	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 2**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Rhythm Shah **Roll no:** 22BCP071			

Objective: To familiarize students with standard Creational design patterns.
Experiment: Explain the abstract factory design pattern and write a program using any object-oriented programming language to demonstrate the working of abstract factory design pattern.

Theory:

Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. In this method we create an object step by step and at last through all objects and method we build a object.

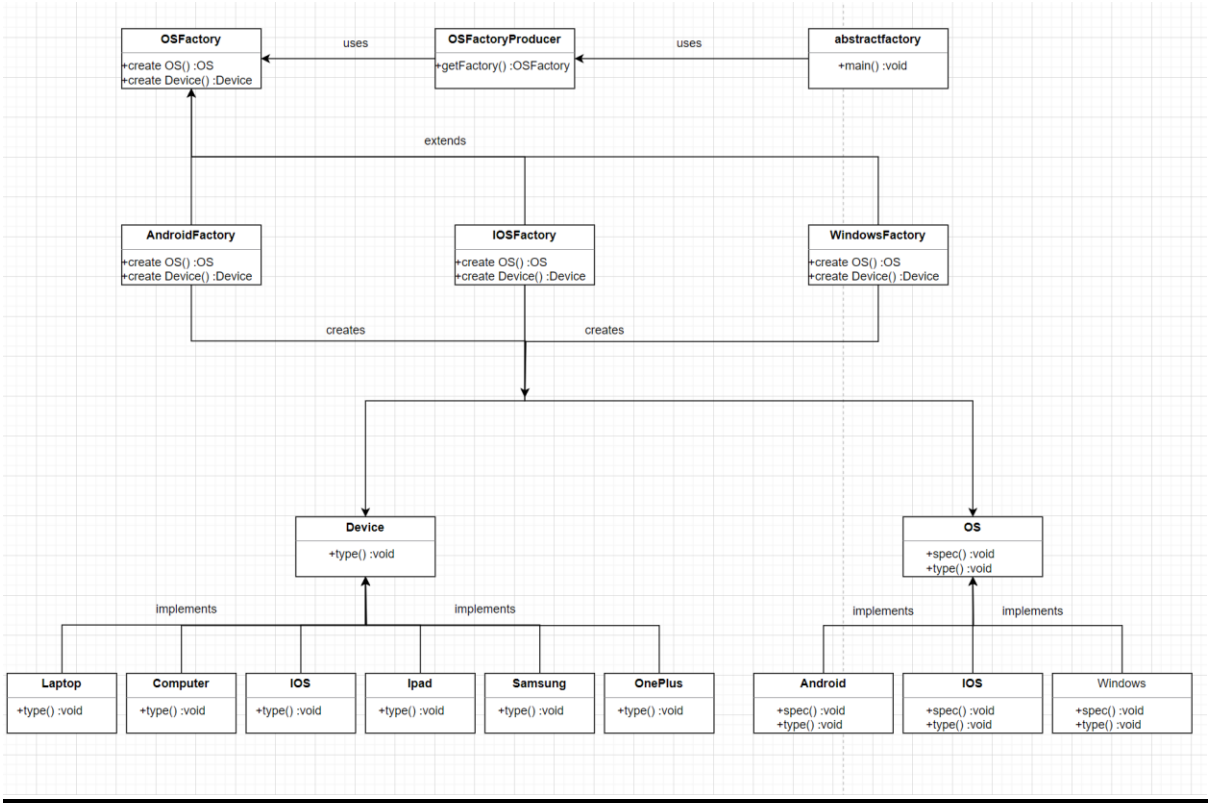
In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

Example (explanation of my implementation):

I created two interfaces(OS and Device) and concrete classes implementing it. We create an abstract factory class OSFactory as next step. Factory classes AndroidFactory,IOS Factory and Windows Factory defined, which extends OSFactory. A factory creator/generator class OSFactoryProducer is created.

abstractfactory, our demo class uses OSFactoryProducer to get a OSFactoryobject. It will pass information (Android/ IOS/ WINDOWS for OS) and (Samsung/OnePlus) for Android , (Ipad/IPhone) for ios and (Laptop/Computer) for windows to OSFactory to get the type of object it needs.

UML Diagram:



Code:

```
import java.util.Scanner;

interface OS {
   void spec();
   void type();
}

class Android implements OS {
   public void spec() {
      System.out.println("Most powerful OS");
   }
```

```
    public void type() {
        System.out.println("The latest Android version is 14");
    }
}

class IOS implements OS {
    public void spec() {
        System.out.println("Most secure OS");
    }

    public void type() {
        System.out.println("The latest iPhone device is iPhone 15");
    }
}

class Windows implements OS {
    public void spec() {
        System.out.println("I am about to die");
    }

    public void type() {
        System.out.println("The latest generation is i7-13th Gen");
    }
}

interface Device {
    void type();
}

class Samsung implements Device {
    public void type() {
        System.out.println("The device used is Samsung S23");
    }
}

class OnePlus implements Device {
    public void type() {
        System.out.println("The device used is OnePlusNordCE3");
    }
}

class IPhone implements Device {
    public void type() {
        System.out.println("The device used is iPhone13");
    }
}

class IPad implements Device {
    public void type() {
        System.out.println("The device used is iPad air2");
```

```
    }
}

class Laptop implements Device {
    public void type() {
        System.out.println("The device used is Lenovo Ideapad");
    }
}

class Computer implements Device {
    public void type() {
        System.out.println("The device used is DELL");
    }
}

abstract class OSFactory {
    abstract OS createOS();

    abstract Device createDevice();
}

class AndroidFactory extends OSFactory {
    OS createOS() {
        return new Android();
    }

    Device createDevice() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter 'samsung' or 'oneplus' for Android device: ");
        String userInputDevice = scanner.nextLine().toLowerCase();
        scanner.close();
        if (userInputDevice.equals("samsung")) {
            return new Samsung();
        } else if (userInputDevice.equals("oneplus")) {
            return new OnePlus();
        } else {
            System.out.println("Invalid Android device input.");
            return null;
        }
    }
}

class IOSFactory extends OSFactory {
    OS createOS() {
        return new IOS();
    }

    Device createDevice() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter 'iphone' or 'ipad' for iOS device: ");
```

```
        String userInputDevice = scanner.nextLine().toLowerCase();
        scanner.close();
        if (userInputDevice.equals("iphone")) {
            return new IPhone();
        } else if (userInputDevice.equals("ipad")) {
            return new IPad();
        } else {
            System.out.println("Invalid iOS device input.");
            return null;
        }
    }
}

class WindowsFactory extends OSFactory {
    OS createOS() {
        return new Windows();
    }

    Device createDevice() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter 'laptop' or 'computer' for Windows device: ");
        String userInputDevice = scanner.nextLine().toLowerCase();
        scanner.close();
        if (userInputDevice.equals("laptop")) {
            return new Laptop();
        } else if (userInputDevice.equals("computer")) {
            return new Computer();
        } else {
            System.out.println("Invalid Windows device input.");
            return null;
        }
    }

}

class OSFactoryProducer {
    public static OSFactory getFactory(String userInputOS) {
        if (userInputOS.equals("android")) {
            return new AndroidFactory();
        } else if (userInputOS.equals("ios")) {
            return new IOSFactory();
        } else if (userInputOS.equals("windows")) {
            return new WindowsFactory();
        } else {
            System.out.println("Invalid operating system input.");
            return null;
        }
    }
}
```

```
public class abstractfactory {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter 'android', 'ios', or 'windows' for operating system: ");
        String userInputOS = scanner.nextLine().toLowerCase();

        OSFactory osFactory = OSFactoryProducer.getFactory(userInputOS);

        if (osFactory != null) {
            OS osObj = osFactory.createOS();
            Device deviceObj = osFactory.createDevice();

            if (deviceObj != null) {
                osObj.spec();
                osObj.type();
                deviceObj.type();
            }
        }

        scanner.close();
    }
}
```

Output:

```
Enter 'android', 'ios', or 'windows' for operating system: android
Enter 'samsung' or 'oneplus' for Android device: oneplus
Most powerful OS
The latest Android version is 14
The device used is OnePlusNordCE3
PS E:\Fourth sem\Design pattern lab>
```