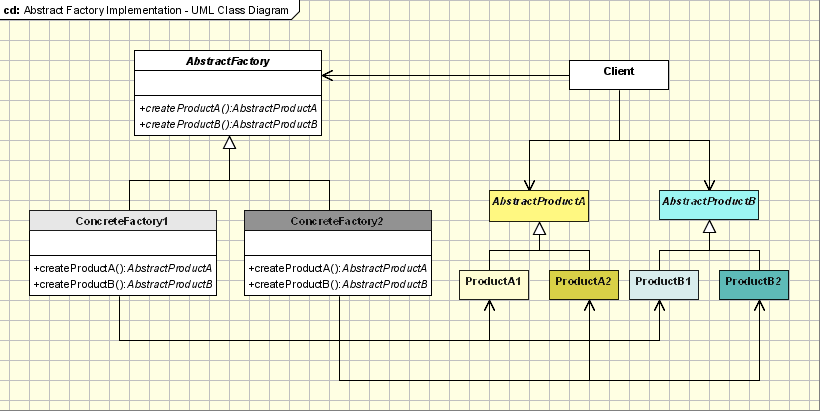
Abstract Factory

GOF : Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

# **Class Diagram**



# **Java Code**

abstract class AbstractProductB{

//public abstract void operationB1();

//public abstract void operationB2();

}

abstract class AbstractProductA {

public abstract void operationA1();

public abstract void operationA2();

}

class ProductA2 extends AbstractProductA {

ProductA2(String arg) {

System.out.println("Hello "+arg);

} // Implement the code here

public void operationA1() { };

public void operationA2() { };

}

class ProductA1 extends AbstractProductA {

ProductA1(String arg){

System.out.println("Hello "+arg);

} // Implement the code here

public void operationA1() { };

public void operationA2() { };

}

abstract class AbstractFactory{

abstract AbstractProductA createProductA();

abstract AbstractProductB createProductB();

}

class ConcreteFactory2 extends AbstractFactory {

AbstractProductA createProductA() {

return new ProductA2("ProductA2");

}

AbstractProductB createProductB(){

return new ProductB2("ProductB2");

}

}

class ConcreteFactory1 extends AbstractFactory {

AbstractProductA createProductA() {

return new ProductA1("ProductA1");

}

AbstractProductB createProductB() {

return new ProductB1("ProductB1");

}

}

class ProductB2 extends AbstractProductB {

ProductB2(String arg) {

System.out.println("Hello "+arg);

} // Implement the code here

}

class ProductB1 extends AbstractProductB {

ProductB1(String arg) {

System.out.println("Hello "+arg);

} // Implement the code here

}

//Factory creator - an indirect way of instantiating the factories

class FactoryMaker{

private static AbstractFactory pf=null;

static AbstractFactory getFactory(String choice) {

if(choice.equals("a")) {

pf=new ConcreteFactory1();

}else if(choice.equals("b")) {

pf=new ConcreteFactory2();

} return pf;

}

}

// Client

public class Client {

public static void main(String args[]) {

AbstractFactory pf=FactoryMaker.getFactory("a");

AbstractProductA product=pf.createProductA();

//more function calls on product

}

}