
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1
], s = 300, c = 'yellow', label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()

mtp.scatter(x[y_kmeans==0,0], x[y_kmeans==0,1], s=60, c='red', label =
'CLuster 1')

#  Here, X[y_kmeans==0,0] is X axis and X[y_kmeans==0,1] is Y axis
#  We're plotting scatters for cluster=0 i.e, our first cluster , in re
d color

mtp.scatter(x[y_kmeans==1,0], x[y_kmeans==1,1], s=60, c='blue', label =
 'CLuster 2')
mtp.scatter(x[y_kmeans==2,0], x[y_kmeans==2,1], s=60, c='green', label
= 'CLuster 3')
mtp.scatter(x[y_kmeans==3,0], x[y_kmeans==3,1], s=60, c='violet', label
 = 'CLuster 4')
mtp.scatter(x[y_kmeans==4,0], x[y_kmeans==4,1], s=60, c='yellow', label
 = 'CLuster 5')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1
], s = 100, c = 'black', label = 'Centroids')
mtp.legend()
mtp.show()
```





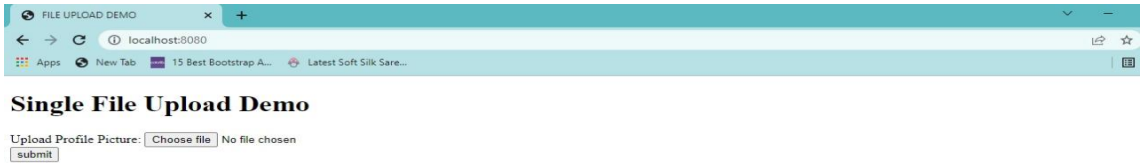
Q.3 A] Create a Node.js file that writes an HTML form, with an upload field.

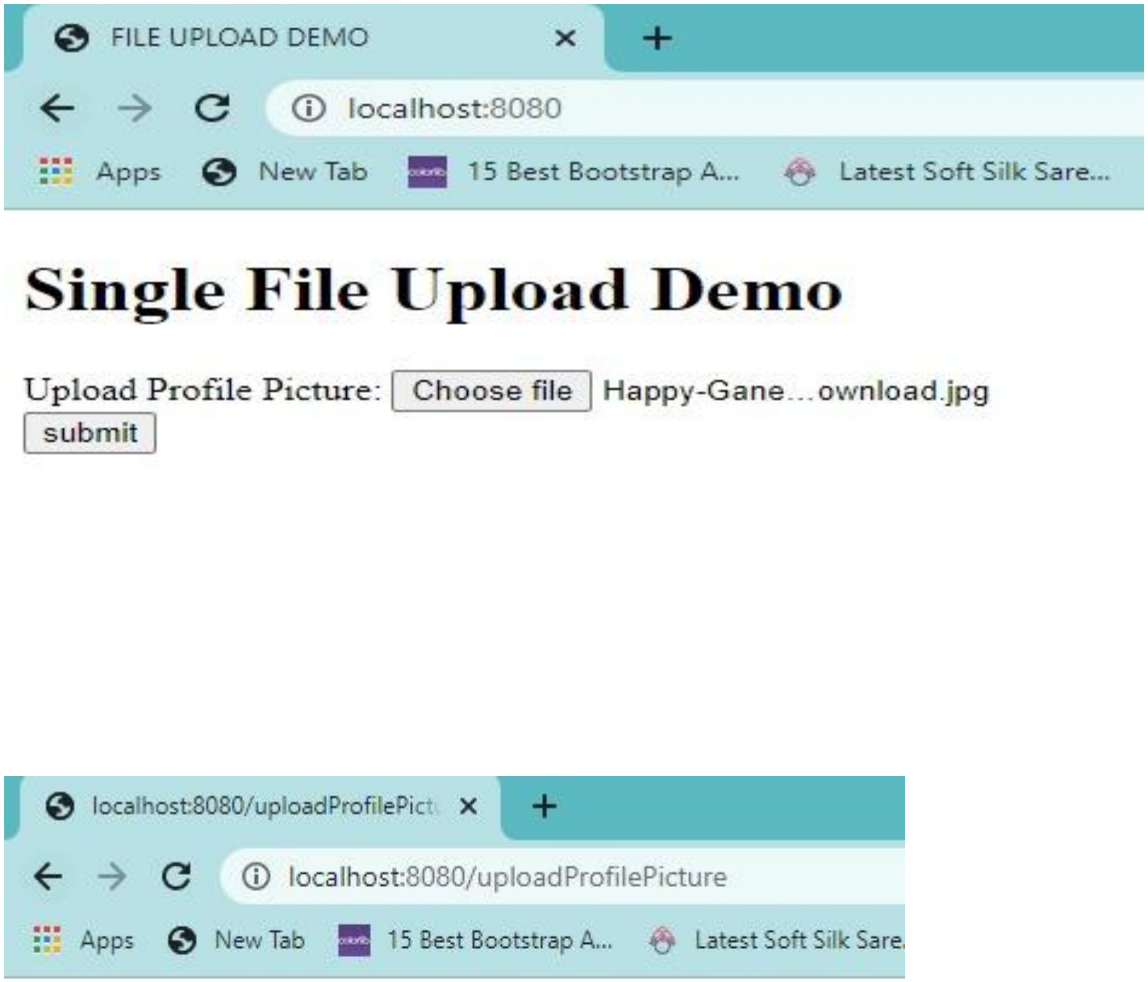
Program:

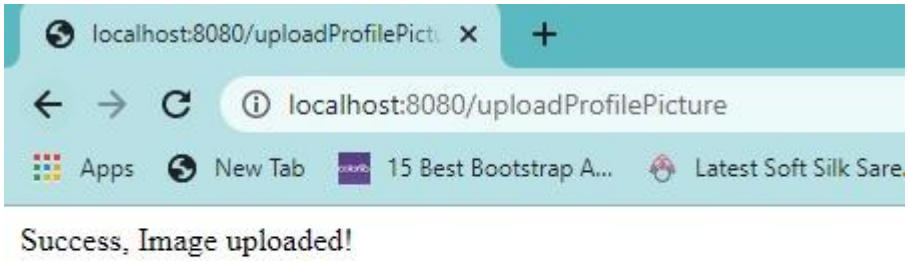
```
const express = require("express")
const path = require("path")
const multer = require("multer")
const app = express()

app.set("views",path.join(__dirname,"views"))
app.set("view engine","ejs")


var storage = multer.diskStorage({
    destination: function (req, file, cb) {


        cb(null, "uploads")
    },
    filename: function (req, file, cb) {
      cb(null, file.fieldname + "-" + Date.now()+".jpg")
    }
  })

const maxSize = 1 * 1000 * 1000;

var upload = multer({
    storage: storage,
    limits: { fileSize: maxSize },
    fileFilter: function (req, file, cb){


        var filetypes = /jpeg|jpg|png/;
        var mimetype = filetypes.test(file.mimetype);

        var extname = filetypes.test(path.extname(
                    file.originalname).toLowerCase());

        if (mimetype && extname) {
            return cb(null, true);
        }

        cb("Error: File upload only supports the "
                + "following filetypes - " + filetypes);
```

```
      }

}).single("mypic");

app.get("/",function(req,res){
    res.render("Signup");
})

app.post("/uploadProfilePicture",function (req, res, next) {


    upload(req,res,function(err) {

        if(err) {

            res.send(err)
        }
        else {


            res.send("Success, Image uploaded!")
        }
    })
})

app.listen(8080,function(error) {
    if(error) throw error
        console.log("Server created Successfully on PORT 8080")
})
```

Ouput:




FILE UPLOAD DEMO
localhost:8080
Apps
New Tab
15 Best Bootstrap A...
Latest Soft Silk Sare...
Single File Upload Demo
Upload Profile Picture:
Choose file
Happy-Gane...ownload.jpg
submit


localhost:8080/uploadProfilePict
localhost:8080/uploadProfilePicture
Apps
New Tab
15 Best Bootstrap A...
Latest Soft Silk Sare
Success, Image uploaded!

File Edit Selection View Go Run Terminal Help Signup.ejs - HTML form, with an upload field

EXPLORER

HTML FORM, WITH AN UPL...

- node_modules
- uploads
  - mypic-1645034098551.jpg
  - mypic-1645136747477.jpg
- views
  - Signup.ejs
- index.js

Welcome | Signup.ejs | index.js

views > Signup.ejs > html > body > form

```
5     <title>FILE UPLOAD DEMO</title>
6 </head>
7
8 <body>
9     <h1>Single File Upload Demo</h1>
10
11    <form action="/uploadProfilePicture"
```