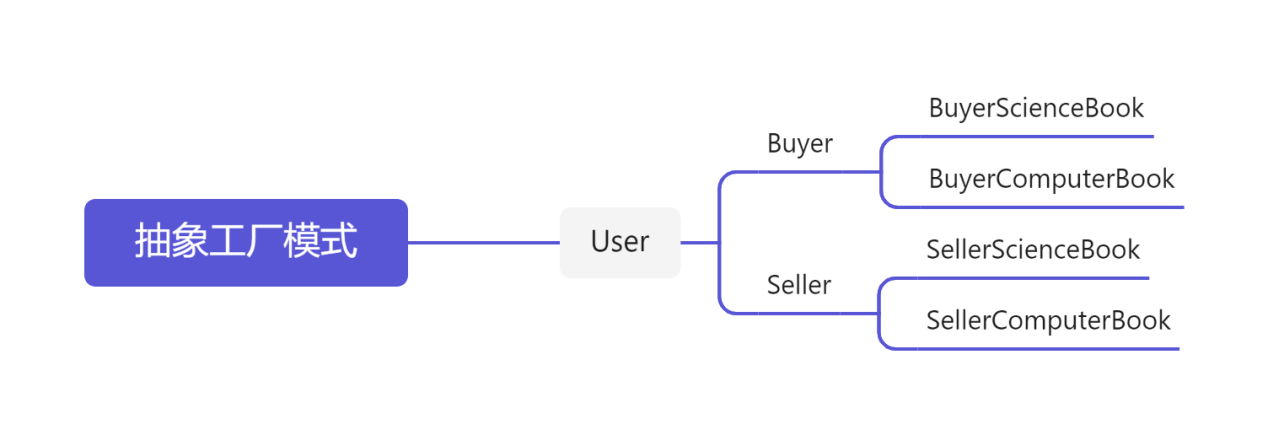
**设计模式：抽象工厂模式**

****

**类实现：**

//定义抽象工厂类,定义具体工厂的公共接口：上传求书信息+选择图书

abstract class User{

    public abstract Book OperateScienceBook();//上传信息

    public abstract Book OperateComputerBook();//选择图书

}

//创建抽象产品族类，定义具体产品的公共接口

abstract class AbstractBook{

    public abstract void Show();

    public abstract void Select();

}

//抽象产品类 定义公共接口

abstract class ScienceBook extends AbstractBook{

    public abstract void Show();

    public abstract void Select();

}

abstract class ComputerBook extends AbstractBook{

    public abstract void Show();

    public abstract void Select();

}

//具体产品类继承抽象产品类，定义生产的具体产品

class BuyerScienceBook extends ScienceBook{

    public void Select(){

        System.out.println("选好科学类图书");

    }

}

class BuyerComputerBook extends ComputerBook{

    public void Select(){

        System.out.println("选好计算机类图书");

    }

}

class SellerScienceBook extends ScienceBook{

    public void Show(){

        System.out.println("上传科学类图书的具体信息");

    }

}

class SellerComputerBook extends ComputerBook{

    public void Show(){

        System.out.println("上传计算机类图书的具体信息");

    }

}

//具体工厂继承抽象工厂类，定义创建对应具体产品实例的方法

class Buyer extends User{

    public Book OperateScienceBook(){

        return new BuyerScienceBook();

    }

    public Book OperateComputerBook(){

        return new BuyerComputerBook();

    }

}

class Seller extends User{

    public Book OperateScienceBook(){

        return new SellerScienceBook();

    }

    public Book OperateComputerBook(){

        return new SellerComputerBook();

    }

}

//客户端通过实例化具体的工厂类，并调用其创建不同目标产品类的实例

public class AbstractProductPattern{

    public static void main(string[] args){

        Buyer mBuyer = new Buyer();

        Seller mSeller = new Seller();

        mBuyer.OperateScienceBook().Select();

        mBuyer.OperateComputerBook().Select();

        mSeller.OperateScienceBook().Show();

        mSeller.OperateComputerBook().Show();

    }

}