Spring Microservices

Pre-requisites

* Core Java – OOPS & Design pattern (Factory pattern)
* Comparator & Lambda expressions
* Java 8 Streams – stream(), forEach(), filter(), collect(), sorted()
* Spring Framework – Dependency Injection & Annotations
* Spring Boot – Webservices & Configurations

Factory Design pattern

It is to abstract object creation at the client side

interface DBOperations {   
 void store();  
 void delete();  
}

class One implements DBOperations { }   
class Two implements DBOperations { }  
class Three implements DBOperations { }

Scenario1: Client creates object – leads to tightly coupled code

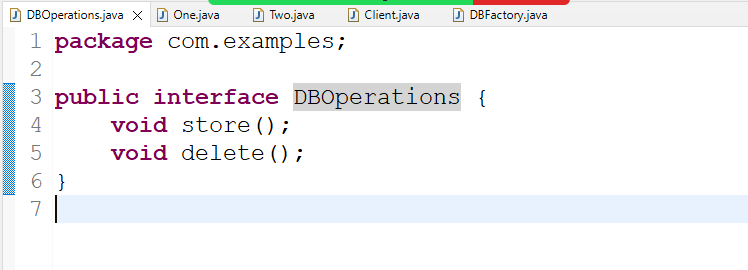
DBOperations db = new One(); // if new implementation must be used, then code must be modified here  
db.store();  
db.delete();

Scenario2: Client doesn’t create object – they use factory pattern to get the object – makes code loosely coupled

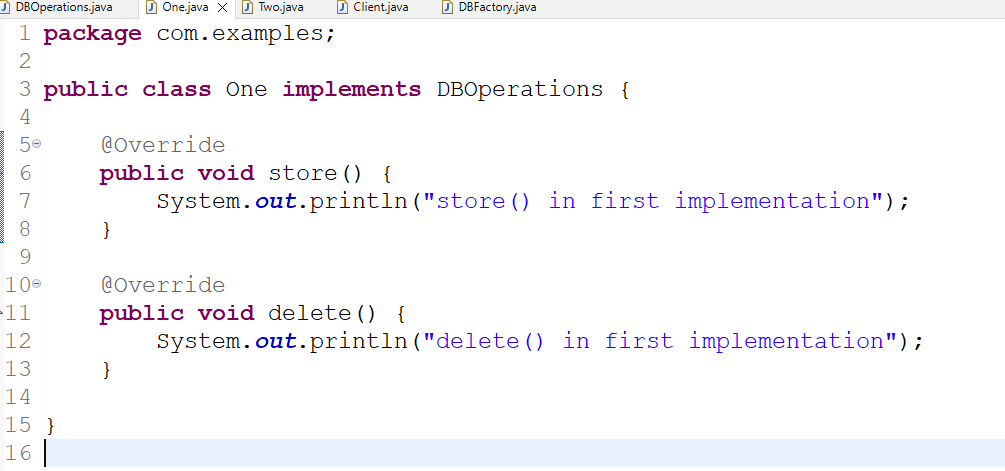
class DBFactory {   
 public static DBOperations getInstance() {   
 return new Two();  
 }  
}

DBOperations db = DBFactory.getInstance(); // client doesn’t know which implementation object is returned.  
db.store();  
db.delete();

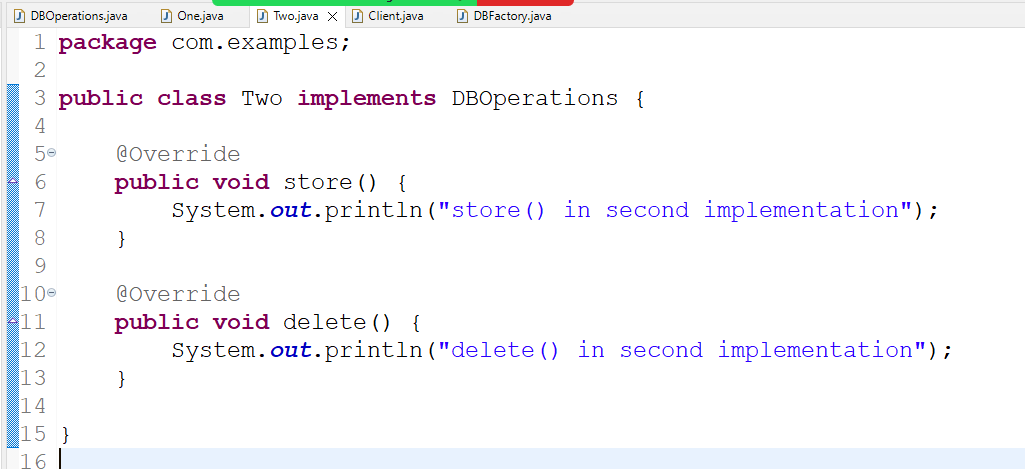
DBOperations.java



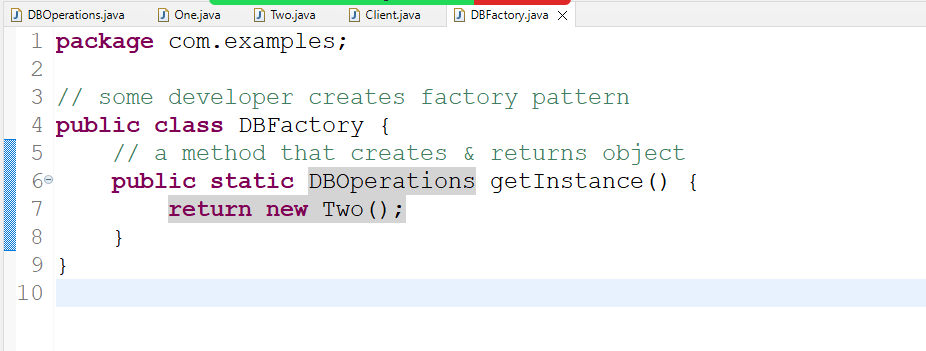
One.java



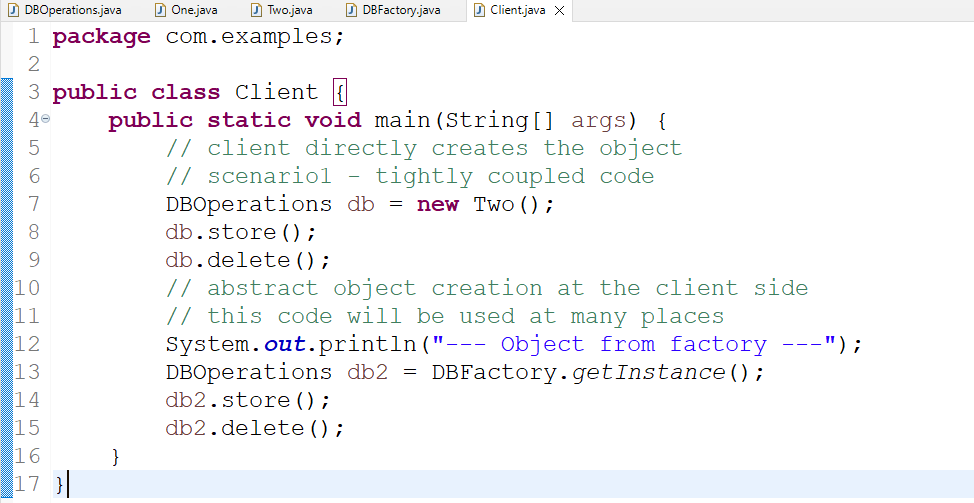
Two.java



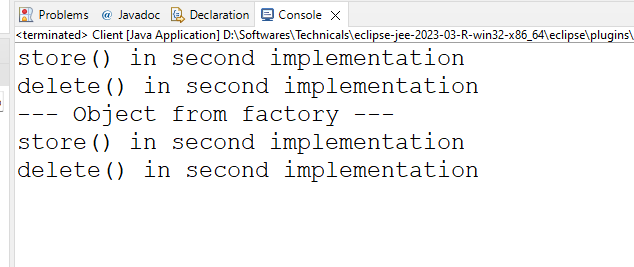
DBFactory.java



Client.java



Output:



Summary:

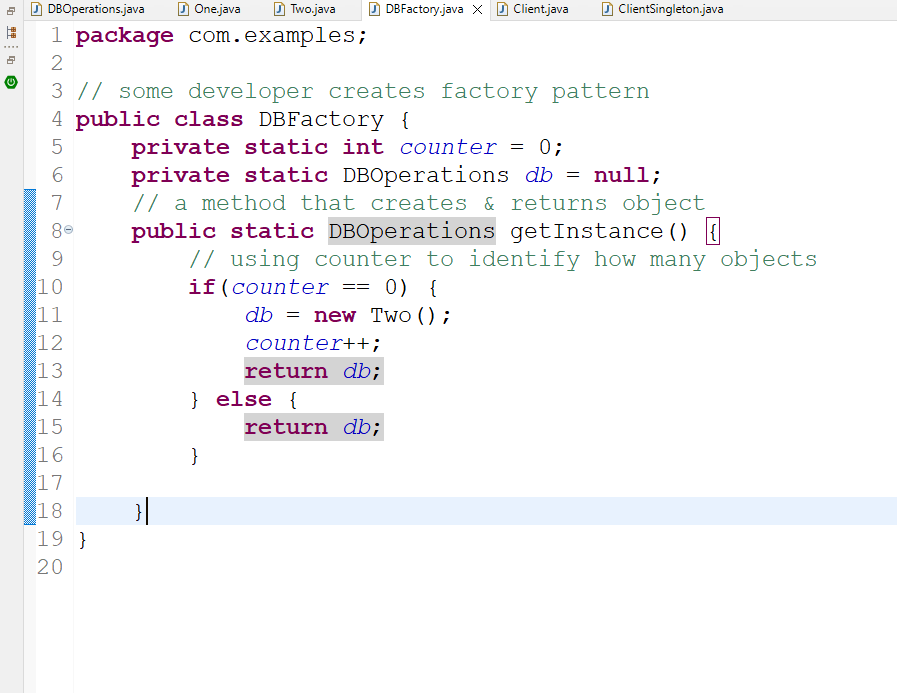
Developers who create object directly need to change their code when a new implementation needs to be used, developers who gets the object from the factory pattern need not to change the code because factory pattern takes care of giving the object.

Factory pattern vs Singleton pattern

Factory pattern just creates the object, however singleton pattern is a factory pattern where on a class only one object will be created even if you call the factory method more than once

Previous factory pattern creates object more than once based on the how many time you call the getInstance() factory method, this leads to more number of object creation, to avoid this we can change the factory method to give only one object regardless of how many times you call the factory method

Modifying the factory to return a singleton object



ClientSingleton.java

