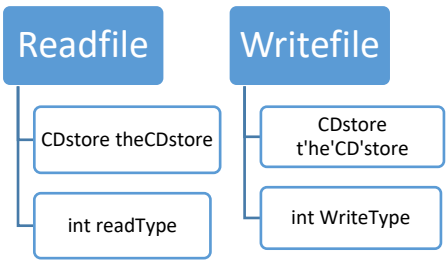
Lab5 实验报告

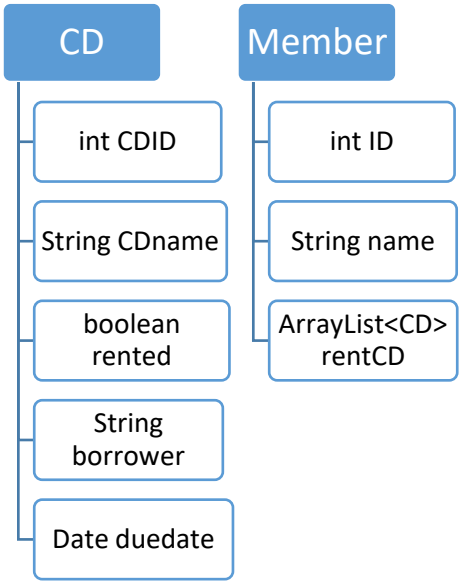
一、概览

除主类外共有六个类

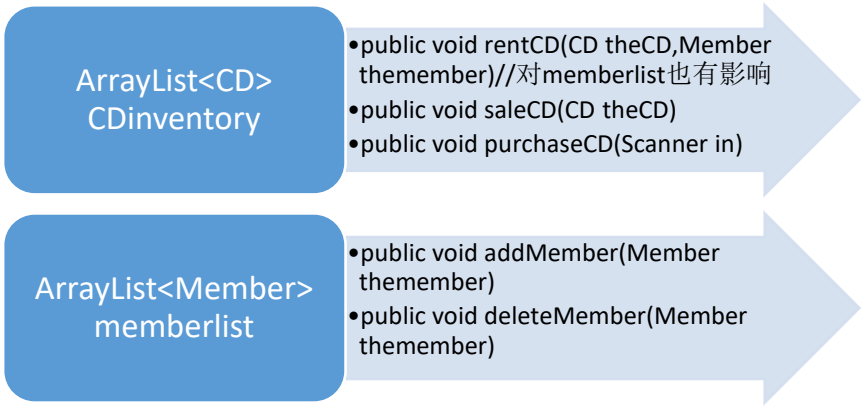
两个对文件进行处理的类：Readfile 和 Writefile



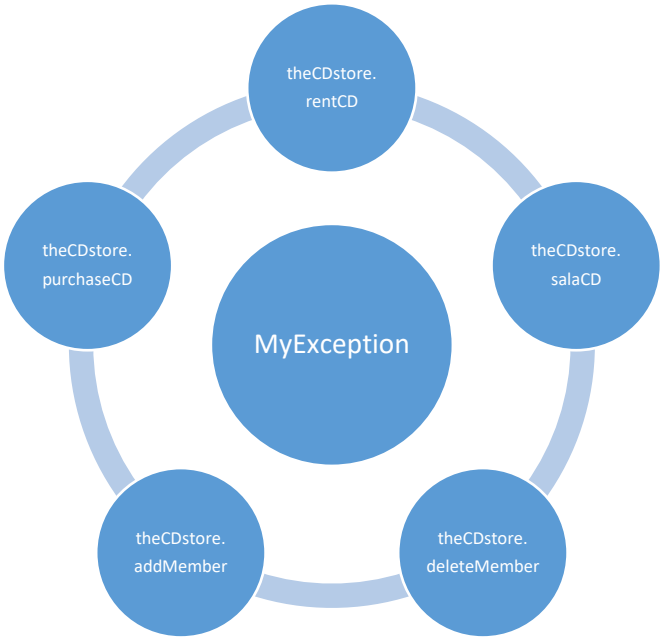
用于描述 CD 和会员基本信息的两个类 CD 和 Member



描述 CD 店的类 CDstore（由 ArrayList<CD> CDinventory 和 ArrayList<Member>memberlist 组成）有多个方法



用于自定义异常的类 MyException



二、Readfile 类

成行读取 txt 文件中的数据

Readfile 类用于读取文件中的数据，并且将读取到的数据保存成一个 CDstore 对象

该类中含有两个变量，一个 CDstore 对象（也就是存入数据的 CDstore）一个用于区别读入的文件是 CD 库存文件还是会员名单的 int 型变量 readType

以及三个方法，其中 inputCD 和 inputMember 分别用于将输入的数据存储为 CD 形式和 Member 形式，也就是构建 CDstore 中的 CDinventory 和 memberlist 两个部分，函数 readFile 是输入的主要函数，读入文件中的每一行将其以空格分割并调用 inputCD 和 inputMember 构建 CDinventory 和 memberlist

使用 StringTokenizer 类来将每行读入的数据用空格分割

BufferdReader 将数据批量读入缓冲区

```
class Readfile
{
    CDstore theCDstore;
    int readType;//区别读入的文件是CD库存文件还是会员名单
    public void readFile(String filepath){
        File file=new File(filepath);
        if(file.exists()){
            try
            {
                FileReader fileReader=new FileReader(file);
                BufferedReader buffreader=new BufferedReader(fileReader);
                String line=null;
                while((line=buffreader.readLine())!=null)//逐行读取
                {
                    StringTokenizer token=new StringTokenizer(line," ");
                    if(this.readType==1)
```

```
                    this.inputCD(token);//将输入的数据存储为CD形式
                else if(this.readType==2)
                    this.inputMember(token);//将输入的数据存储为Member形式
            }
            buffreader.close();
            fileReader.close();
        }
        catch(FileNotFoundException e)
        {
            System.out.println("找不到文件");
            e.printStackTrace();
        }
        catch(IOException e)
        {
            System.out.println("输入流出错");
            e.printStackTrace();
        }
    }
}
public void inputCD(StringTokenizer token)//将读取的数据存成CD
{
    SimpleDateFormat dateformat=new SimpleDateFormat("yyyy-MM-dd");
    String theDate="";
    String str1=token.nextToken();
    String str2=token.nextToken();
    int theID=Integer.parseInt(str1);
    CD theCD=new CD(theID,str2);
    theCD.borrower=token.nextToken();
    theDate=token.nextToken();
    try {
        theCD.duedate=dateformat.parse(theDate);
    } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    this.theCDstore.CDinventory.add(theCD);
}
public void inputMember(StringTokenizer token)//将读取的数据存成Member
{
    Member thememberr=new Member();
    String str1=token.nextToken();
    thememberr.ID=Integer.parseInt(str1);
    thememberr.name=token.nextToken();
    String str2="";
```

```
        String str3="";
        while(token.hasMoreTokens())
        {
            str2=token.nextToken();
            str3=token.nextToken();
            int theID=Integer.parseInt(str2);
            CD theCD=new CD(theID,str3);
            thememberrentCD.add(theCD);
        }
        this.theCDstore.memberlist.add(thememberr);

    }
    }
```

三、Writefile 类

用于将修改过的数据按照之前读入的格式写回文件中

含有 CDstore theCDstore 和 int writeType 两个变量，其中 theCDstore 是将要写入文件的数据，writeType 用于分辨要写入 CD 库存文件还是会员名单文件

含有一个方法 writeFile，用于向目标文件中写入相应格式的数据

使用 BufferedWriter 将文件先读取到缓冲区，再从缓冲区批量写入文件

```
class Writefile
{
    CDstore theCDstore;
    int writeType;
    public void writeFile(String filepath)
    {
        File file=new File(filepath);
        if(file.exists()){
            try{
                FileWriter fileWriter=new FileWriter(file);
                BufferedWriter out=new BufferedWriter(fileWriter);
                if(this.writeType==1)
                {
                    for(int i=0;i<this.theCDstore.CDinventory.size();i++)
                    {
                        CD theCD=new CD();
                        theCD=this.theCDstore.CDinventory.get(i);
                        out.write(theCD.CDID+" "+theCD.CDname+"
"+theCD.rented+" "+theCD.borrower+" "+theCD.duedate+"/r/n");
                    }

                }
                if(this.writeType==2)
                {
                    for(int i=0;i<this.theCDstore.memberlist.size();i++)
```

```
                {
                    Member thememberr=new Member();
                    thememberr=this.theCDstore.memberlist.get(i);
                    out.write(thememberr.ID+" "+thememberr.name+" ");
                    for(int j=0;j<thememberr.rentCD.size();i++)
                    {
                        CD tempCD=new CD();
                        tempCD=thememberr.rentCD.get(i);
                        out.write(tempCD.CDID+" "+tempCD.CDname+"/r/n");
                    }
                }
            }
            out.flush();
            out.close();
        }
        catch(FileNotFoundException e)
        {
            System.out.println("找不到文件");
            e.printStackTrace();
        }
        catch(IOException e)
        {
            System.out.println("输出流出错");
            e.printStackTrace();
        }
    }
}
}
```

四、CD 类

用于描述每张 CD 的基本信息

含有 int CDID（每张 CD 都不同，用于区分 CD）、String CDname、boolean rented（用于辨别该 CD 是否已被租出）、String borrower（借走该 CD 的会员）、Date duedate（应还时间）一共 5 个变量

两个构造方法，一个默认构造方法和用 CDID 及 CDname 初始化的构造方法

两个其它方法，一个用于检查 CD 是否需要归还的方法 boolean isExpire，另一个用于判断相等的方法 boolean equals

```
class CD{
    int CDID;
    String CDname;
    boolean rented;
    String borrower;
    Date duedate;
    public CD() {}
```

```
    public CD(int theID,String thename)
    {
        this.CDID=theID;
        this.CDname=thename;
    }
    public boolean isExpire()
    {
        Date now=new Date();
        if(this.duedate.before(now))
            return true;
        else
            return false;
    }
    public boolean equals(CD theCD)
    {
        if(this.CDID==theCD.CDID&&this.CDname.equals(theCD.CDname))
            return true;
        else
            return false;
    }
    }
```

五、Member 类

用于描述会员的基本信息

包括 int ID（每个会员不同，用于区分会员）、String name、ArrayList<CD> rentCD（存储会员借走的所有 CD）三个变量

含有两个方法，方法 boolean needReturn 用于查询该会员是否有需要归还的 CD，方法 equals 用于判断两个 Member 型变量是否相等（因为要求每个人都唯一对应一个 ID 号，因此 ID 号相同即可认为两个变量相等）

```
class Member{
    int ID;
    String name;
    ArrayList<CD> rentCD;
    public boolean needReturn()
    {
        for(int i=0;i<rentCD.size();i++)
        {
            if(rentCD.get(i).isExpire())
                return true;
        }
        return false;
    }
    public boolean equals(Member thememberr)
    {
        if(this.ID==thememberr.ID)
```

```
            return true;
        else
            return false;
    }
    }
```

六、MyException 类

自定义异常

有一个默认构造函数及一个用 String 初始化的构造函数

```
class MyException extends Exception
{
    public MyException()
    {
        super();
    }
    public MyException(String message)
    {
        super(message);
    }
    }
```

七、CDstore 类

存储 ArrayList<CD> CDinventory 以及 ArrayList<Member> memberlist 两个链式数组，也就是 CD 的库存表以及会员的名单

有多个对名单或者库存表进行操作的方法

1、 **public void** addMember(Member thememember)

用于添加新的会员，如果会员的 ID 号重复，则抛出异常，也可以在会员 ID 号缺省的情况下自动生成 ID 号并添加

```
public void addMember(Member thememember){
        if(thememember.ID==0)
        {
            thememember.ID=this.memberlist.size()+1;
            this.memberlist.add(thememember);
        }
        else
        {
            try
            {
                for(int i=0;i<memberlist.size();i++)
                {
                    if(this.memberlist.get(i).ID==thememember.ID)
                    {
                        throw new MyException("该ID号已存在");
                    }
                }
```

```
            }
            catch(MyException repeat)
            {
                System.out.println(repeat);
                return ;
            }
            this.memberlist.add(thememember);
        }
        }
```

2、 **public void** deleteMember(Member thememember)

用来删除某个会员

如果会员不存在于会员表 memberlist 中，抛出异常

使用迭代器来遍历删除会员，用 boolean 型变量 existMember 来代表变量是否存在

```
    public void deleteMember(Member thememember){
        try{
            boolean existMember=false;
            Iterator<Member> theListIterator=this.memberlist.iterator();
            while(theListIterator.hasNext())
            {
                Member toDelete=theListIterator.next();
                if(toDelete.equals(thememember))
                {
                    theListIterator.remove();
                    existMember=true;
                }
            }
            if(!existMember)
                throw new MyException("该会员不在记录中");
        }
        catch(MyException e)
        {
            System.out.println(e);
        }
        }
```

3、 **public void** rentCD(CD theCD,Member thememember)

借出 CD，传入要借出的 CD 以及要借走 CD 的会员

查询该 CD 是否存在，如果 CD 不存在就抛出一个异常，同时通过 rented 变量查看查询 CD 是否已被租出，如果 CD 已被租出，输出“已出租”以及借出人 borrower 和应该归还的日期 duedate，如果 CD 未租出，查询会员是否在会员名单中，如果不在，则抛出异常，当 catch 到该异常时，新建一个会员对象并将该对象插入会员表 memberlist 中，如果存在，则将该 CD 的 rented 改为 true，在会员的已借 CD 列表 ArrayList<CD> rentCD 中插入一个 CD

```
public void rentCD(CD theCD,Member thememember){
```

```
        try{
            boolean beenrented=false;
            boolean CDexist=false;
            boolean memberexist=false;
            int findCD=0;
            for(int i=0;i<this.CDinventory.size();i++)
            {
                if(this.CDinventory.get(i).equals(theCD))
                {
                    findCD=i;
                    if(this.CDinventory.get(i).rented)
                    {
                        SimpleDateFormat dateformat=new
SimpleDateFormat("yyyy-MM-dd");
                        System.out.println("CD已出租");
                        System.out.println("CD已出租"+theCD.borrower+"
"+dateformat.format(theCD.duedate));//输出日期有错
                        beenrented=true;
                    }
                    CDexist=true;
                }
            }
            if(CDexist==false)
            {
                throw new MyException("不存在该CD");
            }
            else if(beenrented==false)
            {
                this.CDinventory.get(findCD).rented=true;
                try{

                    for(int i=0;i<this.memberlist.size();i++)
                    {
                        if(this.memberlist.get(i).equals(thememder))
                        {

    this.memberlist.get(i).rentCD.add(this.CDinventory.get(findCD));
                            memberexist=true;
                        }
                    }
                    if(memberexist==false)
                        throw new MyException("该会员不存在");
                }
                catch(MyException memberExc)
```

```
                {
                    System.out.println(memberExc);
                    System.out.println("请输入会员信息");
                    Member newmember=new Member();
                    Scanner input=new Scanner(System.in);
                    newmember.name=input.next();
                    newmember.rentCD.add(theCD);
                    this.addMember(newmember);//此处用size生成ID号
                }
                System.out.println("操作成功");
            }
        }
        catch(MyException e)
        {
            System.out.print(e);
        }
    }
```

4、 **public void** saleCD(CD theCD)

出售 CD

遍历 CD 库存表 CDinventory 用 boolean CDexist 代表 CD 是否存在，如果不存在就抛出异常，在遍历过程中同时判断该 CD 是否被借出，如果被借出也抛出一个异常，如果没有被借出，就将 CD 库存中关于这张 CD 的记录删除

```
public void saleCD(CD theCD){
        try{
            boolean CDexist=false;
            Iterator<CD> CDiterator=this.CDinventory.iterator();
            while(CDiterator.hasNext())
            {
                CD toDelete=CDiterator.next();
                if(toDelete.equals(theCD))
                {
                    CDexist=true;
                    if(toDelete.rented==true)
                        throw new MyException("该CD已被借出");
                    else
                        CDiterator.remove();
                }
            }
            if(CDexist==false)
                throw new MyException("该CD已售出或不存在");
        }
        catch(MyException e)
        {
```

```
            System.out.println(e);
        }
    }
```

5、 **public void** purchaseCD(Scanner in)

购买 CD

将输入的 CD 信息插入 CD 库存表 CDinventory 中，并且确保 CDID 不能重复，如果有重复的 CDID，抛出一个异常

```
public void purchaseCD(Scanner in){
        String end=null;
        System.out.println("请输入购入的CD信息，结束请输入end");
        while(end!="end")
        {
            System.out.println("请输入购入CD的ID号");
            int theID=in.nextInt();
            System.out.println("请输入购入CD的名称");
            String thename=in.next();
            end=thename;
            try
            {
                for(int i=0;i<this.CDinventory.size();i++)
                {
                    if(this.CDinventory.get(i).CDID==theID)
                        throw new MyException("CD编号重复");
                }
            }
            catch(MyException e)
            {
                System.out.println(e);
                return;
            }
            CD broughtCD= new CD(theID,thename);
            this.CDinventory.add(broughtCD);
        }


        }
```

八、未实现的部分

1、 查询某个 CD 或 Member 对象，给出部分信息，比如序列号或者名字，查询出符合条件的所有对象，可以生成逾期未还的会员的名单等等，可能可以通过传递泛型参数来实现

2、 按文件输入购买的 CD，将文件路径名当作参数传入文件，将数据添加到 ArrayList<CD>CDinventory 中，最后在所有操作完成后将 CDinventory 中的数据全部写到 CD 库存文件中

3、 Main 主类比较简陋，没有实现保存文件的部分

九、源码

```
package lab5;
import java.util.*;
import java.util.Date;
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.io.File;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class Main{
    public static void main(String[] arg){
        Scanner out=new Scanner(System.in);
        String filepath=out.next();
        Readfile CDs=new Readfile();
        CDs.readFile(filepath);
        int require=out.nextInt();
        Member thememberr=new Member();
        CD theCD=new CD();
        if(require==1)
        {
            System.out.println("请输入会员ID号");
            thememberr.ID=out.nextInt();
            System.out.println("请输入会员姓名");
            thememberr.name=out.next();
            String end=null;
            System.out.println("请输入会员借出的CD信息，结束请输入end");
            while(end!="end")
            {
                System.out.println("请输入借出人CD姓名");
                theCD.borrower=out.next();
                System.out.println("请输入借出CD的ID号");
                theCD.CDID=out.nextInt();
                System.out.println("请输入借出CD的名称");
                theCD.CDname=out.next();
                thememberr.rentCD.add(theCD);
                end=theCD.CDname;
            }
```

```
            CDs.theCDstore.addMember(thememember);
        }
        if(require==2)
        {
            System.out.println("请输入会员ID号");
            thememember.ID=out.nextInt();
            CDs.theCDstore.deleteMember(thememember);
        }
        if(require==3)
        {
            System.out.println("请输入会员ID号");
            thememember.ID=out.nextInt();
            System.out.println("请输入CDID号");
            theCD.CDID=out.nextInt();
            System.out.println("请按照如“2018-01-01”的格式输入CD的应还日期");
            SimpleDateFormat dateformat=new SimpleDateFormat("yyyy-MM-dd");
            String theDate=out.next();
            try {
                theCD.duedate=dateformat.parse(theDate);
            } catch (ParseException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            CDs.theCDstore.rentCD(theCD, thememember);
        }
        if(require==4)
        {
            System.out.println("请输入CDID号");
            theCD.CDID=out.nextInt();
            CDs.theCDstore.saleCD(theCD);
        }
        if(require==5)
        {
            CDs.theCDstore.purchaseCD(out);
        }
    }
}
class Readfile
{
    CDstore theCDstore;
    int readType;//区别读入的文件是CD库存文件还是会员名单
    public void readFile(String filepath){
        File file=new File(filepath);
        if(file.exists()){
```

```
        try
        {
            FileReader fileReader=new FileReader(file);
            BufferedReader buffreader=new BufferedReader(fileReader);
            String line=null;
            while((line=buffreader.readLine())!=null)//逐行读取
            {
                StringTokenizer token=new StringTokenizer(line," ");
                if(this.readType==1)
                    this.inputCD(token);//将输入的数据存储为CD形式
                else if(this.readType==2)
                    this.inputMember(token);//将输入的数据存储为Member形式
            }
            buffreader.close();
            fileReader.close();
        }
        catch(FileNotFoundException e)
        {
            System.out.println("找不到文件");
            e.printStackTrace();
        }
        catch(IOException e)
        {
            System.out.println("输入流出错");
            e.printStackTrace();
        }
    }
}
public void inputCD(StringTokenizer token)//将读取的数据存成CD
{
    SimpleDateFormat dateformat=new SimpleDateFormat("yyyy-MM-dd");
    String theDate="";
    String str1=token.nextToken();
    String str2=token.nextToken();
    int theID=Integer.parseInt(str1);
    CD theCD=new CD(theID,str2);
    theCD.borrower=token.nextToken();
    theDate=token.nextToken();
    try {
        theCD.duedate=dateformat.parse(theDate);
    } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

```
            this.theCDstore.CDinventory.add(theCD);
        }
        public void inputMember(StringTokenizer token)//将读取的数据存成Member
        {
            Member thememberr=new Member();
            String str1=token.nextToken();
            thememberr.ID=Integer.parseInt(str1);
            thememberr.name=token.nextToken();
            String str2="";
            String str3="";
            while(token.hasMoreTokens())
            {
                str2=token.nextToken();
                str3=token.nextToken();
                int theID=Integer.parseInt(str2);
                CD theCD=new CD(theID,str3);
                thememberr.rentCD.add(theCD);
            }
            this.theCDstore.memberlist.add(thememberr);

        }
    }
    class Writefile
    {
        CDstore theCDstore;
        int writeType;
        public void writeFile(String filepath)
        {
            File file=new File(filepath);
            if(file.exists()){
                try{
                    FileWriter fileWriter=new FileWriter(file);
                    BufferedWriter out=new BufferedWriter(fileWriter);
                    if(this.writeType==1)
                    {
                        for(int i=0;i<this.theCDstore.CDinventory.size();i++)
                        {
                            CD theCD=new CD();
                            theCD=this.theCDstore.CDinventory.get(i);
                            out.write(theCD.CDID+" "+theCD.CDname+"
    "+theCD.rented+" "+theCD.borrower+" "+theCD.duedate+"/r/n");
                        }

                    }
```

```
                if(this.writeType==2)
                {
                    for(int i=0;i<this.theCDstore.memberlist.size();i++)
                    {
                        Member thememberr=new Member();
                        thememberr=this.theCDstore.memberlist.get(i);
                        out.write(thememberr.ID+" "+thememberr.name+" ");
                        for(int j=0;j<thememberr.rentCD.size();i++)
                        {
                            CD tempCD=new CD();
                            tempCD=thememberr.rentCD.get(i);
                            out.write(tempCD.CDID+" "+tempCD.CDname+"/r/n");
                        }
                    }
                }
                out.flush();
                out.close();
            }
            catch(FileNotFoundException e)
            {
                System.out.println("找不到文件");
                e.printStackTrace();
            }
            catch(IOException e)
            {
                System.out.println("输出流出错");
                e.printStackTrace();
            }
        }
    }
}
class CD{
    int CDID;
    String CDname;
    boolean rented;
    String borrower;
    Date duedate;
    public CD() {}
    public CD(int theID,String thename)
    {
        this.CDID=theID;
        this.CDname=thename;
    }
    public boolean isExpire()
```

```
    {
        Date now=new Date();
        if(this.duedate.before(now))
            return true;
        else
            return false;
    }
    public boolean equals(CD theCD)
    {
        if(this.CDID==theCD.CDID&&this.CDname.equals(theCD.CDname))
            return true;
        else
            return false;
    }
}
class Member{
    int ID;
    String name;
    ArrayList<CD> rentCD;
    public boolean needReturn()
    {
        for(int i=0;i<rentCD.size();i++)
        {
            if(rentCD.get(i).isExpire())
                return true;
        }
        return false;
    }
    public boolean equals(Member thememberr)
    {
        if(this.ID==thememberr.ID)
            return true;
        else
            return false;
    }
}
class MyException extends Exception
{
    public MyException()
    {
        super();
    }
    public MyException(String message)
    {
```

```
        super(message);
    }
}
class CDstore{
    ArrayList<CD> CDinventory;
    ArrayList<Member> memberlist;

    public void addMember(Member themember){
        if(themember.ID==0)
        {
            themember.ID=this.memberlist.size()+1;
            this.memberlist.add(themember);
        }
        else
        {
            try
            {
                for(int i=0;i<memberlist.size();i++)
                {
                    if(this.memberlist.get(i).ID==themember.ID)
                    {
                        throw new MyException("该ID号已存在");
                    }
                }

            }
            catch(MyException repeat)
            {
                System.out.println(repeat);
                return ;
            }
            this.memberlist.add(themember);
        }
    }

    public void deleteMember(Member themember){
        try{
            boolean existMember=false;
            Iterator<Member> theListIterator=this.memberlist.iterator();
            while(theListIterator.hasNext())
            {
                Member toDelete=theListIterator.next();
                if(toDelete.equals(themember))
                {
```

```
                    theListIterator.remove();
                    existMember=true;
                }
            }
            if(!existMember)
                throw new MyException("该会员不在记录中");
        }
        catch(MyException e)
        {
            System.out.println(e);
        }
    }
    public void rentCD(CD theCD,Member thememberr){
        try{
            boolean beenrented=false;
            boolean CDexist=false;
            boolean memberexist=false;
            int findCD=0;
            for(int i=0;i<this.CDinventory.size();i++)
            {
                if(this.CDinventory.get(i).equals(theCD))
                {
                    findCD=i;
                    if(this.CDinventory.get(i).rented)
                    {
                        SimpleDateFormat dateformat=new
SimpleDateFormat("yyyy-MM-dd");
                        System.out.println("CD已出租");
                        System.out.println(theCD.borrower+"
"+dateformat.format(theCD.duedate));
                        beenrented=true;
                    }
                    CDexist=true;
                }
            }
            if(CDexist==false)
            {
                throw new MyException("不存在该CD");
            }
            else if(beenrented==false)
            {
                this.CDinventory.get(findCD).rented=true;
                try{
```

```
                for(int i=0;i<this.memberlist.size();i++)
                {
                    if(this.memberlist.get(i).equals(thememember))
                    {

this.memberlist.get(i).rentCD.add(this.CDinventory.get(findCD));
                        memberexist=true;
                    }
                }
                if(memberexist==false)
                    throw new MyException("该会员不存在");
            }
            catch(MyException memberExc)
            {
                System.out.println(memberExc);
                System.out.println("请输入会员信息");
                Member newmember=new Member();
                Scanner input=new Scanner(System.in);
                newmember.name=input.next();
                newmember.rentCD.add(theCD);
                this.addMember(newmember);//此处用size生成ID号
            }
            System.out.println("操作成功");
        }
    }
    catch(MyException e)
    {
        System.out.print(e);
    }
}
public void saleCD(CD theCD){
    try{
        boolean CDexist=false;
        Iterator<CD> CDiterator=this.CDinventory.iterator();
        while(CDiterator.hasNext())
        {
            CD toDelete=CDiterator.next();
            if(toDelete.equals(theCD))
            {
                CDexist=true;
                if(toDelete.rented==true)
                    throw new MyException("该CD已被借出");
                else
                    CDiterator.remove();
```

```
                }
            }
            if(CDexist==false)
                throw new MyException("该CD已售出或不存在");
        }
        catch(MyException e)
        {
            System.out.println(e);
        }
    }
    public void purchaseCD(Scanner in){
        String end=null;
        System.out.println("请输入购入的CD信息，结束请输入end");
        while(end!="end")
        {
            System.out.println("请输入购入CD的ID号");
            int theID=in.nextInt();
            System.out.println("请输入购入CD的名称");
            String thename=in.next();
            end=thename;
            try
            {
                for(int i=0;i<this.CDinventory.size();i++)
                {
                    if(this.CDinventory.get(i).CDID==theID)
                        throw new MyException("CD编号重复");
                }
            }
            catch(MyException e)
            {
                System.out.println(e);
                return;
            }
            CD broughtCD= new CD(theID,thename);
            this.CDinventory.add(broughtCD);
        }

    }

}
```