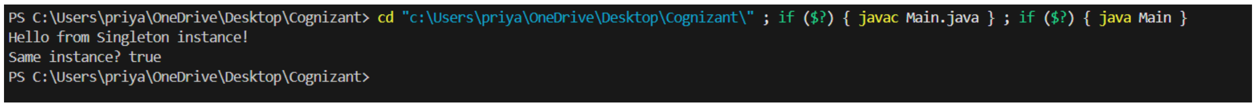
WEEK 1

Exersice 1—

```
class Singleton {


   private Singleton() {


      if (SingletonHelper.INSTANCE != null) {
         throw new IllegalStateException("Singleton already initialized");
      }
   }



   private static class SingletonHelper {
      private static final Singleton INSTANCE = new Singleton();
   }



   public static Singleton getInstance() {
      return SingletonHelper.INSTANCE;
   }



   public void showMessage() {
      System.out.println("Hello from Singleton instance!");
   }
}
public class Main {
    public static void main(String[] args) {

        Singleton singleton = Singleton.getInstance();



        singleton.showMessage();




        Singleton anotherInstance = Singleton.getInstance();
        System.out.println("Same instance? " + (singleton == anotherInstance)); // true
    }
}
```

Output-

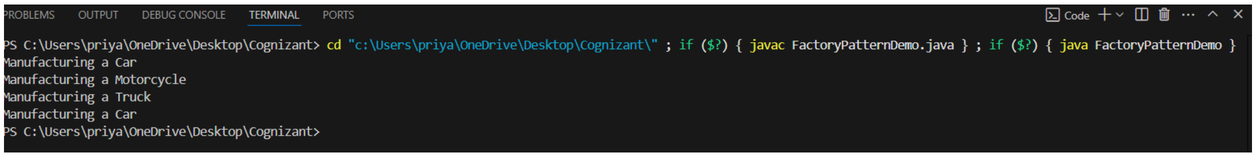


```
PS C:\Users\priya\OneDrive\Desktop\Cognizant> cd "c:\Users\priya\OneDrive\Desktop\Cognizant\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Hello from Singleton instance!
Same instance? true
PS C:\Users\priya\OneDrive\Desktop\Cognizant>
```

Exersice 2—

```
interface Vehicle {
    void manufacture();
}


class Car implements Vehicle {
    @Override
    public void manufacture() {
        System.out.println("Manufacturing a Car");
    }
}

class Motorcycle implements Vehicle {
    @Override
    public void manufacture() {
        System.out.println("Manufacturing a Motorcycle");
    }
}

class Truck implements Vehicle {
    @Override
    public void manufacture() {
        System.out.println("Manufacturing a Truck");
    }
}

abstract class VehicleFactory {

    public abstract Vehicle createVehicle(String type);

    public void manufactureVehicle(String type) {
        Vehicle vehicle = createVehicle(type);
        vehicle.manufacture();
    }
```

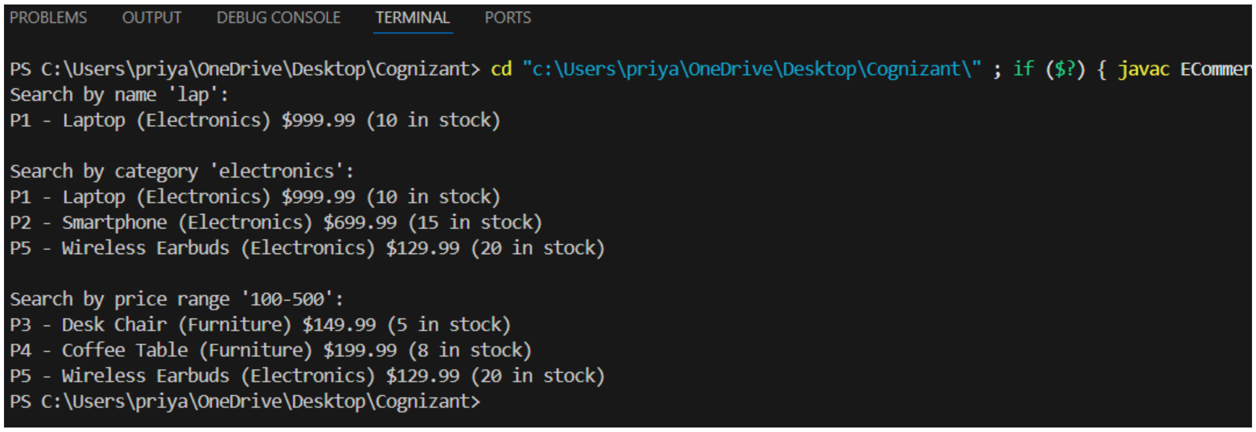
```
}


class ConcreteVehicleFactory extends VehicleFactory {
    @Override
    public Vehicle createVehicle(String type) {
        if (type.equalsIgnoreCase("car")) {
            return new Car();
        } else if (type.equalsIgnoreCase("motorcycle")) {
            return new Motorcycle();
        } else if (type.equalsIgnoreCase("truck")) {
            return new Truck();
        }
        throw new IllegalArgumentException("Unknown vehicle type: " + type);
    }
}
public class FactoryPatternDemo {
    public static void main(String[] args) {
        VehicleFactory factory = new ConcreteVehicleFactory();


        factory.manufactureVehicle("car");
        factory.manufactureVehicle("motorcycle");
        factory.manufactureVehicle("truck");


        Vehicle car = factory.createVehicle("car");
        car.manufacture();
    }
}
```



```
PS C:\Users\priya\OneDrive\Desktop\Cognizant> cd "c:\Users\priya\OneDrive\Desktop\Cognizant\" ; if ($?) { javac FactoryPatternDemo.java } ; if ($?) { java FactoryPatternDemo }
Manufacturing a Car
Manufacturing a Motorcycle
Manufacturing a Truck
Manufacturing a Car
PS C:\Users\priya\OneDrive\Desktop\Cognizant>
```

Exersice 3—

```
import java.util.List;
import java.util.ArrayList;
import java.util.stream.Collectors;

// Product class
class Product {
    private String id;
    private String name;
    private String category;
    private double price;
    private int stock;

    public Product(String id, String name, String category, double price, int stock) {
        this.id = id;
        this.name = name;
        this.category = category;
        this.price = price;
        this.stock = stock;
    }

    // Getters
    public String getId() { return id; }
    public String getName() { return name; }
    public String getCategory() { return category; }
    public double getPrice() { return price; }
    public int getStock() { return stock; }

    @Override
    public String toString() {
        return String.format("%s - %s (%s) $%.2f (%d in stock)",
                id, name, category, price, stock);
    }
}

// Search Strategy interface
interface SearchStrategy {
    List<Product> search(List<Product> products, String query);
}

// Concrete search strategies
class NameSearchStrategy implements SearchStrategy {
    @Override
    public List<Product> search(List<Product> products, String query) {
        return products.stream()
            .filter(p -> p.getName().toLowerCase().contains(query.toLowerCase()))
            .collect(Collectors.toList());
    }
}

class CategorySearchStrategy implements SearchStrategy {
    @Override
    public List<Product> search(List<Product> products, String query) {
        return products.stream()
            .filter(p -> p.getCategory().equalsIgnoreCase(query))
            .collect(Collectors.toList());
    }
}

class PriceRangeSearchStrategy implements SearchStrategy {
    @Override
    public List<Product> search(List<Product> products, String query) {
        try {
            String[] range = query.split("-");
            double min = Double.parseDouble(range[0].trim());
            double max = Double.parseDouble(range[1].trim());
            return products.stream()
                .filter(p -> p.getPrice() >= min && p.getPrice() <= max)
                .collect(Collectors.toList());
        } catch (Exception e) {
            return new ArrayList<>();
        }
    }
}

// Search Strategy Factory
class SearchStrategyFactory {
    public SearchStrategy createStrategy(String searchType) {
        switch (searchType.toLowerCase()) {
            case "name":
                return new NameSearchStrategy();
            case "category":
                return new CategorySearchStrategy();
            case "price":
                return new PriceRangeSearchStrategy();
            default:
                throw new IllegalArgumentException("Unknown search type: " + searchType);
        }
    }
}

// E-commerce Platform with Search Functionality
class ECommercePlatform {
    private List<Product> products;
    private SearchStrategyFactory strategyFactory;
    public ECommercePlatform() {
        this.products = new ArrayList<>();
        this.strategyFactory = new SearchStrategyFactory();
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public List<Product> search(String searchType, String query) {
        SearchStrategy strategy = strategyFactory.createStrategy(searchType);
        return strategy.search(products, query);
    }
}
public class ECommerceDemo {
    public static void main(String[] args) {
        // Create platform and add products
        ECommercePlatform platform = new ECommercePlatform();
        platform.addProduct(new Product("P1", "Laptop", "Electronics", 999.99, 10));
        platform.addProduct(new Product("P2", "Smartphone", "Electronics", 699.99, 15));
        platform.addProduct(new Product("P3", "Desk Chair", "Furniture", 149.99, 5));
        platform.addProduct(new Product("P4", "Coffee Table", "Furniture", 199.99, 8));
        platform.addProduct(new Product("P5", "Wireless Earbuds", "Electronics", 129.99, 20));

        // Perform searches
        System.out.println("Search by name 'lap':");
        platform.search("name", "lap").forEach(System.out::println);


        System.out.println("\nSearch by category 'electronics':");
        platform.search("category", "electronics").forEach(System.out::println);


        System.out.println("\nSearch by price range '100-500':");
        platform.search("price", "100-500").forEach(System.out::println);
    }
}
```



```
PS C:\Users\priya\OneDrive\Desktop\Cognizant> cd "c:\Users\priya\OneDrive\Desktop\Cognizant\" ; if ($?) { javac ECommer
Search by name 'lap':
P1 - Laptop (Electronics) $999.99 (10 in stock)

Search by category 'electronics':
P1 - Laptop (Electronics) $999.99 (10 in stock)
P2 - Smartphone (Electronics) $699.99 (15 in stock)
P5 - Wireless Earbuds (Electronics) $129.99 (20 in stock)

Search by price range '100-500':
P3 - Desk Chair (Furniture) $149.99 (5 in stock)
P4 - Coffee Table (Furniture) $199.99 (8 in stock)
P5 - Wireless Earbuds (Electronics) $129.99 (20 in stock)
PS C:\Users\priya\OneDrive\Desktop\Cognizant>
```