**Exercise 1: Implementing Singleton pattern**

**Code :**

public class President {

private static final int President instance = new President();

private president(){

System.out.println(“President elected”);

}

public static President getInstance(){

return instance;

}

public void speak(){

System.out.println(“im the president”);

}

public static void main(String args[]){

President p1=President.getInstance();

President p2=President.getInstance();

P1.speak();

P2.speak();

If(p1==p2){

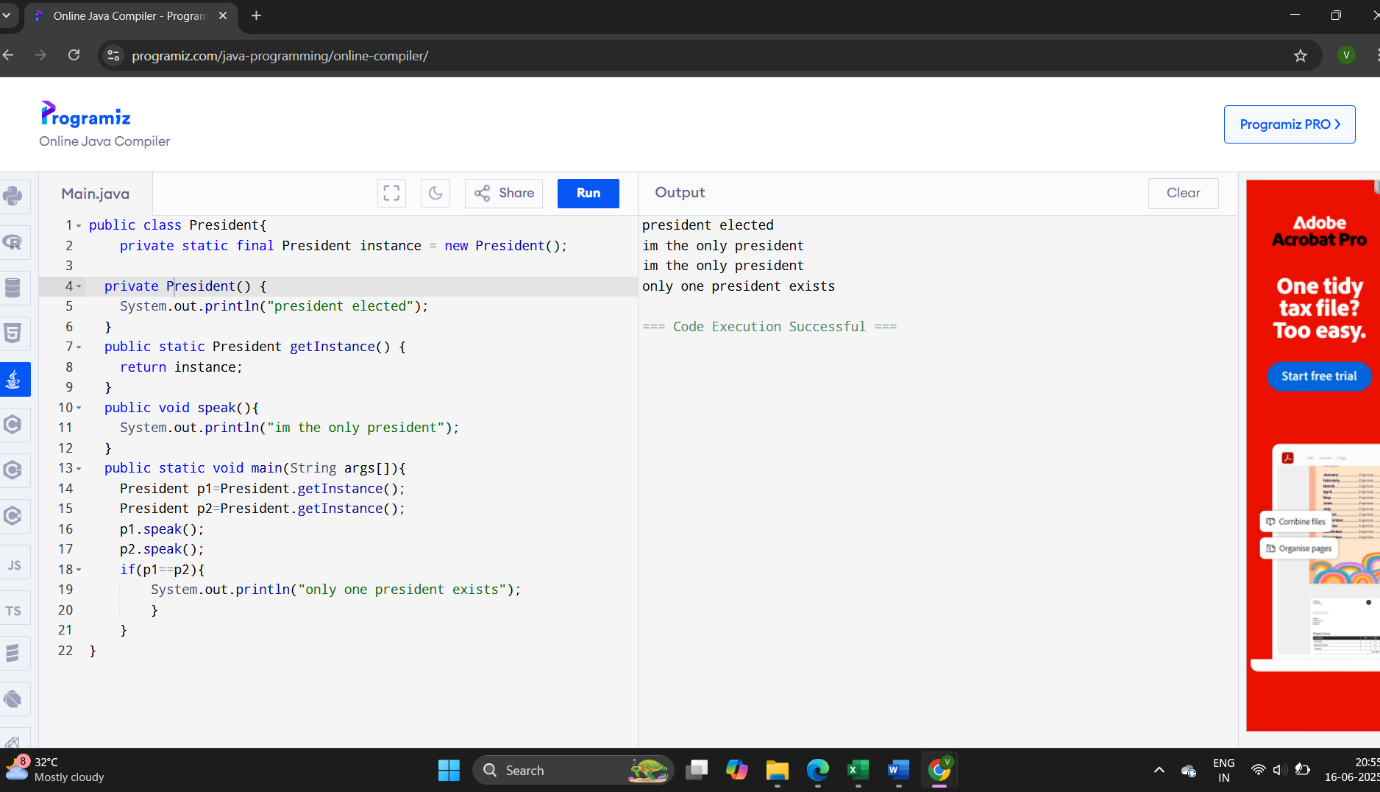
System.out.println(“only one president exists”);

}

}

}

**Output:**



**Exercise 2: Implementing Factory Pattern**

**Code :**

interface Animal{

public void speak();

}

class Cat implements Animal{

public void speak(){

System.out.println("bow bow");

}

}

class Dog implements Animal{

public void speak(){

System.out.println("meow meow");

}

}

class AnimalFactory{

public static Animal getAnimal(String type){

if(type.equalsIgnoreCase("dog")) {

return new Dog();

} else if (type.equalsIgnoreCase("cat")) {

return new Cat();

} else {

return null;

}

}

}

public class Main {

public static void main(String[] args) {

Animal a1 = AnimalFactory.getAnimal("dog");

Animal a2 = AnimalFactory.getAnimal("cat");

a1.speak();

a2.speak();

}

}

**Output:**

