WEEK 1   
Topic 1: Design Patterns and Principles

Exercise 1: Implementing the Singleton Pattern

Code :  
  
class DatabaseConnectionManager{

private static DatabaseConnectionManager instance;

private String connection;

private DatabaseConnectionManager(){

connection = "Connected to Database";

}

public static synchronized DatabaseConnectionManager getInstance(){

if (instance == null){

instance = new DatabaseConnectionManager();

}

return instance;

}

public String getConnection() {

return connection;

}

}

public class Main {

public static void main(String[] args) {

DatabaseConnectionManager manager1 = DatabaseConnectionManager.getInstance();

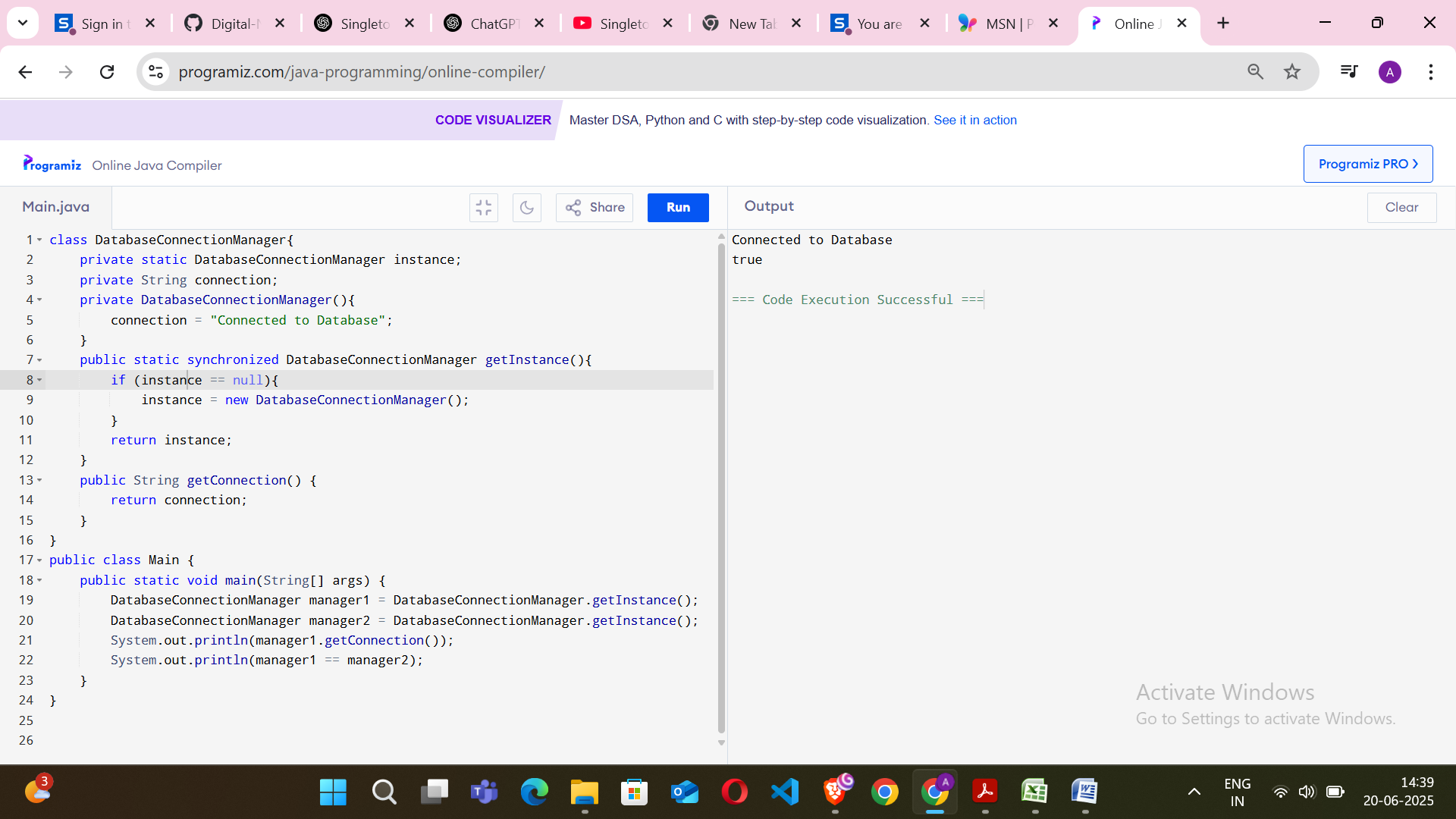
DatabaseConnectionManager manager2 = DatabaseConnectionManager.getInstance();

System.out.println(manager1.getConnection());

System.out.println(manager1 == manager2);

}

}

O/P :   
  


Exercise 2: Implementing the Factory Design Pattern

Code :

interface Notification {

void notifyUser();

}

class EmailNotification implements Notification {

public void notifyUser() {

System.out.println("Sending an Email Notification");

}

}

class SMSNotification implements Notification {

public void notifyUser() {

System.out.println("Sending an SMS Notification");

}

}

class PushNotification implements Notification {

public void notifyUser() {

System.out.println("Sending a Push Notification");

}

}

class NotificationFactory {

public static Notification createNotification(String type) {

if (type == null || type.isEmpty())

return null;

switch (type.toLowerCase()) {

case "sms":

return new SMSNotification();

case "email":

return new EmailNotification();

case "push":

return new PushNotification();

default:

throw new IllegalArgumentException("Unknown notification type: " + type);

}

}

}

public class Main {

public static void main(String[] args) {

Notification notification = NotificationFactory.createNotification("email");

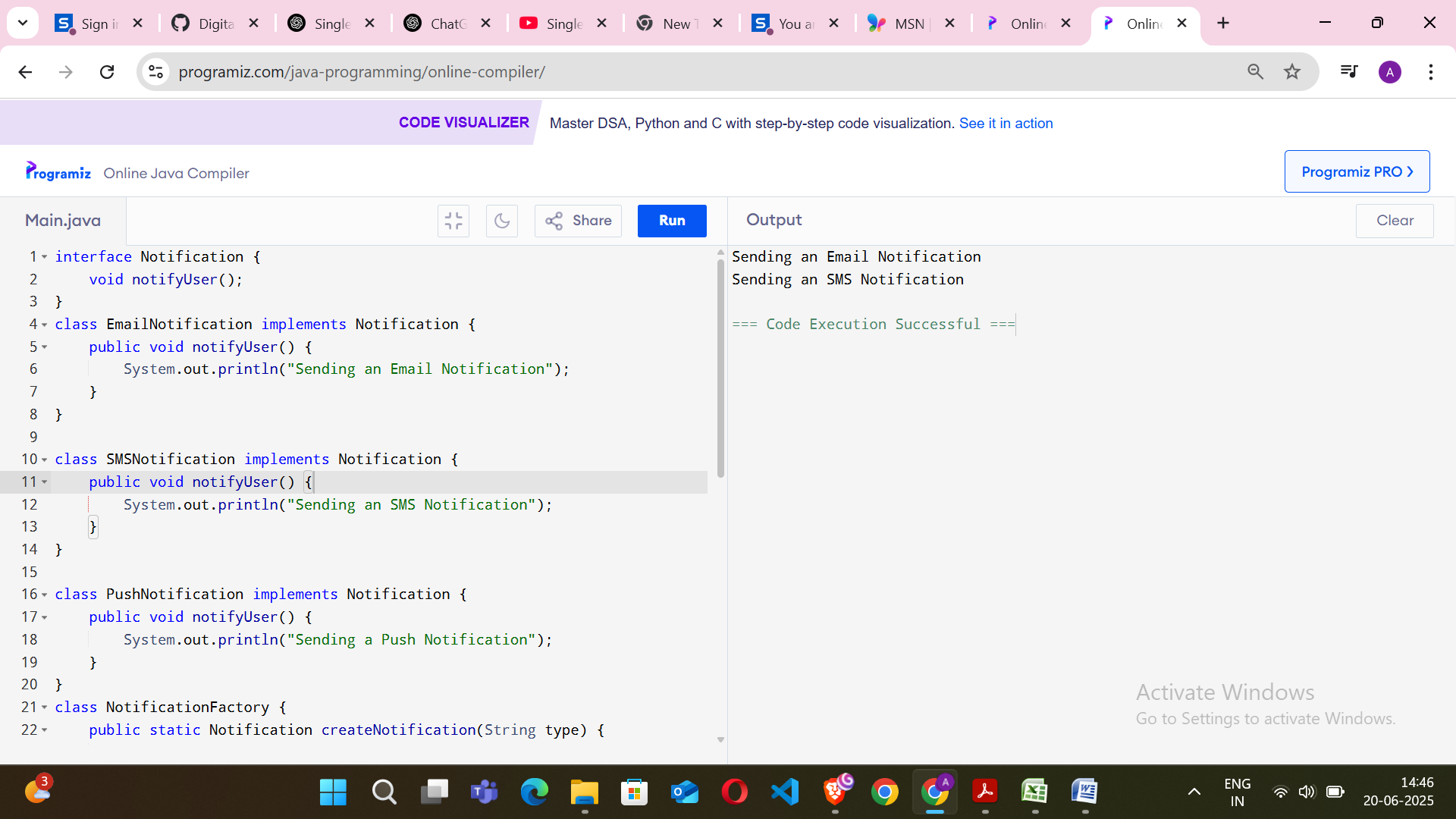
notification.notifyUser();

notification = NotificationFactory.createNotification("sms");

notification.notifyUser();

}

}

O/P:  
  


Topic 2: Algorithms\_Data Structures

Exercise 1: E-commerce Platform Search Function  
  
Code :  
  
import java.util.\*;

public class ECommerceSearch{

static class Product{

private String id;

private String name;

private String category;

public Product(String id, String name, String category){

this.id=id;

this.name=name.toLowerCase();

this.category=category.toLowerCase();

}

public String getName(){

return name;

}

public String getCategory(){

return category;

}

@Override

public String toString(){

return "Product [id=" + id + ", name=" + name + ", category=" + category + "]";

}

}

static class ProductSearch{

private List<Product> products;

public ProductSearch(){

products=new ArrayList<>();

loadProducts();

}

private void loadProducts(){

products.add(new Product("P001", "Samsung Galaxy M14", "Electronics"));

products.add(new Product("P002", "Nike Running Shoes", "Footwear"));

products.add(new Product("P003", "Galaxy Watch", "Accessories"));

products.add(new Product("P004", "Apple iPhone 13", "Electronics"));

products.add(new Product("P005", "Adidas Sports T-shirt", "Clothing"));

}

public List<Product> search(String keyword){

keyword=keyword.toLowerCase();

List<Product> results=new ArrayList<>();

for (Product p:products){

if (p.getName().contains(keyword) || p.getCategory().contains(keyword)) {

results.add(p);

}

}

return results;

}

}

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

ProductSearch ps=new ProductSearch();

System.out.print("Enter search keyword: ");

String keyword=sc.nextLine();

List<Product> results=ps.search(keyword);

if (results.isEmpty()){

System.out.println("No products found.");

}

else{

System.out.println("Search Results:");

for(Product p:results){

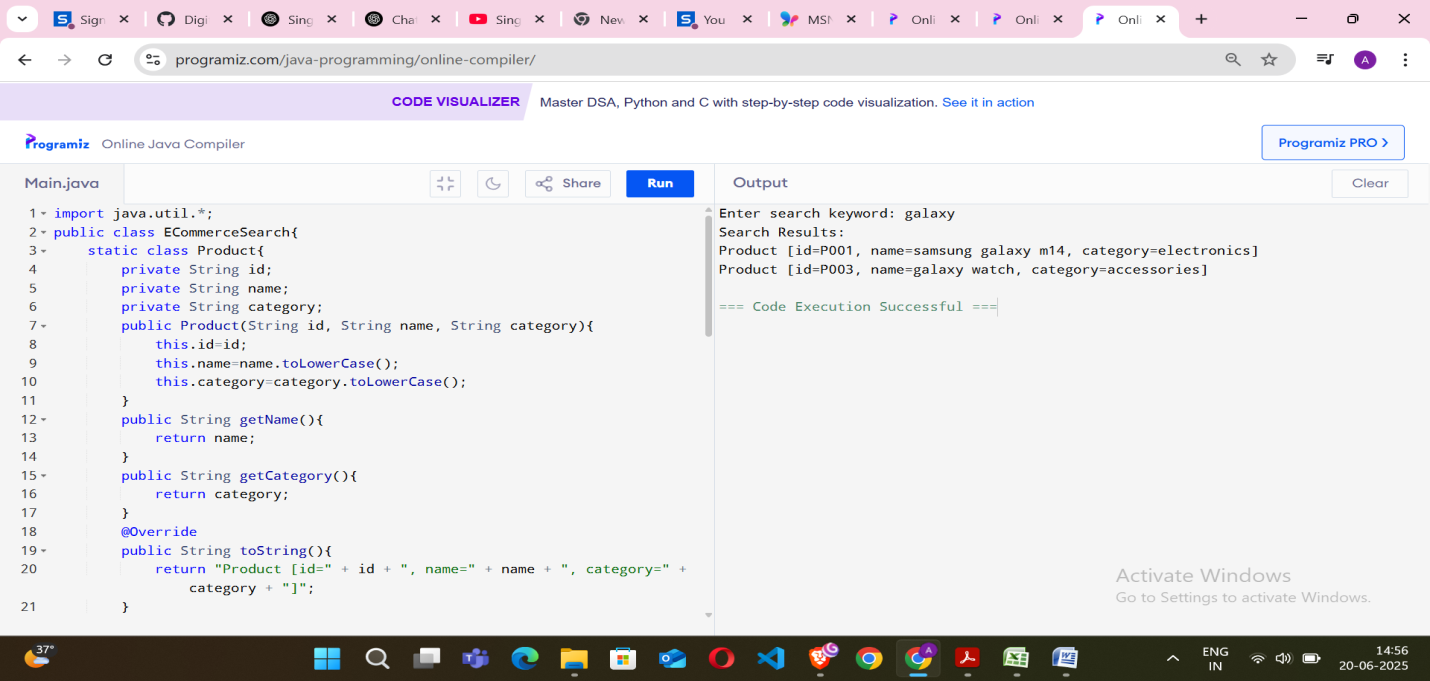
System.out.println(p);

}

}

}

}

O/P:  
  
  
Exercise 2: Financial Forecasting  
  
Code :   
  
import java.util.\*;

public class Main{

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.print("Enter number of months: ");

int n=sc.nextInt();

int[] income=new int[n];

int[] expenses=new int[n];

int[] savings=new int[n];

System.out.println("Enter income for each month:");

for (int i=0;i<n;i++) {

income[i]=sc.nextInt();

}

System.out.println("Enter expenses for each month:");

for (int i=0;i<n;i++) {

expenses[i]=sc.nextInt();

}

int totalSavings=0;

for (int i=0;i<n;i++){

savings[i]=income[i]-expenses[i];

totalSavings+=savings[i];

}

System.out.println("\nMonth-wise Net Savings:");

for(int i=0;i<n;i++){

System.out.printf("Month %d: Rs.%d\n",i+1,savings[i]);

}

System.out.println("Total savings after "+n+ " months: Rs."+totalSavings);

System.out.print("\nEnter target savings: ");

int target=sc.nextInt();

int cumulative=0;

boolean reached=false;

for(int i=0;i<n;i++){

cumulative+=savings[i];

if(cumulative>=target){

System.out.println("Target of Rs."+target+" will be reached in month " + (i+1));

reached=true;

break;

}

}

if(!reached){

System.out.println("Target not reached in the given duration.");

}

}

}  
  
O/P:  
  
