**Week1\_DesignPrinciplesAndPatterns\_HandsOn**

**Exercise 1: Implementing the Singleton Pattern:**

**Code:**

**class** Logger{

**private** **static** Logger *instance*;

**private** Logger()

{

System.***out***.println("Singleton Pattern is achieved");

}

**public** **static** Logger getinstance()

{

**if**(*instance*==**null**)

{

*instance*=**new** Logger();

}

**return** *instance*;

}

}

**public** **class** SingletonPatternExample

{

**public** **static** **void** main(String[] args) {

Logger obj1=Logger.*getinstance*();

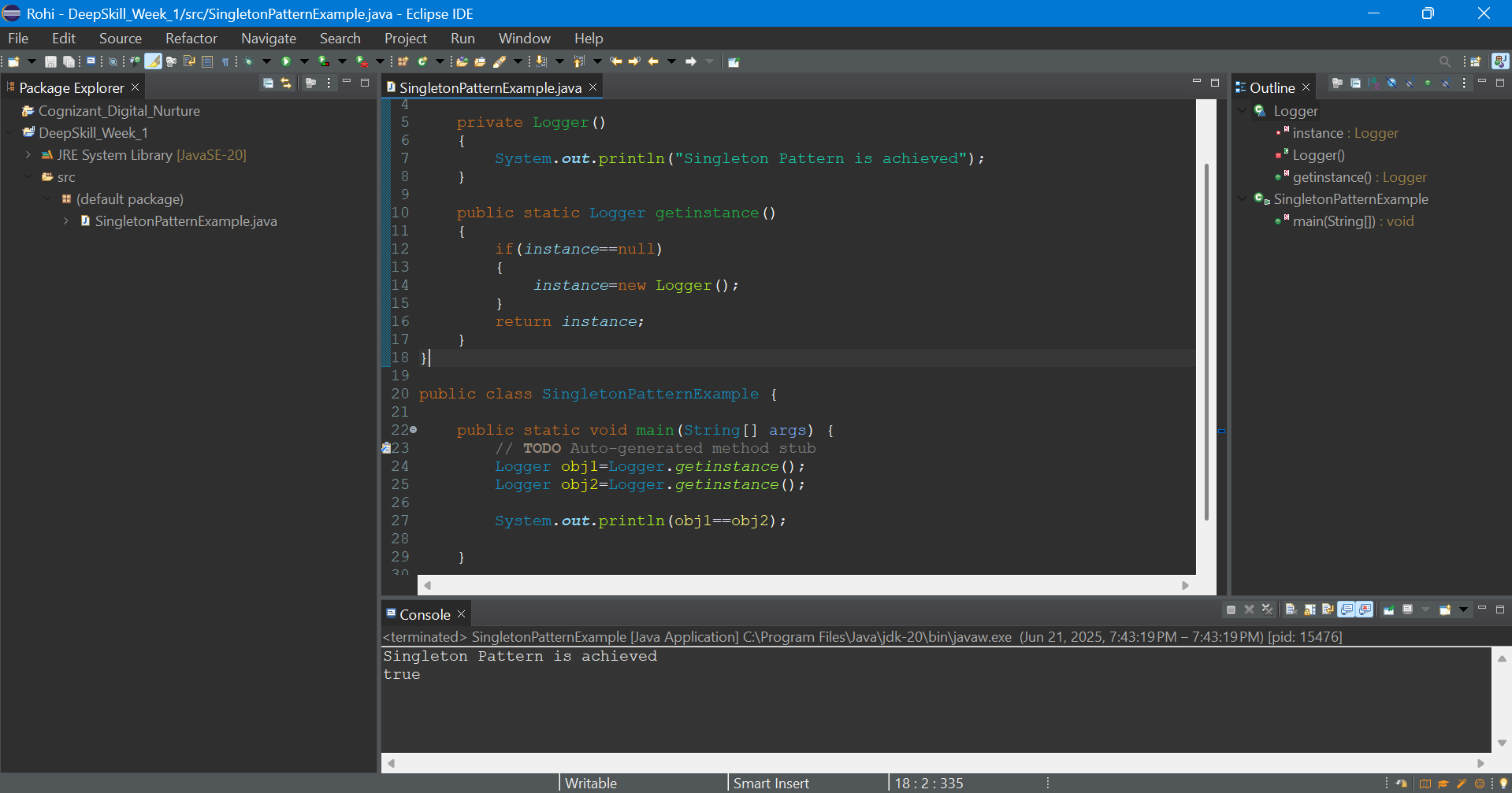
Logger obj2=Logger.*getinstance*();

System.***out***.println(obj1==obj2);

}

}

**Output:**



**Exercise 2: Implementing the Factory Method Pattern :**

**Code:**

**interface** Document {

String create();

}

**class** Word **implements** Document {

**public** String create()

{

**return** "Word is created ";

}

}

**class** Pdf **implements** Document {

**public** String create()

{

**return** "Pdf is created ";

}

}

**class** Excel **implements** Document {

**public** String create()

{

**return** "Excel is created ";

}

}

**abstract** **class** DocumentFactory {

**abstract** Document createDocument();

}

**class** WordFactory **extends** DocumentFactory {

**public** Document createDocument()

{

**return** **new** Word();

}

}

**class** PdfFactory **extends** DocumentFactory {

**public** Document createDocument()

{

**return** **new** Pdf();

}

}

**class** ExcelFactory **extends** DocumentFactory {

**public** Document createDocument()

{

**return** **new** Excel();

}

}

**class** FactoryMethodPatternExample

{

**public** **static** **void** main(String[] args)

{

DocumentFactory factory=**new** WordFactory();

Document obj1=factory.createDocument();

System.***out***.println(obj1.create());

factory=**new** PdfFactory();

Document obj2=factory.createDocument();

System.***out***.println(obj2.create());

factory=**new** ExcelFactory();

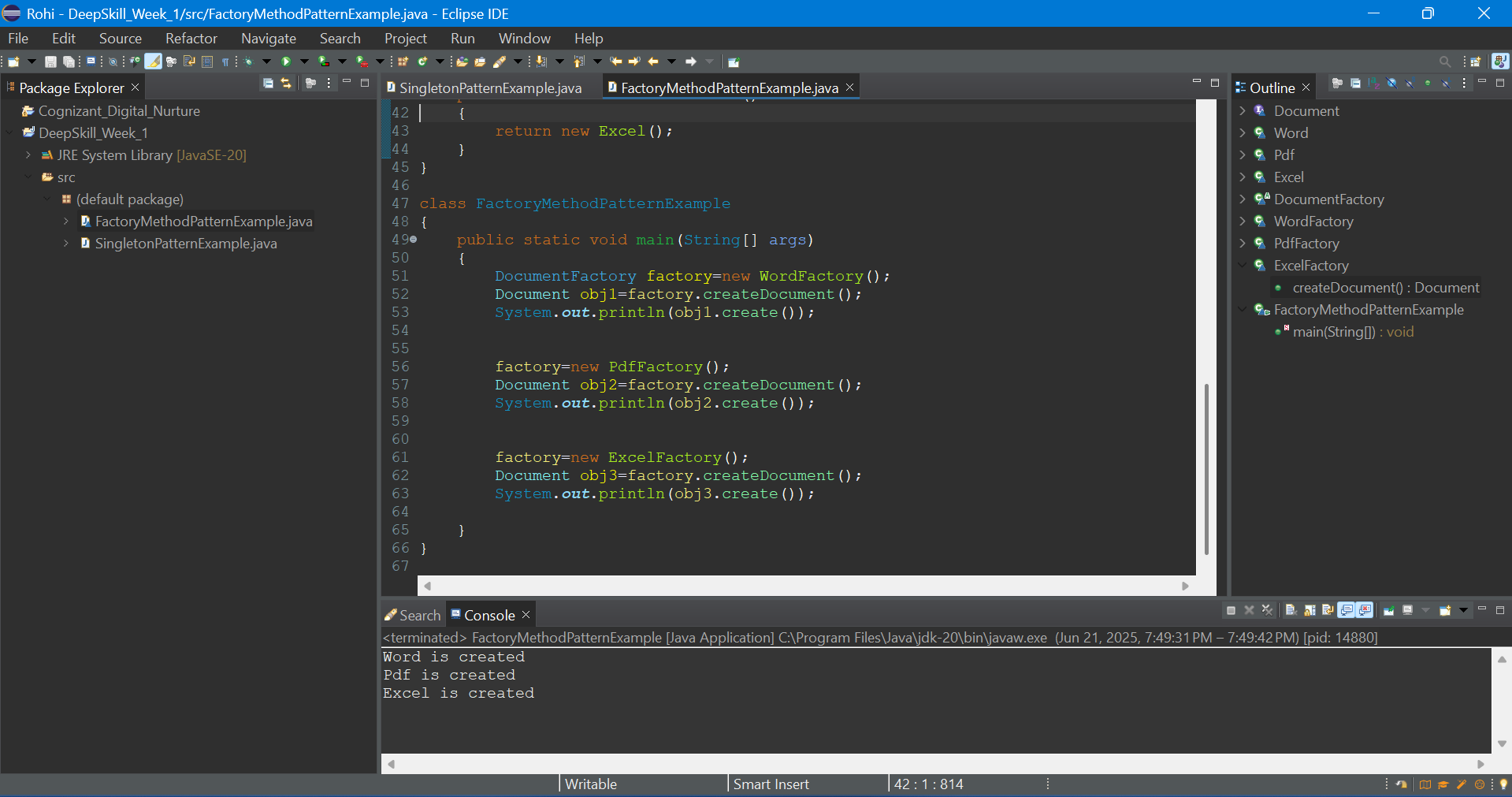
Document obj3=factory.createDocument();

System.***out***.println(obj3.create());

}

}

**Output :**



**Exercise 3: Implementing the Builder Pattern:**

**Code:**

public class BuilderPatternExample {

static class Computer {

private String CPU;

private String RAM;

private String storage;

private String GPU;

private boolean isBluetoothEnabled;

private boolean isWiFiEnabled;

private Computer(Builder builder) {

this.CPU = builder.CPU;

this.RAM = builder.RAM;

this.storage = builder.storage;

this.GPU = builder.GPU;

this.isBluetoothEnabled = builder.isBluetoothEnabled;

this.isWiFiEnabled = builder.isWiFiEnabled;

}

static class Builder {

private String CPU;

private String RAM;

private String storage;

private String GPU;

private boolean isBluetoothEnabled;

private boolean isWiFiEnabled;

public Builder setCPU(String CPU) {

this.CPU = CPU;

return this;

}

public Builder setRAM(String RAM) {

this.RAM = RAM;

return this;

}

public Builder setStorage(String storage) {

this.storage = storage;

return this;

}

public Builder setGPU(String GPU) {

this.GPU = GPU;

return this;

}

public Builder setBluetoothEnabled(boolean isBluetoothEnabled) {

this.isBluetoothEnabled = isBluetoothEnabled;

return this;

}

public Builder setWiFiEnabled(boolean isWiFiEnabled) {

this.isWiFiEnabled = isWiFiEnabled;

return this;

}

public Computer build() {

return new Computer(this);

}

}

@Override

public String toString() {

return "Computer [CPU=" + CPU + ", RAM=" + RAM + ", Storage=" + storage + ", GPU=" + GPU +

", Bluetooth=" + isBluetoothEnabled + ", WiFi=" + isWiFiEnabled + "]";

}

}

public static void main(String[] args) {

Computer gamingPC = new Computer.Builder()

.setCPU("Intel i9")

.setRAM("32GB")

.setStorage("1TB SSD")

.setGPU("RTX 4090")

.setBluetoothEnabled(true)

.setWiFiEnabled(true)

.build();

Computer officePC = new Computer.Builder()

.setCPU("Intel i5")

.setRAM("8GB")

.setStorage("500GB HDD")

.setWiFiEnabled(true)

.build();

System.out.println("Gaming PC: " + gamingPC);

System.out.println("Office PC: " + officePC);

}

}

**Output:**

