Java CD出租销售店作业

2013599 田佳业

一、设计目标:

假设你在业余时间经营一个会员制的CD出租销售店，需要一个管理程序。  
完成功能：  
1.增加、删除会员  
2.出租、销售CD  
3.进货、统计

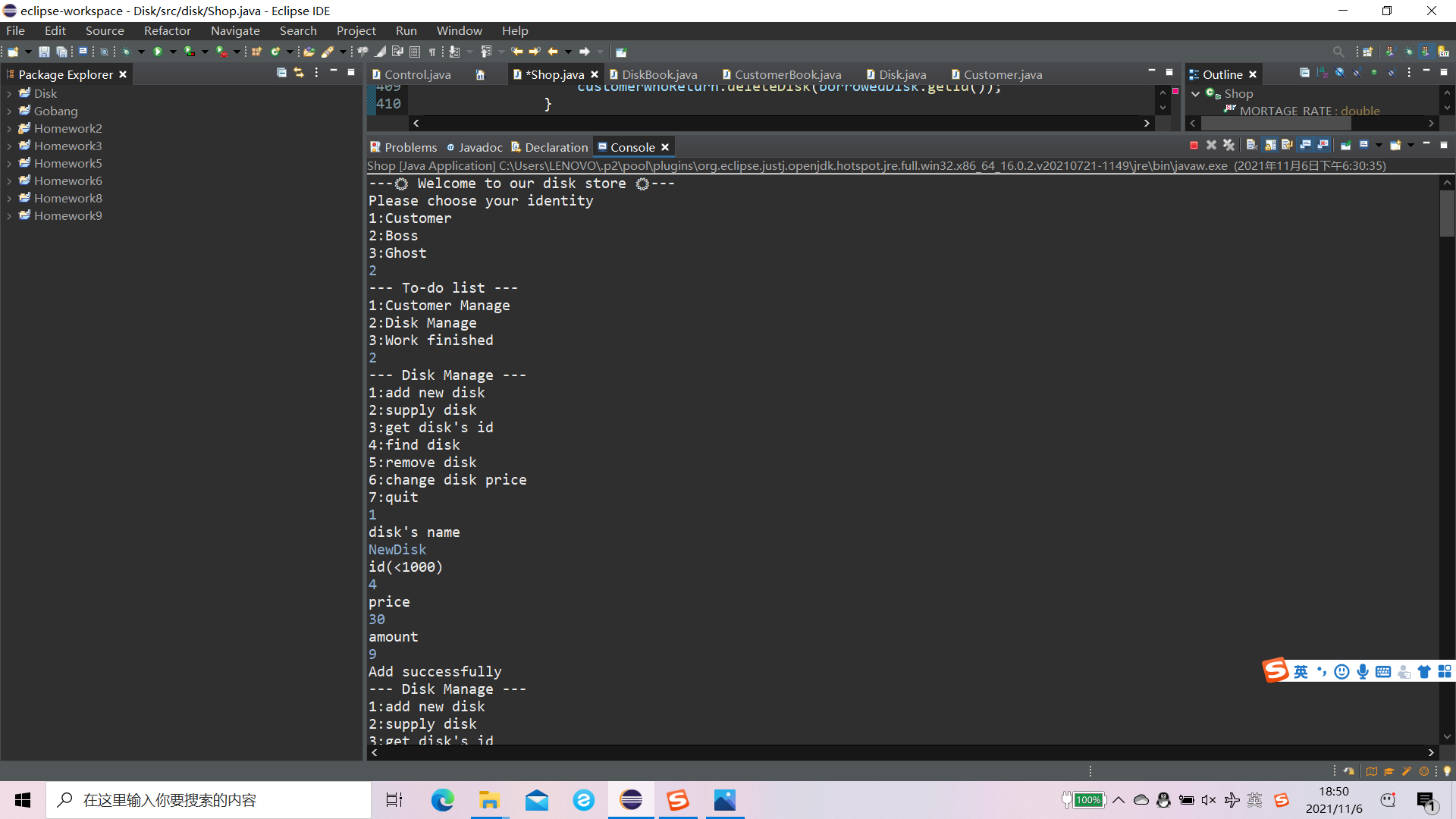
二、程序亮点：  
1.具备输入检查及完备的特殊情况处理 ，有较强的健壮性。

2.优化了面向用户的流程，指引清晰，界面整洁。

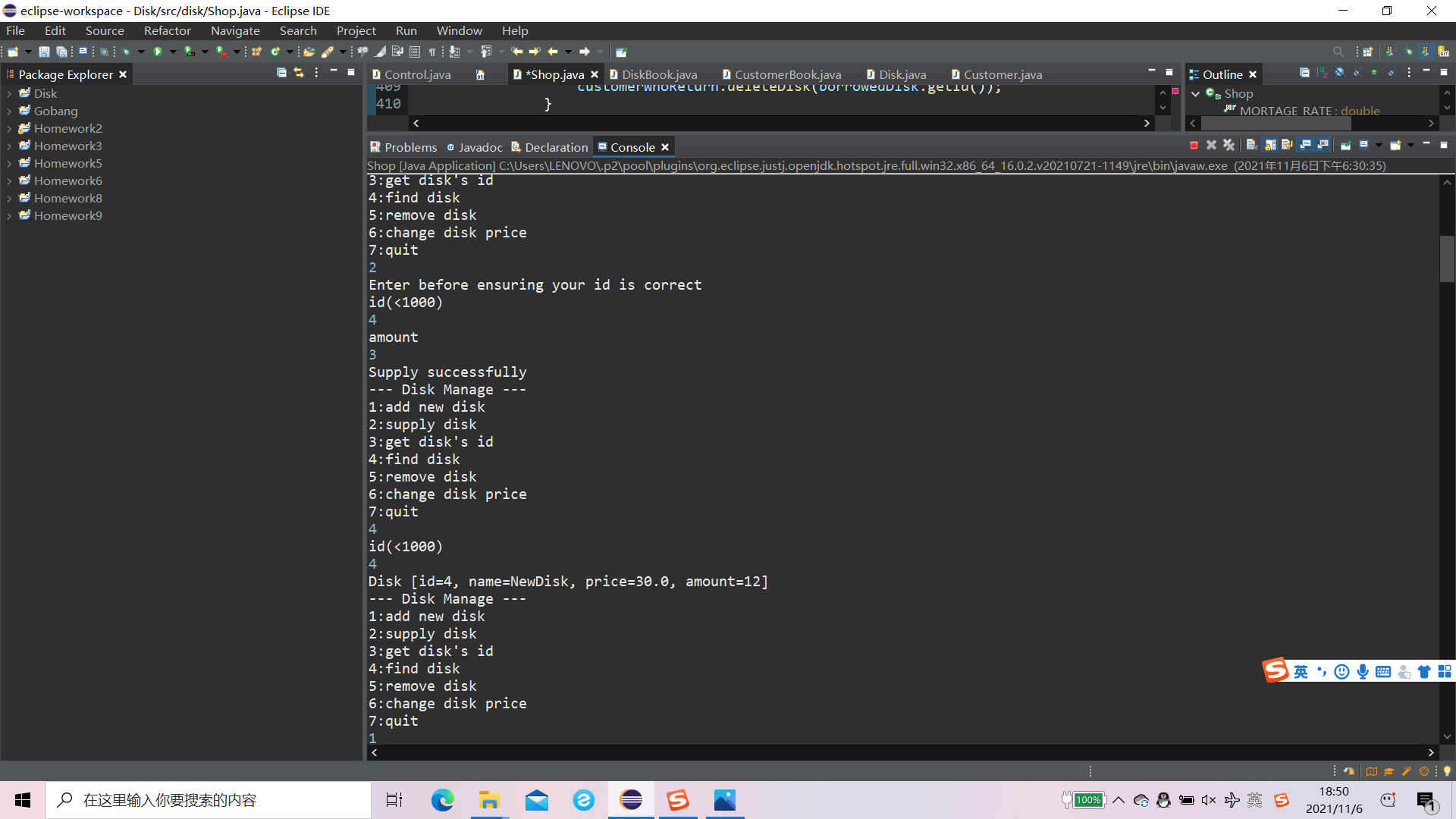
3.代码功能模块明确，可读性强。

三、运行实例:

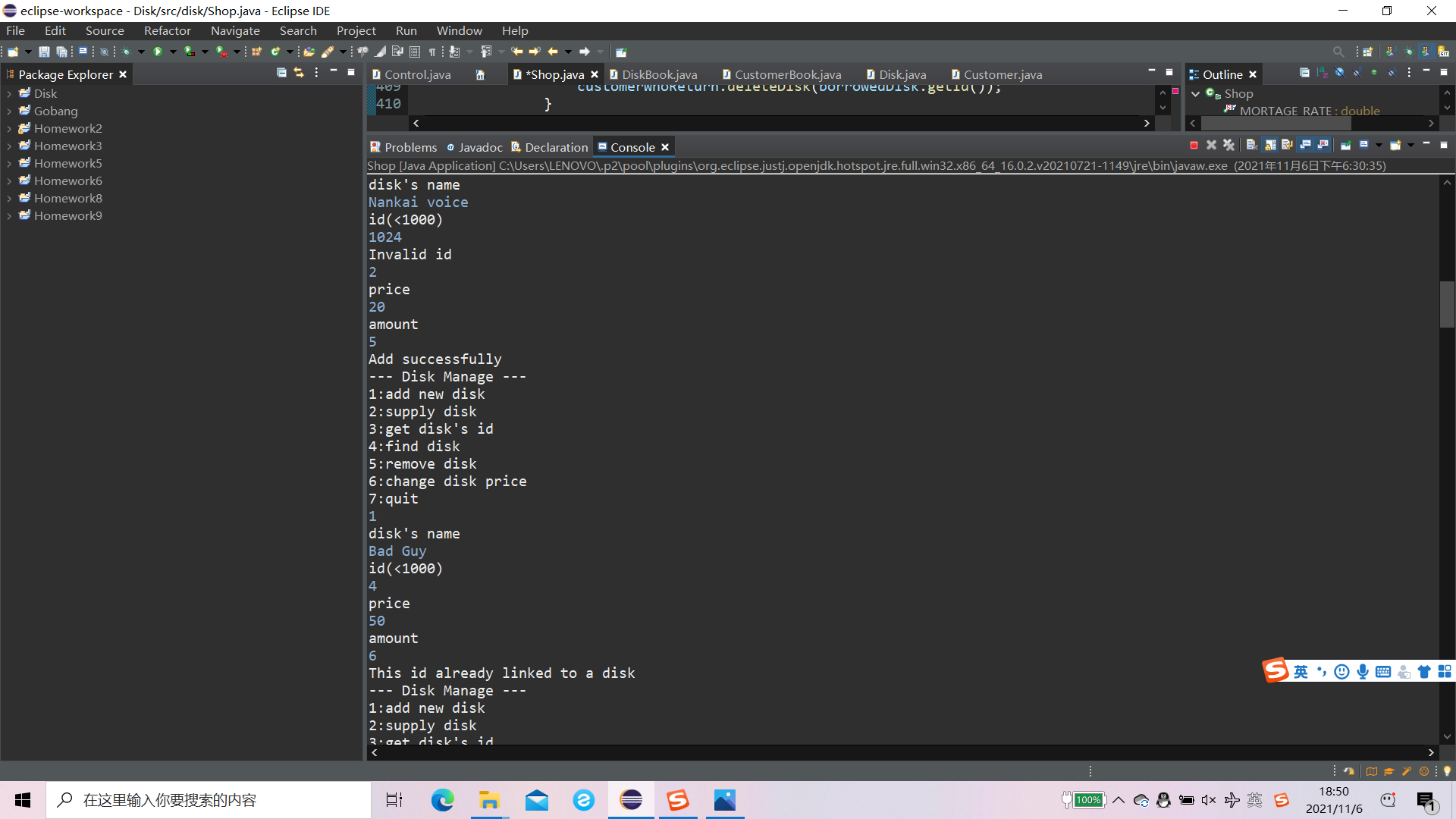
（以连续进行的一次测试作为实例）



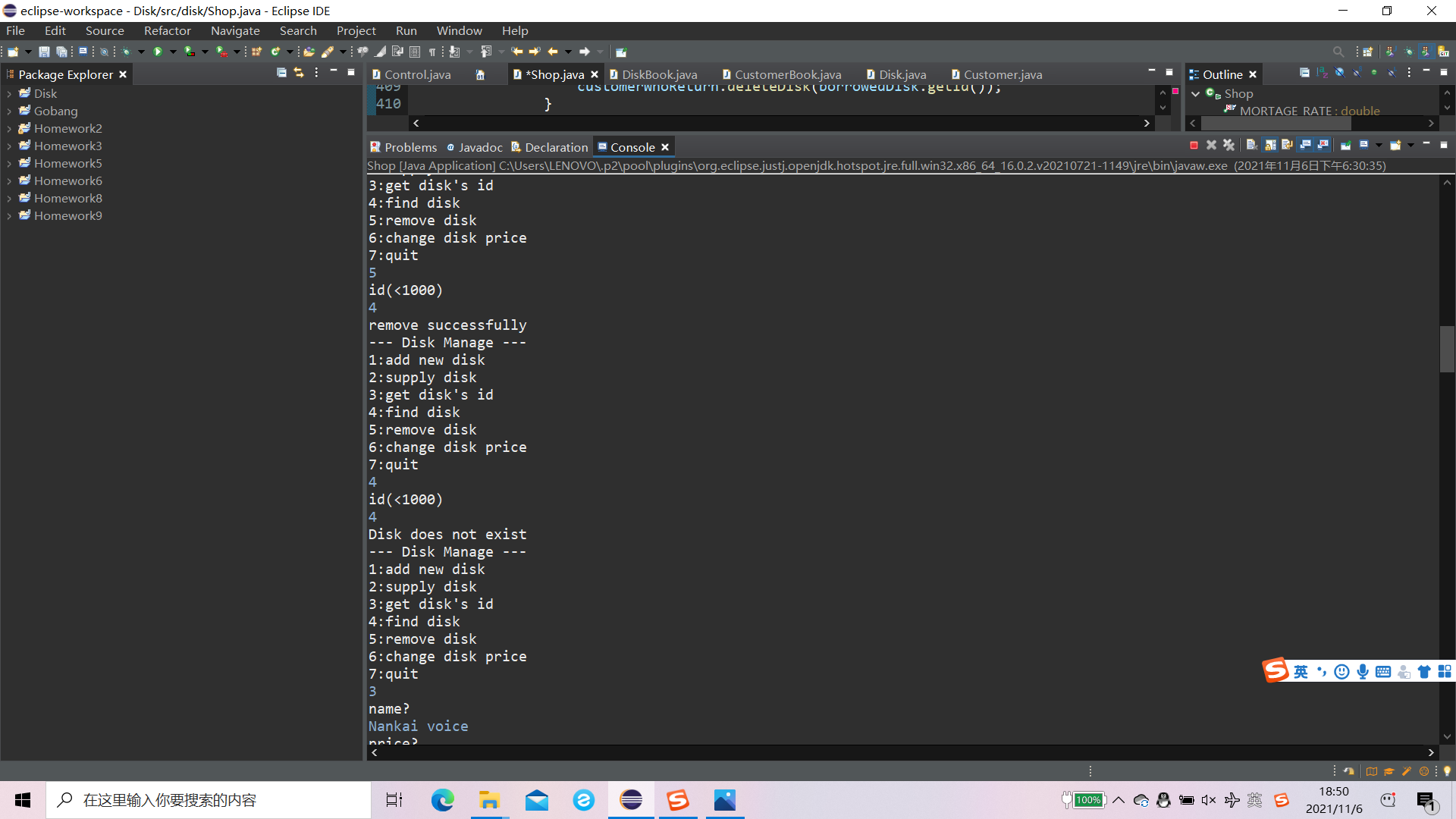
测试添加新CD功能



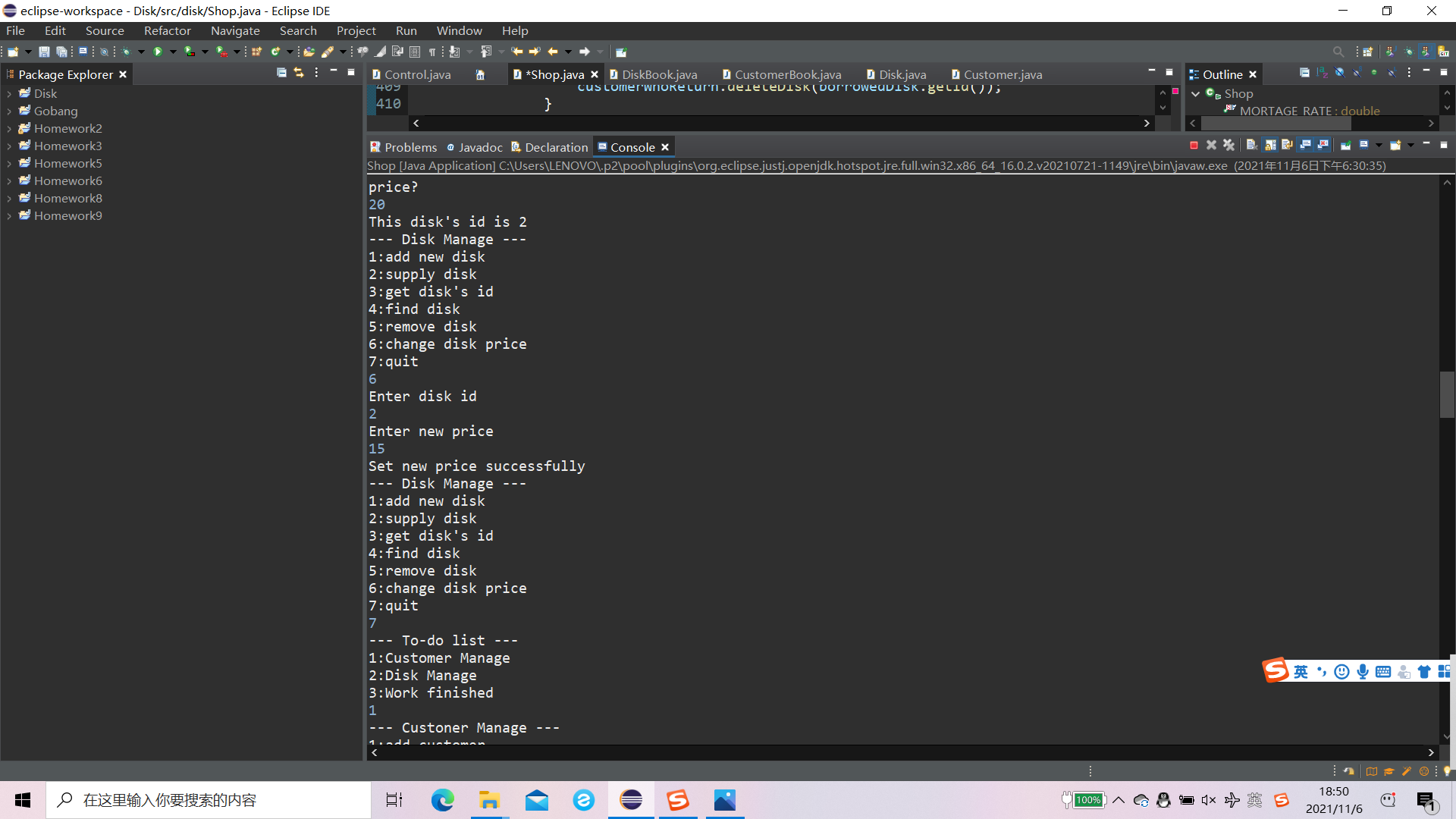
查找刚刚添加的CD



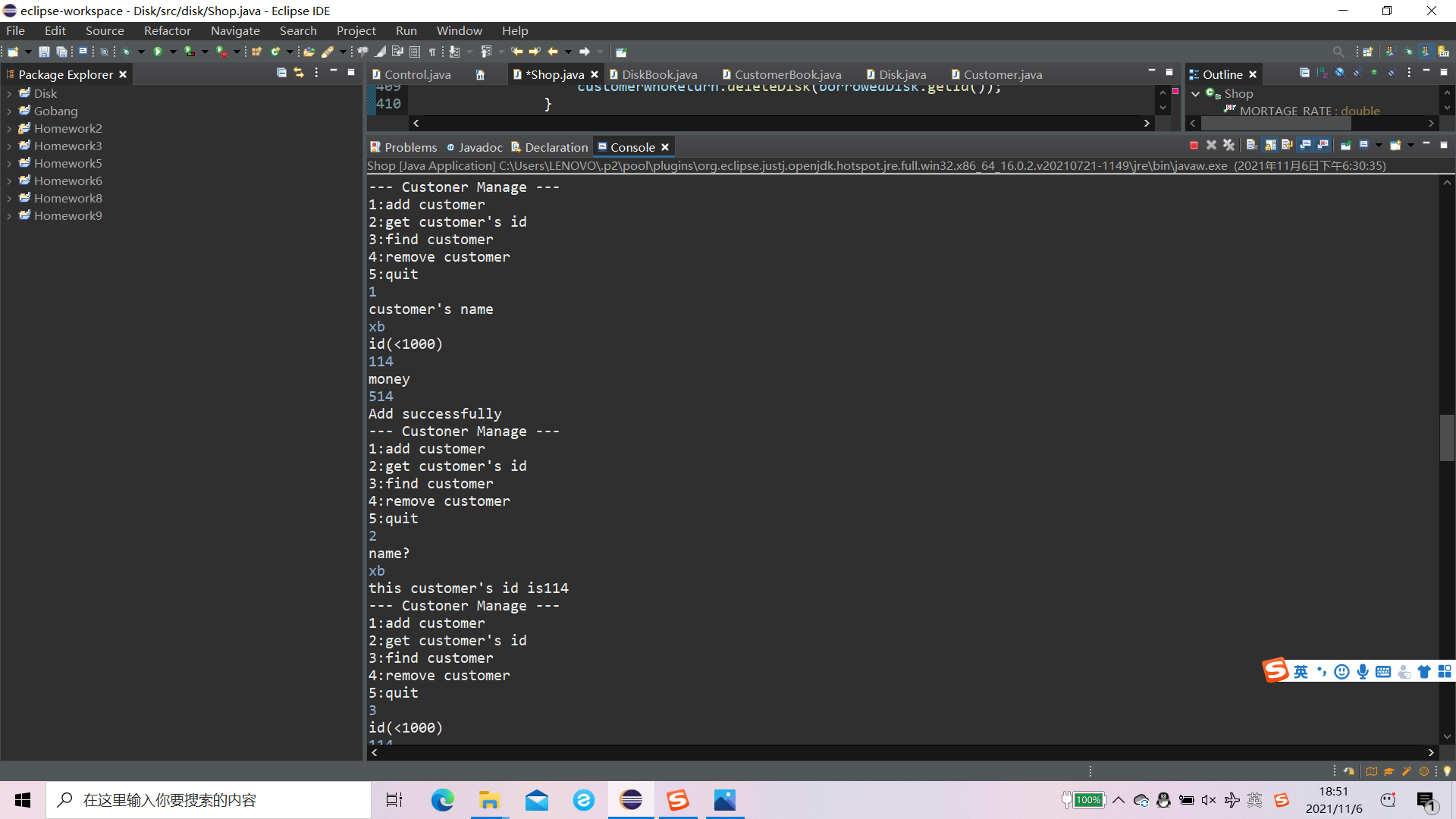
添加CD异常处理



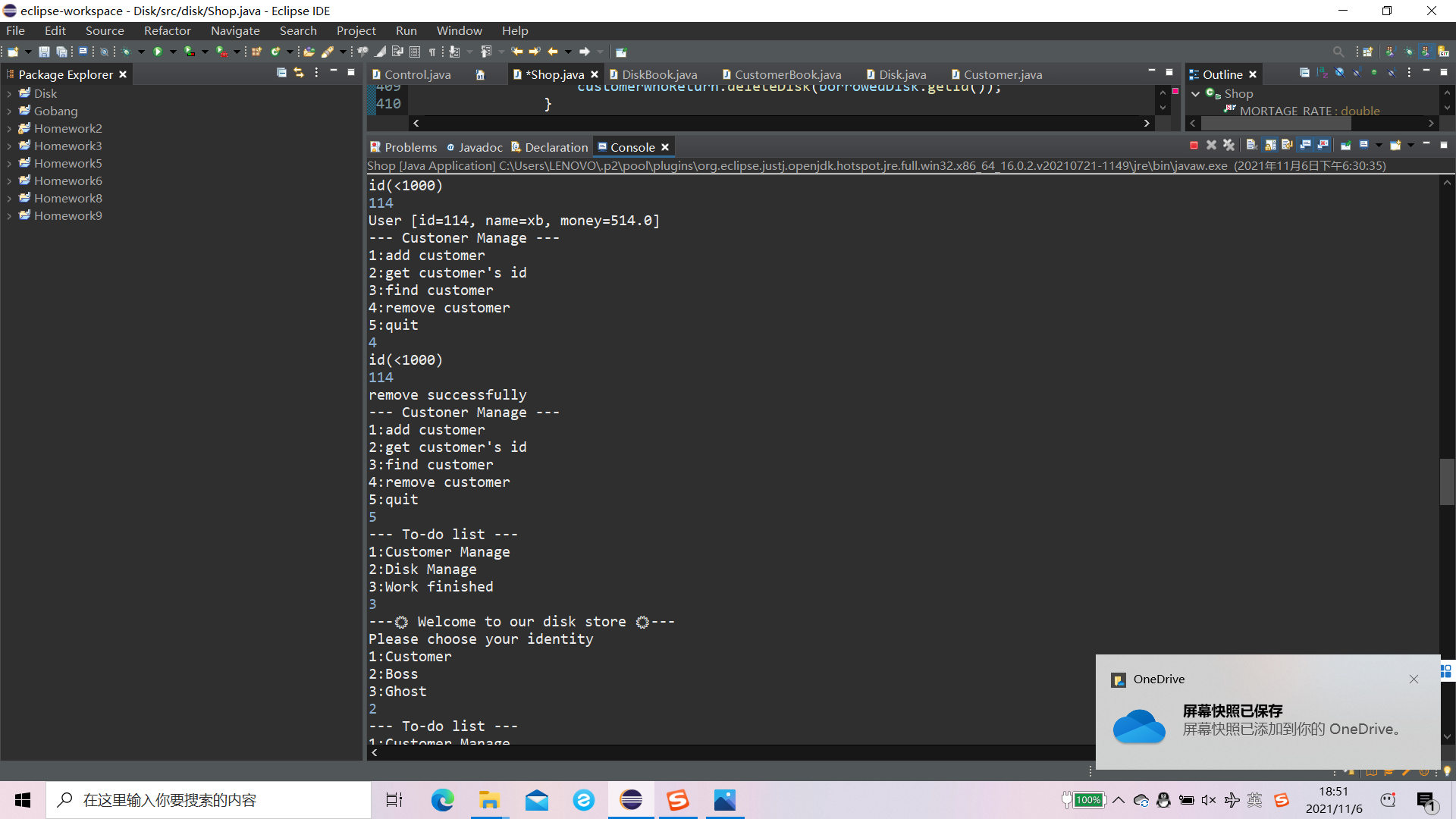
删除CD



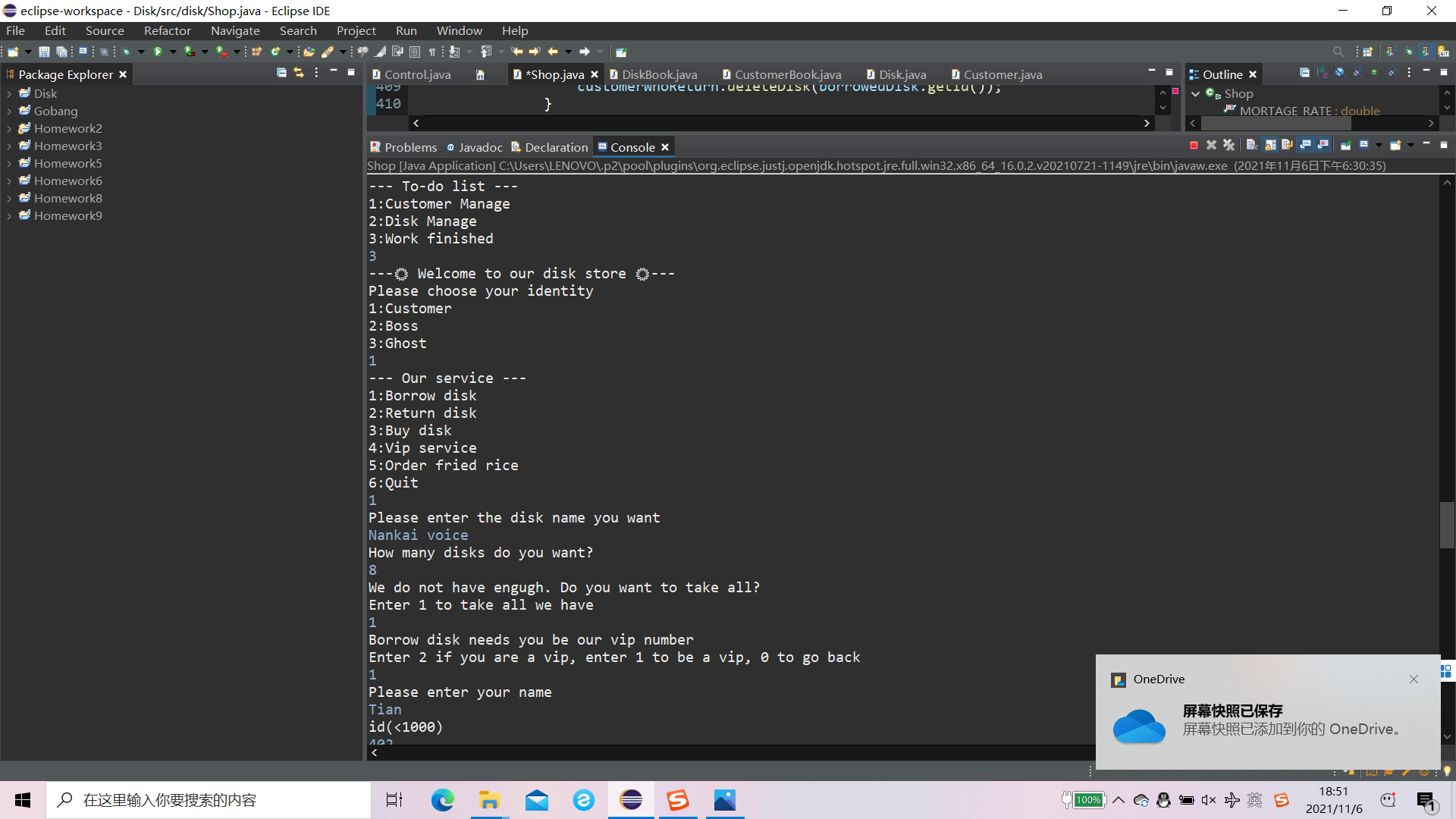
获取ID



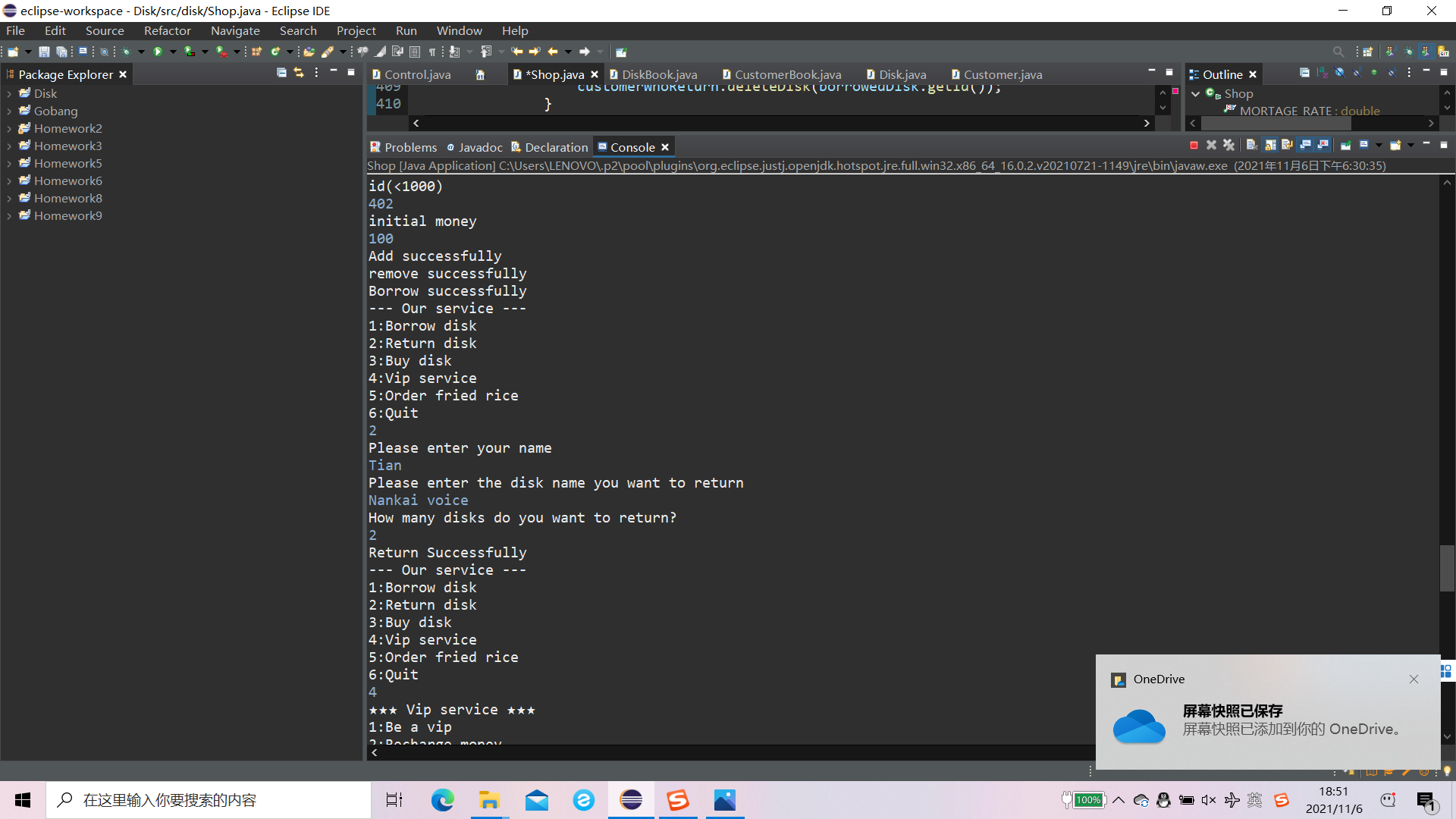
客户管理-添加



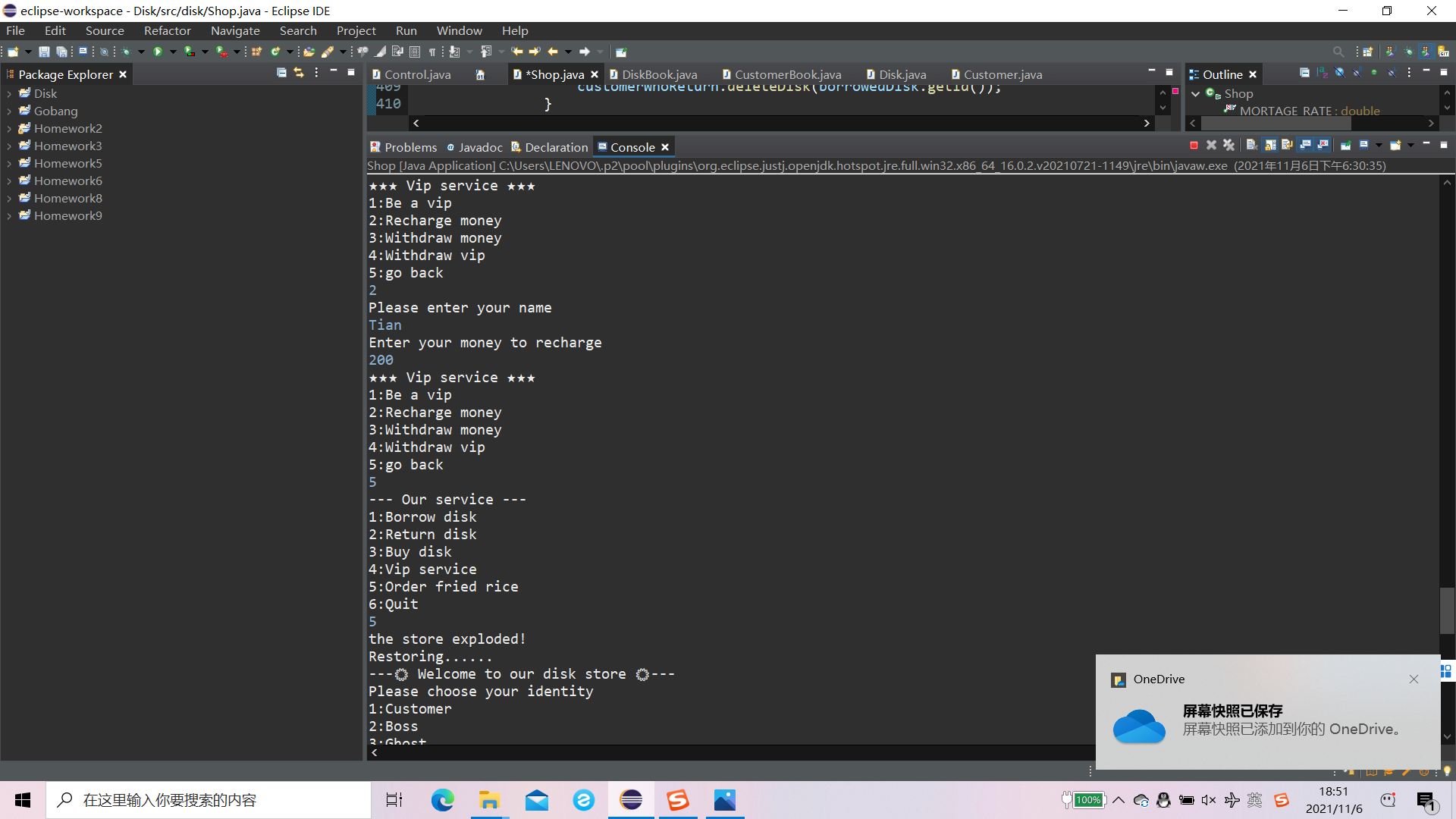
客户管理-删除



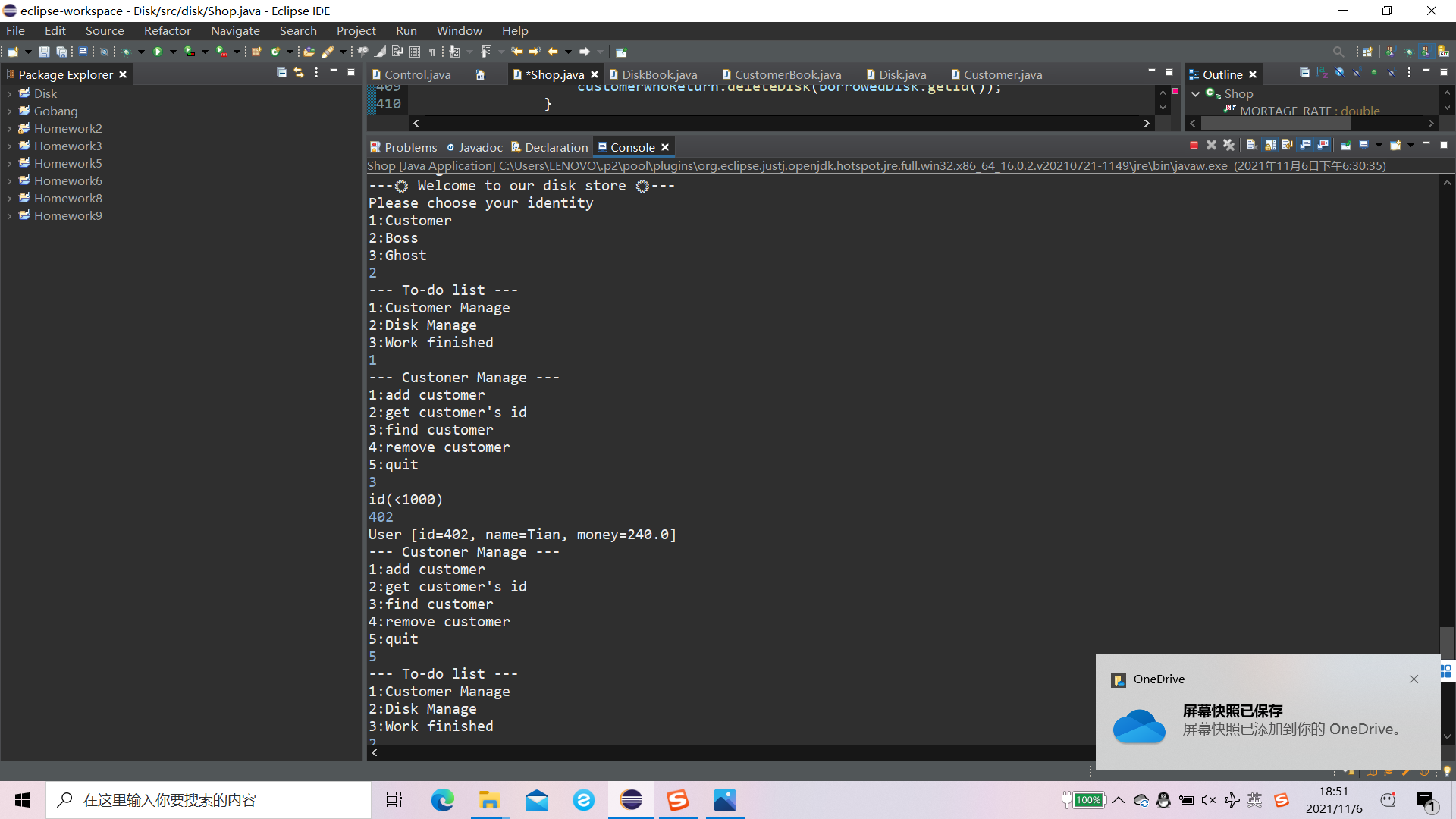
租借CD



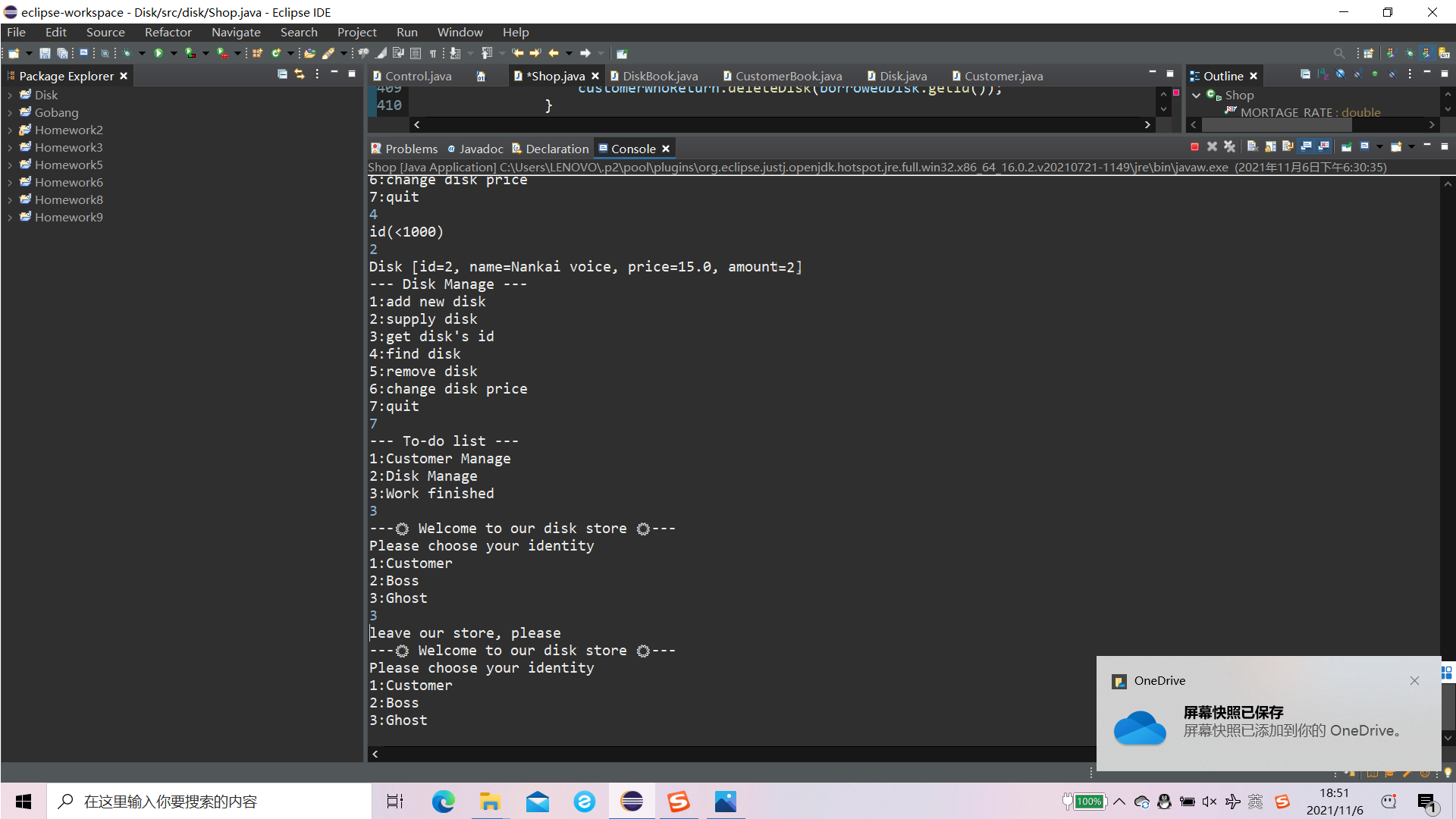
归还CD（remove是指从disk book中移除）



VIP服务（不要在CD店点炒饭！）



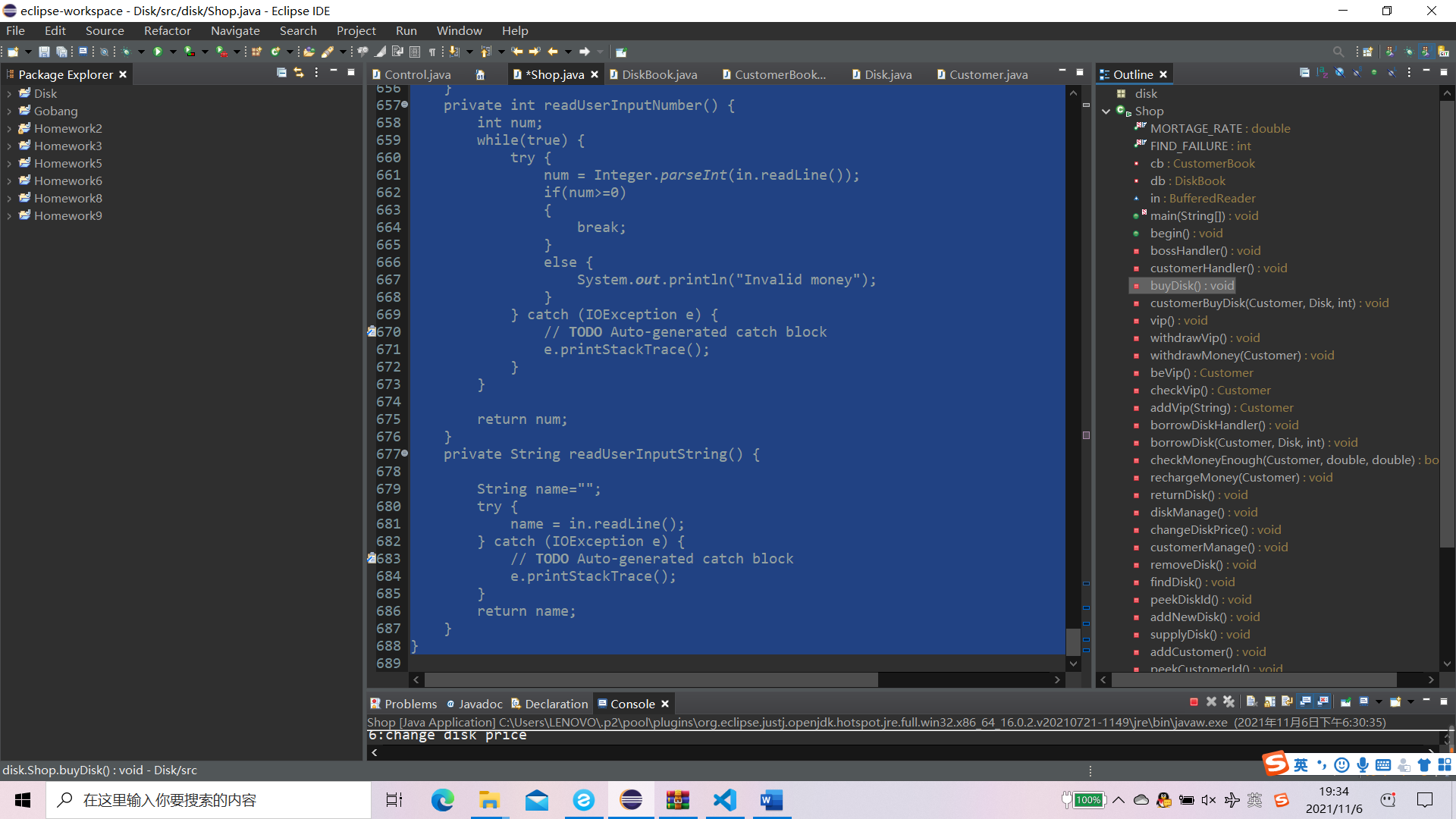
验证顾客余额变化

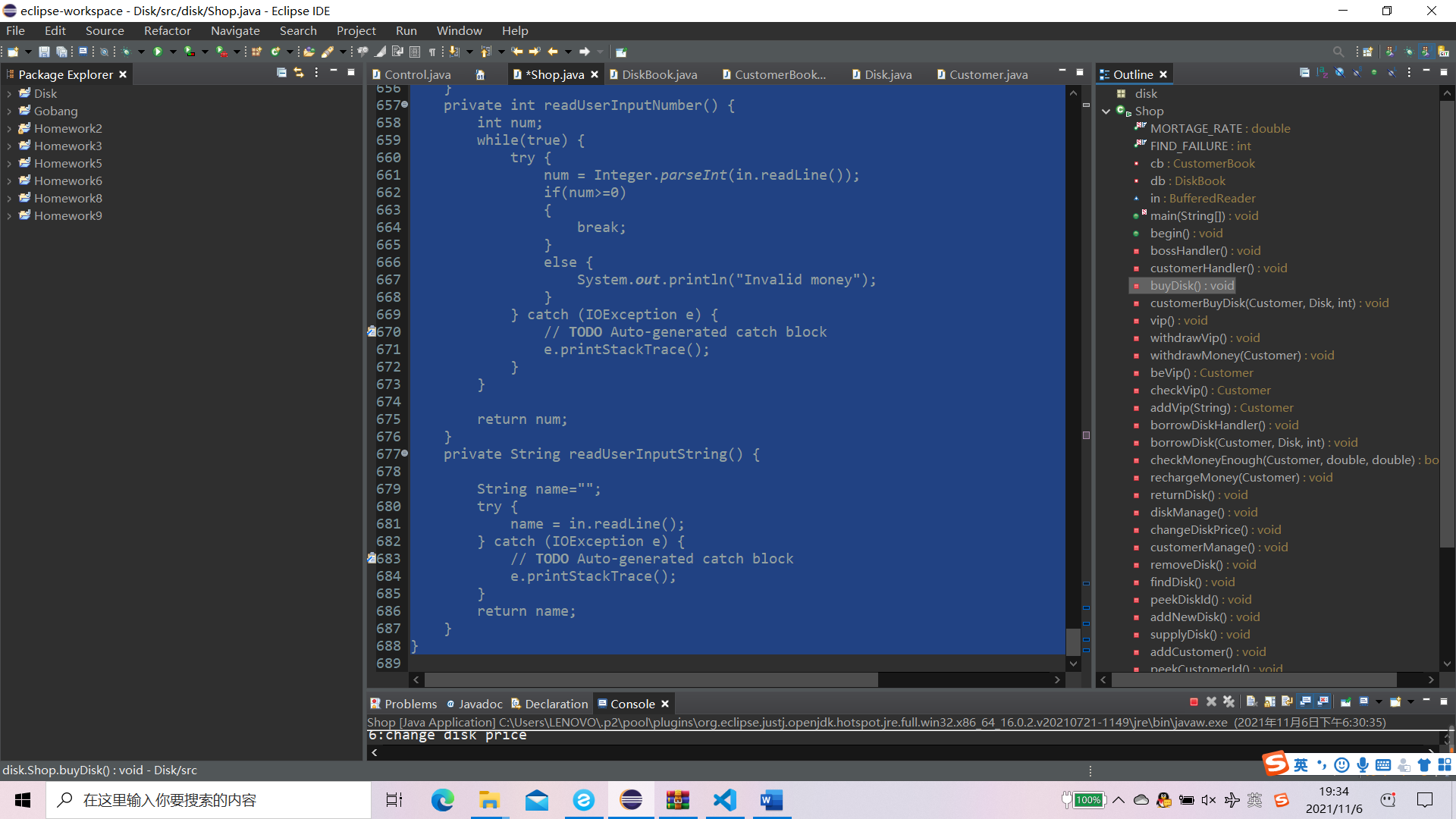
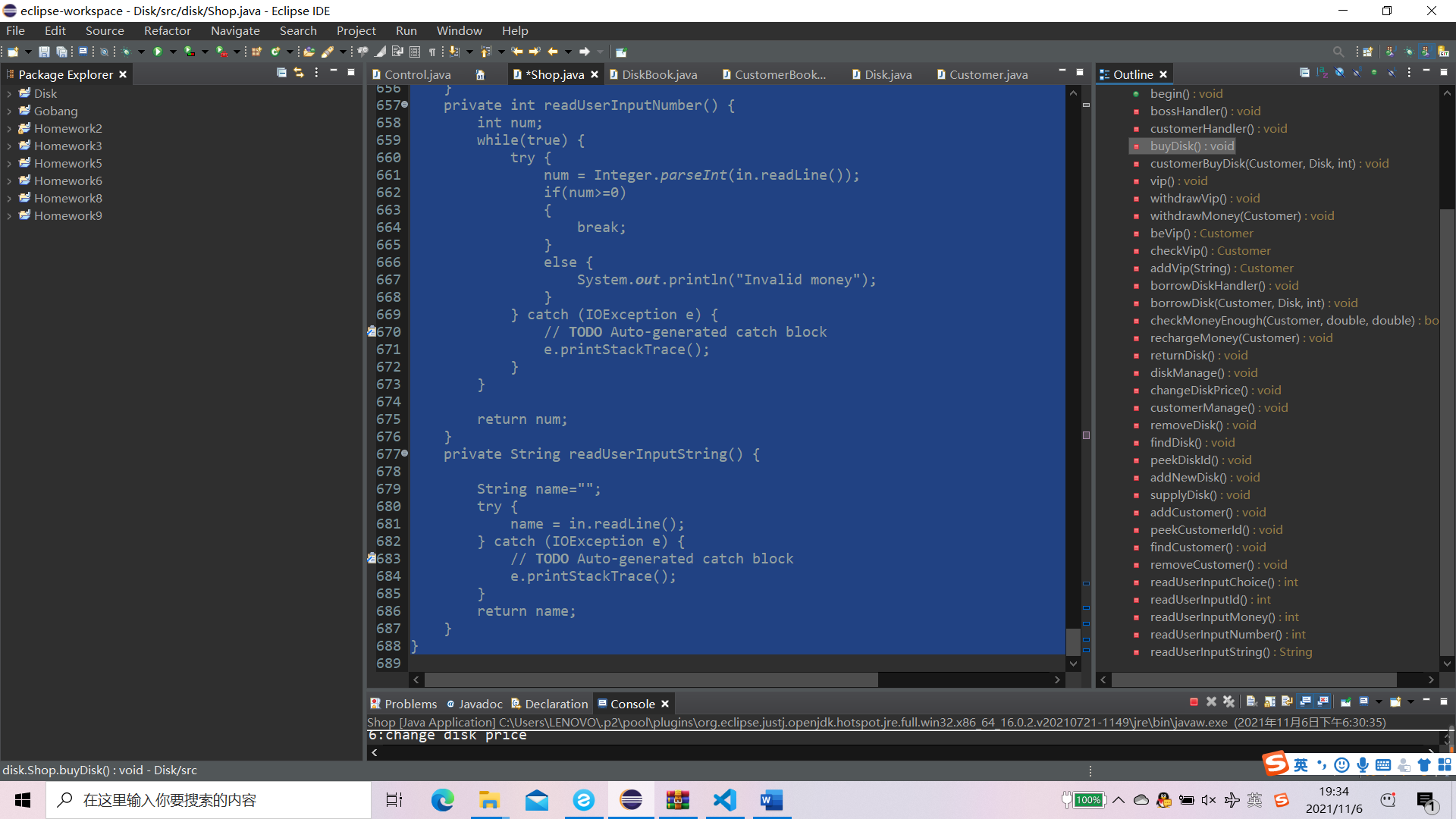


验证CD数量变化

四、程序代码：

主程序Outline





主程序源代码（Shop.java）

package disk;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Shop {

    public static final double MORTAGE\_RATE = 1.2;

    public static final int  FIND\_FAILURE= -1;

    private CustomerBook cb = new CustomerBook();

    private DiskBook db = new DiskBook();

    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

    public static void main(String[] args) {

        Shop shop = new Shop();

        shop.begin();

    }

    public void begin() {

        while(true) {

        //print main menu

            System.out.println("---☀ Welcome to our disk store ☀---");

            System.out.println("Please choose your identity");

            System.out.println("1:Customer");

            System.out.println("2:Boss");

            System.out.println("3:Ghost");

            int choice = readUserInputChoice();

            switch (choice) {

            case 1:

                customerHandler();

                break;

            case 2:

                bossHandler();

                break;

            case 3:

                System.out.println("leave our store, please");

                break;

                default:

                break;

            }

        }

    }

    private void bossHandler() {

        while (true) {

            //print boss menu

            System.out.println("--- To-do list ---");

            System.out.println("1:Customer Manage");

            System.out.println("2:Disk Manage");

            System.out.println("3:Work finished");

            int choice = readUserInputChoice();

            switch (choice) {

            case 1:

                customerManage();

                break;

            case 2:

                diskManage();

                break;

            case 3:

                return;

            }

        }

    }

    private void customerHandler() {

        while(true) {

            //print customer menu

            System.out.println("--- Our service ---");

            System.out.println("1:Borrow disk");

            System.out.println("2:Return disk");

            System.out.println("3:Buy disk");

            System.out.println("4:Vip service");

            System.out.println("5:Order fried rice");

            System.out.println("6:Quit");

            int choice = readUserInputChoice();

            switch (choice) {

            case 1:

                borrowDiskHandler();

                break;

            case 2:

                returnDisk();

                break;

            case 3:

                buyDisk();

                break;

            case 4:

                vip();

                break;

            case 5:

                System.out.println("the store exploded!");

                System.out.println("Restoring......");

                try {

                    Thread.currentThread();

                    Thread.sleep(2000);

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

                return;

            default:

                return;

            }

        }

    }

    private void buyDisk() {

        System.out.println("Please enter the disk name you want");

        String name=readUserInputString();

        int id=db.getDiskId(name);

        if(id==FIND\_FAILURE)

        {

            System.out.println("This store do not have the disk you want");

            return;

        }

        else {

            System.out.println("How many disks do you want?");

            int askNumber=readUserInputNumber();

            int haveNumber=db.findDisk(id).getNum();

            if(askNumber>haveNumber)

            {

                System.out.println("We do not have enough. Do you want to take all?");

                System.out.println("enter 1 to take all we have");

                int choice=readUserInputChoice();

                switch(choice) {

                case 1:

                    askNumber=haveNumber;

                    break;

                default:

                    return;

                }

            }

            Disk diskWhichBuy=db.findDisk(id);

            System.out.println("Buy disks needs you be our vip member");

            System.out.println("enter 2 if you are a vip, enter 1 to be a vip, 0 to go back");

            int choice=readUserInputChoice();

            Customer customerWhoBuy;

            switch(choice) {

            case 2:

                customerWhoBuy=checkVip();

                customerBuyDisk(customerWhoBuy,diskWhichBuy,askNumber);

                break;

            case 1:

                customerWhoBuy=beVip();

                customerBuyDisk(customerWhoBuy,diskWhichBuy,askNumber);

                break;

            case 0:

                return;

            default:

                return;

            }

        }

    }

    private void customerBuyDisk(Customer customerWhoBuy, Disk diskWhichBuy, int askNumber) {

        double price=diskWhichBuy.getPrice();

        double money=customerWhoBuy.getMoney();

        double cost=price\*askNumber;

        boolean wantToborrow=false;

        wantToborrow=checkMoneyEnough(customerWhoBuy,cost,money);

        if(wantToborrow==false)

        {

            return;

        }

        else {

            diskWhichBuy.setNum(diskWhichBuy.getNum()-askNumber);//change in disk book

            db.setDisk(diskWhichBuy);

            diskWhichBuy.setNum(askNumber);//change in customer borrowed book

            customerWhoBuy.addBuyDisk(diskWhichBuy);

            customerWhoBuy.setMoney(money-cost);

            cb.setCustomer(customerWhoBuy);

            System.out.println("You got this disk!");

        }

    }

    private void vip() {

        while(true) {

            System.out.println("★★★ Vip service ★★★");

            System.out.println("1:Be a vip");

            System.out.println("2:Recharge money");

            System.out.println("3:Withdraw money");

            System.out.println("4:Withdraw vip");

            System.out.println("5:go back");

            int choice = readUserInputChoice();

            switch (choice) {

            case 1:

                beVip();

                break;

            case 2:

                rechargeMoney(checkVip());

                break;

            case 3:

                withdrawMoney(checkVip());

                break;

            case 4:

                withdrawVip();

                break;

            case 5:

                return;

            }

        }

    }

    private void withdrawVip() {

        Customer customer=checkVip();

//      withdrawMoney(customer);

        System.out.println("Sorry, your money can not withdraw. Continue?");

        System.out.println("Enter 1 to delete your vip information");

        int choice=readUserInputChoice();

        switch(choice) {

        case 1:

            cb.removeCustomer(customer.getId());

            break;

        default:

            return;

        }

    }

    private void withdrawMoney(Customer customer) {

//      The reason I make them as comment is

//      not they are incorrect for running the program

//      but in most store in reality this service is not available(

        System.out.println("Sorry, you can not do this");

//      System.out.println("Enter your money to withdraw");

//      Integer money=readUserInputMoney();

//      double moneyInVip=customer.getMoney();

//      if(money-moneyInVip>0)

//      {

//          System.out.println("you do not have so much money to withdraw");

//      }

//      else {

//          customer.setMoney(customer.getMoney()-money);

//      }

    }

    private Customer beVip() {

        System.out.println("Please enter your name");

        String name=readUserInputString();

        int id=cb.getCustomerId(name);

        if(id>=0)

        {

            System.out.println("You have already been a vip");

            return cb.findCustomer(id);

        }

        else {

            return addVip(name);

        }

    }

    private Customer checkVip() {

        System.out.println("Please enter your name");

        String name=readUserInputString();

        int id=cb.getCustomerId(name);

        if(id>=0)

        {

            return cb.findCustomer(id);

        }

        else {

            System.out.println("Did not find your information, please add vip");

            return addVip(name);

        }

    }

    private Customer addVip(String name) {

        boolean successIndicator=false;

        Customer nc=null;

        while(successIndicator==false) {

            System.out.println("id(<1000)");

            int id = readUserInputId();

            System.out.println("initial money");

            int money =readUserInputMoney();

            nc = new Customer(id, name, money);

            successIndicator=cb.addCustomer(nc);

        }

        return nc;

    }

    private void borrowDiskHandler() {

        System.out.println("Please enter the disk name you want");

        String name=readUserInputString();

        int id=db.getDiskId(name);

        if(id==FIND\_FAILURE)

        {

            System.out.println("This store does not have the disk you want");

            return;

        }

        else {

            System.out.println("How many disks do you want?");

            int askNumber=readUserInputNumber();

            int haveNumber=db.findDisk(id).getNum();

            if(askNumber>haveNumber)

            {

                System.out.println("We do not have engugh. Do you want to take all?");

                System.out.println("Enter 1 to take all we have");

                int choice=readUserInputChoice();

                switch(choice) {

                case 1:

                    askNumber=haveNumber;

                    break;

                default:

                    return;

                }

            }

            Disk diskWhichBorrow=db.findDisk(id);

            System.out.println("Borrow disk needs you be our vip number");

            System.out.println("Enter 2 if you are a vip, enter 1 to be a vip, 0 to go back");

            int choice=readUserInputChoice();

            Customer customerWhoBorrow;

            switch(choice) {

            case 2:

                customerWhoBorrow=checkVip();

                borrowDisk(customerWhoBorrow,diskWhichBorrow,askNumber);

                break;

            case 1:

                customerWhoBorrow=beVip();

                borrowDisk(customerWhoBorrow,diskWhichBorrow,askNumber);

                break;

            case 0:

                return;

            default:

                return;

            }

        }

    }

    private void borrowDisk(Customer customerWhoBorrow,Disk diskWhichBorrow,int askNumber) {

        double price=diskWhichBorrow.getPrice();

        double money=customerWhoBorrow.getMoney();

        double cost=MORTAGE\_RATE\*price\*askNumber;

        boolean wantToborrow=false;

        wantToborrow=checkMoneyEnough(customerWhoBorrow,cost,money);

        if(wantToborrow==false)

        {

            return;

        }

        else {

            if(diskWhichBorrow.getNum()==askNumber)

            {

                db.removeDisk(diskWhichBorrow.getId());

            }

            else {

                diskWhichBorrow.setNum(diskWhichBorrow.getNum()-askNumber);//change in disk book

                db.setDisk(diskWhichBorrow);

            }

            Disk customerGotDisk=diskWhichBorrow;

            customerGotDisk.setNum(askNumber);

            customerWhoBorrow.addBorrowDisk(customerGotDisk);

            customerWhoBorrow.setMoney(money-cost);

            cb.setCustomer(customerWhoBorrow);

            System.out.println("Borrow successfully");

        }

    }

    private boolean checkMoneyEnough(Customer customerWhoBorrow,double cost, double money) {

        while(money-cost<0)

        {

            System.out.println("Your money is not enough. Do you want to recharge money in your account? ");

            System.out.println("Enter 1 to recharge, 0 to go back");

            int choice=readUserInputChoice();

            switch(choice) {

            case 1:

                rechargeMoney(customerWhoBorrow);

                break;

            case 0:

                return false;

            default:

                return false;

            }

        }

        return true;

    }

    private void rechargeMoney(Customer customer) {

        System.out.println("Enter your money to recharge");

        int money=readUserInputMoney();

        customer.setMoney(customer.getMoney()+money);

    }

    private void returnDisk() {

        Customer customerWhoReturn=checkVip();

        System.out.println("Please enter the disk name you want to return");

        String name=readUserInputString();

        Disk borrowedDisk=customerWhoReturn.checkBorrowDisk(name);

        if(borrowedDisk==null)

        {

            System.out.println("You do not have the disk you want");

            return;

        }

        else {

            System.out.println("How many disks do you want to return?");

            int returnNumber=readUserInputNumber();

            int borrowNumber=borrowedDisk.getNum();

            if(returnNumber>borrowNumber)

            {

                System.out.println("You do not have so many disks");

                return;

            }

            else

            {

            double returnMoney=returnNumber\*borrowedDisk.getPrice();

                customerWhoReturn.setMoney(customerWhoReturn.getMoney()+returnMoney);

                if(borrowNumber==returnNumber)

                {

                    customerWhoReturn.deleteDisk(borrowedDisk.getId());

                }

                else {

                    borrowedDisk.setNum(borrowNumber-returnNumber);

                    customerWhoReturn.setDisk(borrowedDisk);

                }

                Disk returnedDisk=borrowedDisk;

                Disk returnedDiskInDiskBook=db.findDisk(returnedDisk.getId());

                if(returnedDiskInDiskBook==null)

                {

                    db.setDisk(returnedDisk);

                }

                else

                {

                    returnedDisk.setNum(returnedDiskInDiskBook.getNum()+returnNumber);

                    db.setDisk(returnedDisk);

                }

                cb.setCustomer(customerWhoReturn);

                System.out.println("Return Successfully");

            }

        }

    }

    private void diskManage() {

        while(true) {

            //disk manage menu

            System.out.println("--- Disk Manage ---");

            System.out.println("1:add new disk");

            System.out.println("2:supply disk");

            System.out.println("3:get disk's id");

            System.out.println("4:find disk");

            System.out.println("5:remove disk");

            System.out.println("6:change disk price");

            System.out.println("7:quit");

            int choice = readUserInputChoice();

            switch(choice) {

            case 1:

                addNewDisk();

                break;

            case 2:

                supplyDisk();

                break;

            case 3:

                peekDiskId();

                break;

            case 4:

                findDisk();

                break;

            case 5:

                removeDisk();

                break;

            case 6:

                changeDiskPrice();

                break;

            case 7:

                return;

            default:

                return;

            }

        }

    }

    private void changeDiskPrice() {

        System.out.println("Enter disk id");

        int id=readUserInputId();

        Disk disk=db.findDisk(id);

        if(disk==null)

        {

            System.out.println("Disk does not exist");

        }

        else {

            System.out.println("Enter new price");

            int price=readUserInputMoney();

            disk.setPrice(price);

            db.setDisk(disk);

            System.out.println("Set new price successfully");

        }

    }

    private void customerManage() {

        while(true) {

//          customer manage menu

            System.out.println("--- Custoner Manage ---");

            System.out.println("1:add customer");

            System.out.println("2:get customer's id");

            System.out.println("3:find customer");

            System.out.println("4:remove customer");

            System.out.println("5:quit");

            int choice = readUserInputChoice();

            switch(choice) {

            case 1:

                addCustomer();

                break;

            case 2:

                peekCustomerId();

                break;

            case 3:

                findCustomer();

                break;

            case 4:

                removeCustomer();

                break;

            case 5:

                return;

            default:

                return;

            }

        }

    }

    private void removeDisk() {

        System.out.println("id(<1000)");

            int id = readUserInputId();

            db.removeDisk(id);

    }

    private void findDisk() {

        System.out.println("id(<1000)");

            int id = readUserInputId();

            Disk disk = db.findDisk(id);

            if(disk==null)

            {

                System.out.println("Disk does not exist");

            }

            else {

                System.out.println(disk);

            }

    }

    private void peekDiskId() {

            System.out.println("name?");

            String name =readUserInputString();

            System.out.println("price?");

            int price = readUserInputMoney();

            db.peekDiskId(name,price);

    }

    private void addNewDisk() {

            System.out.println("disk's name");

            String name =readUserInputString();

            System.out.println("id(<1000)");

            int id = readUserInputId();

            System.out.println("price");

            int price =readUserInputMoney();

            System.out.println("amount");

            int number =readUserInputNumber();

            Disk nd = new Disk(id,name,price,number);

            db.addNewDisk(nd);

    }

    private void supplyDisk() {

            System.out.println("Enter before ensuring your id is correct");

            System.out.println("id(<1000)");

            int id =readUserInputId();

            System.out.println("amount");

            int number = readUserInputNumber();

            db.supplyDisk(id,number);

    }

    private void addCustomer() {

            System.out.println("customer's name");

            String name = readUserInputString();

            System.out.println("id(<1000)");

            int id = readUserInputId();

            System.out.println("money");

            int money =readUserInputMoney();

            Customer nc = new Customer(id, name, money);

            cb.addCustomer(nc);

    }

    private void peekCustomerId() {

            System.out.println("name?");

            String name =readUserInputString();

            cb.peekCustomerId(name);

    }

    private void findCustomer() {

        System.out.println("id(<1000)");

            int id = readUserInputId();

            Customer customer = cb.findCustomer(id);

            if(customer==null)

            {

                System.out.println("The customer does not exist");

            }

            else {

                System.out.println(customer);

            }

    }

    private void removeCustomer() {

        System.out.println("id(<1000)");

            int id = readUserInputId();

            cb.removeCustomer(id);

    }

//Input handlers with checking incorrect format

    private int readUserInputChoice() {

        try {

            String line;

            line = in.readLine();

            return Integer.parseInt(line);

        } catch (IOException e) {

            e.printStackTrace();

        }

        return 0;

    }

    private int readUserInputId() {

        int id;

        while(true) {

            try {

                id = Integer.parseInt(in.readLine());

                if(id>0&&id<1000)

                {

                    break;

                }

                else {

                    System.out.println("Invalid id");

                }

            } catch (IOException e) {

                // TODO Auto-generated catch block

                e.printStackTrace();

            }

        }

        return id;

    }

    private int readUserInputMoney() {

        int money;

        while(true) {

            try {

                money = Integer.parseInt(in.readLine());

                if(money>=0)

                {

                    break;

                }

                else {

                    System.out.println("Invalid money");

                }

            } catch (IOException e) {

                // TODO Auto-generated catch block

                e.printStackTrace();

            }

        }

        return money;

    }

    private int readUserInputNumber() {

        int num;

        while(true) {

            try {

                num = Integer.parseInt(in.readLine());

                if(num>=0)

                {

                    break;

                }

                else {

                    System.out.println("Invalid money");

                }

            } catch (IOException e) {

                // TODO Auto-generated catch block

                e.printStackTrace();

            }

        }

        return num;

    }

    private String readUserInputString() {

        String name="";

        try {

            name = in.readLine();

        } catch (IOException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        }

        return name;

    }

}

Disk.Java

package disk;

public class Disk {

    private int id;

    private String name;

    private double price;

    private int num;

    public Disk(int id, String name, int price, int num) {

        super();

        this.id = id;

        this.name = name;

        this.price = price;

        this.num = num;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public double getPrice() {

        return price;

    }

    public void setPrice(double price) {

        this.price = price;

    }

    public int getNum() {

        return num;

    }

    public void setNum(int num) {

        this.num = num;

    }

    @Override

    public String toString() {

        return "Disk [id=" + id + ", name=" + name + ", price=" + price + ", amount=" + num + "]";

    }

}

DiskBook.java

package disk;

public class DiskBook {

    private Disk[] data = new Disk[1000];

    public void addNewDisk(Disk d) {

        int id = d.getId();

        Disk disk = findDisk(id);

        if(disk==null) {

            data[id]=d;

            System.out.println("Add successfully");

        }else {

            System.out.println("This id already linked to a disk");

        }

    }

    public void print() {

        for (Disk disk : data) {

            System.out.println(disk);

        }

    }

    public void removeDisk(int id) {

        if(data[id]!=null)

        {

            data[id]=null;

            System.out.println("remove successfully");

        }

        else

        {

            System.out.println("this customer does not exist");

        }

    }

    public Disk findDisk(int id) {

        return data[id];

    }

    @Override

    public String toString() {

        String result = "";

        for (Disk disk : data) {

            if(disk!=null) {

                result += disk+"\n";

            }

        }

        return result;

//      return "DiskBook [data=" + Arrays.toString(data) + "]";

    }

    public void supplyDisk(int id,int number) {

        Disk disk = findDisk(id);

        if(disk==null) {

            System.out.println("Please choose add new disk");

        }else {

            int num = disk.getNum()+number;

            disk.setNum(num);

            System.out.println("Supply successfully");

        }

    }

    public void peekDiskId(String name,int price) {

        int min=Integer.MAX\_VALUE;

        int id=-1;

        for (Disk disk : data) {

            if(disk!=null)

            if(name.equals(disk.getName())&&(double)price==disk.getPrice())

            {

                    //if more than 1 disk is the same, show the less one

                    if(min>disk.getNum())

                    {

                        min=disk.getNum();

                        id=disk.getId();

                    }

            }

        }

        if(id==-1)

        {

            System.out.println("This disk does not exist");

        }

        else {

            System.out.println("This disk's id is "+id);

        }

    }

    public int getDiskId(String name) {

        double min=Double.POSITIVE\_INFINITY;

        int id=-1;

        for (Disk disk : data) {

            if(disk!=null)

               if(name.equals(disk.getName()))

               {

                    //if more than 1 disk is the same, get the cheapest one

                    if(min>disk.getPrice())

                    {

                        min=disk.getPrice();

                        id=disk.getId();

                    }

            }

        }

        return id;

    }

    public void setDisk(Disk disk)

    {

        data[disk.getId()]=disk;

    }

}

Customer.Java（包含两个数组用于保存接走和买走的CD）

package disk;

public class Customer {

    int id;

    String name;

    double money;

    private Disk[] diskBorrow=new Disk[1000];

    private Disk[] diskBuy=new Disk[1000];

    DiskBook note=new DiskBook();

    public Customer(int id, String name, int money) {

        super();

        this.id = id;

        this.name = name;

        this.money = money;

    }

    public Disk checkBorrowDisk(String cname)

    {

        double max=Double.NEGATIVE\_INFINITY;

        Disk targetDisk=null;

        for (Disk disk : diskBorrow) {

            if(disk!=null)

            if(cname.equals(disk.getName()))

            {

                    //if more than 1 disk is the same, get the most expensive one

                    if(max<disk.getPrice())

                    {

                        max=disk.getPrice();

                        targetDisk=disk;

                    }

            }

        }

        return targetDisk;

    }

    public void addBorrowDisk(Disk disk)

    {

        diskBorrow[disk.getId()]=disk;

    }

    public void setDisk(Disk setDisk)

    {

        diskBorrow[setDisk.getId()]=setDisk;

    }

    public void deleteDisk(int id)

    {

        diskBorrow[id]=null;

    }

    public void addBuyDisk(Disk disk)

    {

        diskBuy[disk.getId()]=disk;

    }

    @Override

    public int hashCode() {

        final int prime = 31;

        int result = 1;

        result = prime \* result + id;

        result = (int) (prime \* result + money);

        result = prime \* result + ((name == null) ? 0 : name.hashCode());

        return result;

    }

    @Override

    public boolean equals(Object obj) {

        if (this == obj)

            return true;

        if (obj == null)

            return false;

        if (getClass() != obj.getClass())

            return false;

        Customer other = (Customer) obj;

        if (id != other.id)

            return false;

        if (money != other.money)

            return false;

        if (name == null) {

            if (other.name != null)

                return false;

        } else if (!name.equals(other.name))

            return false;

        return true;

    }

    @Override

    public String toString() {

        return "User [id=" + id + ", name=" + name + ", money="

+ money + "]";

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public double getMoney() {

        return money;

    }

    public void setMoney(double d) {

        this.money = d;

    }

    public DiskBook getNote() {

        return note;

    }

    public void setNote(DiskBook note) {

        this.note = note;

    }

}

CustomerBook.java

package disk;

public class CustomerBook {

    private Customer[] data = new Customer[1000];

    public boolean addCustomer(Customer u) {

        if(data[u.id]!=null)

        {

            System.out.println("This id already linked to a customer");

            return false;

        }

        else

        {

            data[u.id] = u;

            System.out.println("Add successfully");

            return true;

        }

    }

    public Customer findCustomer(int id) {

            return data[id];

    }

    public void removeCustomer(int id) {

        if(data[id]!=null)

        {

            data[id]=null;

            System.out.println("remove successfully");

        }

        else

        {

            System.out.println("this customer does not exist");

        }

    }

    public void print() {

    }

    @Override

    public String toString() {

        String result = "";

        for (Customer customer : data) {

            result += customer+"\n";

        }

        return result;

    }

    public void peekCustomerId(String name) {

        boolean match=false;

        for (Customer customer : data) {

            if(customer!=null)

            if(name.equals(customer.getName()))

            {

                match=true;

                    System.out.println("this customer's id is"+customer.id);

            }

        }

        if(match==false)

        {

            System.out.println("this customer does not exist");

        }

    }

    public int getCustomerId(String name) {

        int id=-1;

        for (Customer customer : data) {

            if(customer!=null)

            if(name.equals(customer.getName()))

            {

                id=customer.getId();

            }

        }

            return id;

    }

    public void setCustomer(Customer customer)

    {

        data[customer.getId()]=customer;

    }

}