Group 6

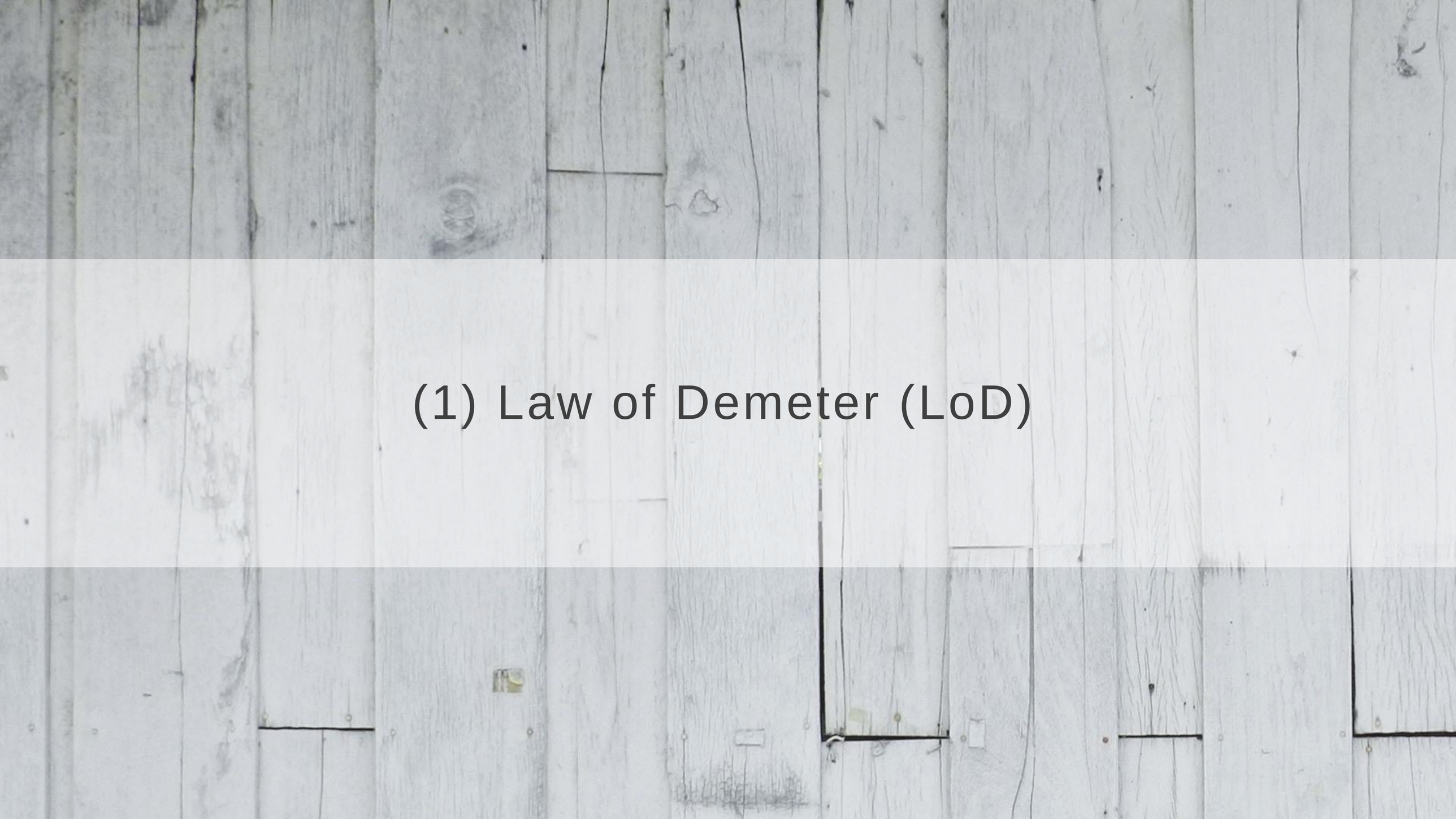
The Library Management System

Team work 2

A10523006	Maggie
A10523049	Peggy
B10423003	Kurumi
B10423029	Bean
B10523020	Kendy
B10523030	Jerry
B10523053	Lynn
M10723001	Joe

Content :)

- (1) Law of Demeter (LoD)
- (2) Coupling
- (3) Cohesion
- (4) Connascence
- (5) CRC card
- (6) Contract, Method specification, Activity diagram
- (7) Data coupling, Cohesion, Convention connascence
- (8) Referential integrity
- (9) Normalizationions
- (10) Denormalize and New Class diagram
- (11) Inter-file clustering and Index strategies



(1) Law of Demeter (LoD)

1-1. to itself (0 itself)

class InitialController

```
package librarySystem;

import java.util.ArrayList;
public class InitialController {
    @FXML
    private TextField UserID;
    @FXML
    private TextField Password;
    @FXML
    private Button LoginButton;
    @FXML
    private Button GuestButton;
    public void onButtonClick(ActionEvent event)
    {
        String ID=UserID.getText();
        String password=Password.getText();
        ArrayList<Librarian> storeLibrarian=new ArrayList<Librarian>();
        ArrayList<Member> storeMember=new ArrayList<Member>();
        Librarian checkLibrairan = new Librarian();
        Member checkMember = new Member();
        try
        {
        char beginChar=ID.charAt(0);
        if(beginChar=='L')
        {
            storeLibrarian=LibraryDBMgr.searchData(ID,"librarian");
            checkLibrairan=storeLibrarian.get(0);
            if(password.equals(checkLibrairan.getlibrarianPassword()))
            {
                tiggerLibrarianGUI();
                final Node source = (Node) event.getSource();
                final Stage stage = (Stage) source.getScene().getWindow();
                stage.close();
            }
```

```
public void tiggerSearchGUI()
{

}
public void tiggerLibrarianGUI()
{
    try
    {
        LibrarianGUI librarianGUI=new LibrarianGUI();
        librarianGUI.showWindow();
    }catch(Exception e)
    {
    }
}
```

If password equals data's record, execute its own method.

1-2. Any objects created/instantiated within M

class LibrarianController

```
public void editMemberButtonClick(ActionEvent event)
{
    ArrayList<Member> storeMembers=new ArrayList<Member>();
    Member haveMember=new Member();
    JFrame editMemberFrame=new JFrame("修改member介面");
    JPanel p= new JPanel();
    editMemberFrame.setDefaultLookAndFeelDecorated(true);
    String input=JOptionPane.showInputDialog("輸入memberID");
    if(input!=null||input!="")
    {
        try {
            storeMembers=LibraryDBMgr.searchData(input,"member");
        }catch(Exception e)
        {
    //
        }
            haveMember=storeMembers.get(0);
            if(haveMember.getmemberID()==null||haveMember.getmemberID()=="")
            {
                JOptionPane.showMessageDialog(editMemberFrame,"can not find member",
                        "Error", JOptionPane.ERROR_MESSAGE);
            }else
            {

                System.out.print(input+"in");
                JButton b1 = new JButton("修改");
                JTextField ID = new JTextField(haveMember.getmemberID(),15);
                JTextField memberPassword = new JTextField(haveMember.getmemberPassword(),15);
                JTextField IMN = new JTextField(haveMember.getmemberName(),15);
```

1-3. to an object that is passed as a parameter to the method (M’s parameters)

class LibrarianDBMgr

```
package librarySystem;
import java.sql.*;
public class LibraryDBMgr {
    public static void editData(String editID,ArrayList input,String editTable)
    {
        ArrayList<Librarian> ALibrarian=new ArrayList<Librarian>();
        ArrayList<Member> AMember;
        ArrayList<EBook> AEBook;
        ArrayList<PaperBook> APaperBook=new ArrayList<PaperBook>();
        Connection conn = null;
            try {
                    Class.forName("com.mysql.jdbc.Driver");
                    String datasource="jdbc:mysql://localhost/library?user=kendy&password=ken033580964";
                    conn = DriverManager.getConnection(datasource);
                    System.out.println("成功");
                    Statement st = conn.createStatement();

                    }
                    else if(editTable.equals("member"))
                    {
                        Member storeMember;
                        AMember=input;
                        storeMember=AMember.get(0);
                        System.out.println("成功載入MEMBER並給值");
                        boolean i=storeMember.getright();
                        int x;
                        if(i)
                        {
                            x=1;
                        }

 System.out.println(storeMember.getmemberID()+storeMember.getmemberPassword()+storeMember.getmemberemail()+storeMember.getnumbe
 String SQL = String.format("UPDATE member SET memberID='%s',memberPassword='%s',memberName='%s',memberRepublicofChinaNational
         ,storeMember.getmemberID()
         ,storeMember.getmemberPassword()
         ,storeMember.getmemberName()
         ,storeMember.memberRepublicofChinaNationalID()
         ,storeMember.getmemberemail()
         ,storeMember.getnumberOfBorrowBook()
         ,storeMember.getnumberOfOverdueBook()
         ,storeMember.getnumberOfNoticeBook()
         ,x
         ,editID
         );
 st.executeUpdate(SQL);
 System.out.println("成功寫入MEMBER");
```

1-4. to an object that is created by the method (0's direct component objects)

class InitialController

```
package librarySystem;

import java.util.ArrayList;
public class InitialController {
    @FXML
    private TextField UserID;
    @FXML
    private TextField Password;
    @FXML
    private Button LoginButton;
    @FXML
    private Button GuestButton;
    public void onButtonClick(ActionEvent event)
    {
        String ID=UserID.getText();
        String password=Password.getText();
```

(2) Coupling

2-1. Control Coupling

class InitialController

```
public void onButtonClick(ActionEvent event)
{
    String ID=UserID.getText();
    String password=Password.getText();
    ArrayList<Librarian> storeLibrarian=new ArrayList<Librarian>
    ArrayList<Member> storeMember=new ArrayList<Member>();
    Librarian checkLibrairan = new Librarian();
    Member checkMember = new Member();
    try
    {
    char beginChar=ID.charAt(0);
    if(beginChar=='L')
    {
        storeLibrarian=LibraryDBMgr.searchData(ID,"librarian");
```

```
public static ArrayList searchDat(String inputID,String usefunction) throws Exception
{
...
if(usefunction.equals("librarian"))//搜尋librarian
{
    Librarian r1=new Librarian();
    String SQL = String.format("SELECT * FROM librarian Where LibrarianID = '%s' ",inputID);
```

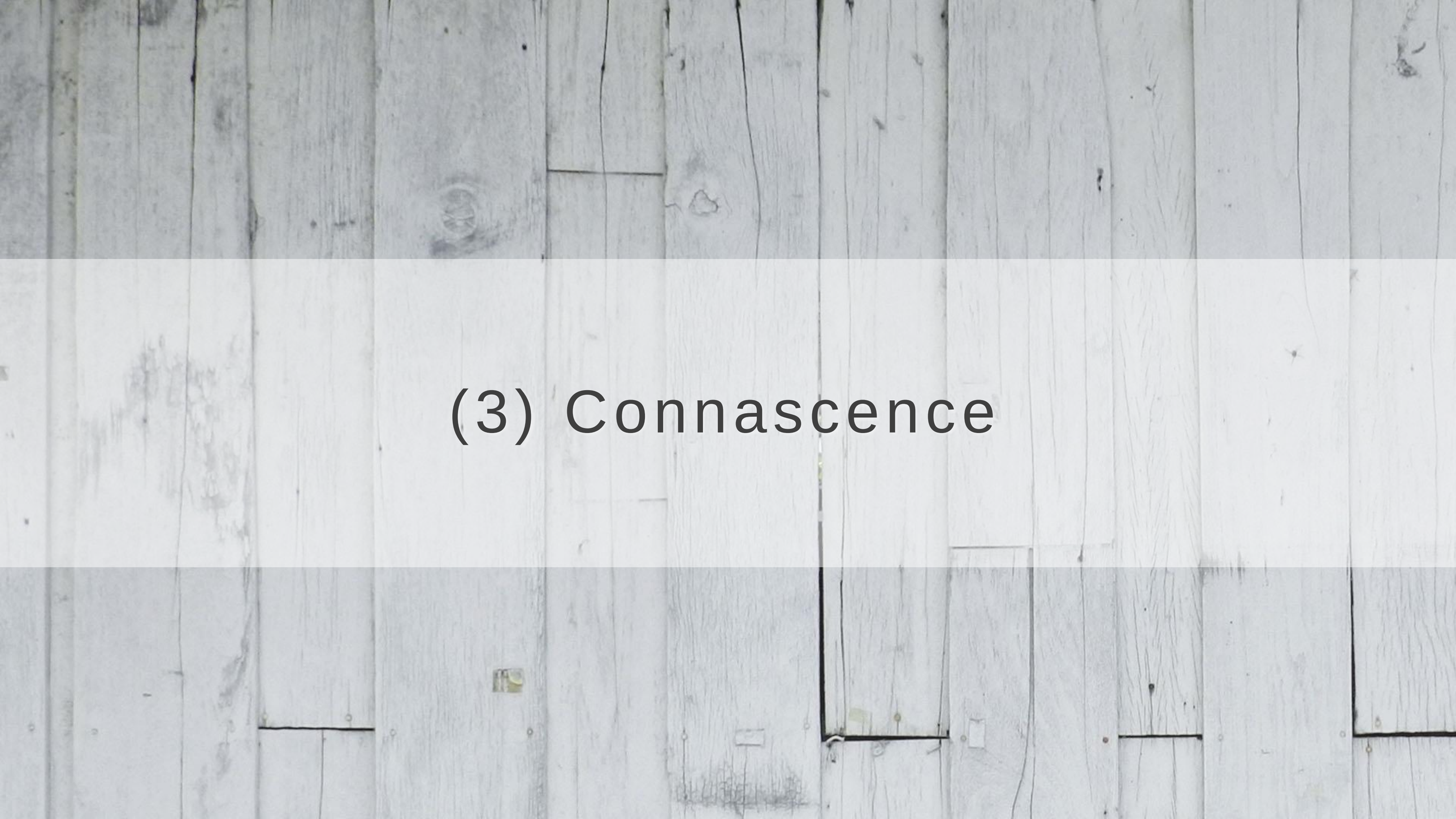
2-2. Interaction, Data Coupling

```
package librarySystem;
import java.util.Properties;
public class SendEmail {
    public static void send(String from,String password,String to,String sub,String msg){
        //Get properties object
        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class",
                  "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        //get Session
        Session session = Session.getDefaultInstance(props,
         new javax.mail.Authenticator() {
         protected PasswordAuthentication getPasswordAuthentication() {
         return new PasswordAuthentication(from,password);
         }
        });
        //compose message
        try {
         MimeMessage message = new MimeMessage(session);
         message.addRecipient(Message.RecipientType.TO,new InternetAddress(to));
         message.setSubject(sub);
         message.setText(msg);
         //send message
         Transport.send(message);
         System.out.println("message sent successfully");
        } catch (MessagingException e) {throw new RuntimeException(e);}
    }
}
```

2-3. Stamp Coupling

class LibrarianController

```
public void editMemberButtonClick(ActionEvent event)
{
    ArrayList<Member> storeMembers=new ArrayList<Member>();
    Member haveMember=new Member();
    JFrame editMemberFrame=new JFrame("修改member介面");
    JPanel p= new JPanel();
    editMemberFrame.setDefaultLookAndFeelDecorated(true);
    String input=JOptionPane.showInputDialog("請輸入memberID");
    if(input!=null||input!="")
    {
        try {
            storeMembers=LibraryDBMgr.searchData(input,"member");
        }catch(Exception e)
        {
    //
        }
            haveMember=storeMembers.get(0);
            if(haveMember.getmemberID()==null||haveMember.getmemberID()=="")
            {
                JOptionPane.showMessageDialog(editMemberFrame,"can not find member",
                        "Error", JOptionPane.ERROR_MESSAGE);
            }else
            {

                System.out.print(input+"in");
                JButton b1 = new JButton("修改");
                JTextField ID = new JTextField(haveMember.getmemberID(),15);
                JTextField memberPassword = new JTextField(haveMember.getmemberPassword(),15);
                JTextField IMN = new JTextField(haveMember.getmemberName(),15);
```



(3) Connascence

3-1. Function Cohesion

class CheckOverdueBook - 1

```
package librarySystem;
import java.util.Date;
public class CheckOverdueBook extends TimerTask{
    public void run() {
        System.out.println("执用");
       ArrayList getCheckBook =new ArrayList();
       java.text.SimpleDateFormat format = new java.text.SimpleDateFormat("yyyy-MM-dd");
       long day = 0;
       Date now =new Date();
       try {
       getCheckBook=LibraryDBMgr.searchData(null, "CheckOverdueBook");
       }catch(Exception e)
       {}
       ArrayList<PaperBook> PBA =new ArrayList<PaperBook>();
       PBA=getCheckBook;
       for (int i = 0; i < getCheckBook.size(); i++)
       {

           PaperBook store = new PaperBook();
           store=PBA.get(i);
           String borrowTimeString = store.getborrowerTime();
```

class CheckOverdueBook - 2

```
{
    java.util.Date beginDate = format.parse(borrowTimeString);
    day=(now.getTime() - beginDate.getTime())/(24*60*60*1000);
    System.out.println(day);
}catch(Exception e)
{
    System.out.println(e);
}
if(day>4 && day<=7)
{
    System.out.println("day<7");
    ArrayList getMSearch =new ArrayList();
    ArrayList<Member> haveM =new ArrayList<Member>();
    ArrayList SEP =new ArrayList();
    ArrayList SEM =new ArrayList();
    ArrayList<Member> OMAL =new ArrayList<Member>();
    Member OM = new Member();
    store=PBA.get(i);
    store.setbookState("notice");
    try
    {
```

3-1. Function Cohesion

class CheckOverdueBook - 3

```
    {
    getMSearch=LibraryDBMgr.searchData(store.getborrower(),"member");
    }catch(Exception e)
    {}
    haveM=getMSearch;
    OM=haveM.get(0);
    OM.setnumberOfBorrowBook(OM.getnumberOfBorrowBook()-1);
    OM.setnumberOfNoticeBook(OM.getnumberOfNoticeBook()+1);
    SEP.add(store);
    SEM.add(OM);
    LibraryDBMgr.editData(store.getbookID(),SEP,"paperbook");
    LibraryDBMgr.editData(OM.getmemberID(),SEM,"member");
}
else if(day>7)
{
    System.out.println("day>7");
    ArrayList getMSearch =new ArrayList();
    ArrayList<Member> haveM =new ArrayList<Member>();
    ArrayList SEP =new ArrayList();
    ArrayList SEM =new ArrayList();
    ArrayList<Member> OMAL =new ArrayList<Member>();
```

class CheckOverdueBook - 4

```
        store=PBA.get(i);
        Member OM = new Member();
        store.setbookState("overdue");
        try
        {
        getMSearch=LibraryDBMgr.searchData(store.getborrower(),"member");
        }catch(Exception e)
        {}
        haveM=getMSearch;
        OM=haveM.get(0);
        OM.setnumberOfNoticeBook(OM.getnumberOfNoticeBook()-1);
        OM.setnumberOfOverdueBook(OM.getnumberOfOverdueBook()+1);
        OM.setright(false);
        SEP.add(store);
        SEM.add(OM);
        LibraryDBMgr.editData(store.getbookID(),SEP,"paperbook");
        LibraryDBMgr.editData(OM.getmemberID(),SEM,"member");
    }
}
```

class CheckOverdueBook - 5

```
        }
    }
    ArrayList getOMA =new ArrayList();
    ArrayList<Member> getOM =new ArrayList<Member>();
    try {
        getOMA=LibraryDBMgr.searchData(null, "GOM");
        }catch(Exception e)
         {}
    getOM=getOMA;
    for (int i = 0; i < getOM.size(); i++)
    {
        Member sendNoticeEmail = new Member();
        sendNoticeEmail=getOM.get(i);
        SendEmail.send(sendNoticeEmail);
    }
}
```

3-2. Temporal Cohesion

class Time

```
10          public Time() {
11              Calendar calendar = Calendar.getInstance();
12              calendar.set(Calendar.HOUR_OF_DAY,1);
13              calendar.set(Calendar.MINUTE,0);
14              calendar.set(Calendar.SECOND,0);
15
```

3-3. Logical Chhesion-

class LibraryDBMgr

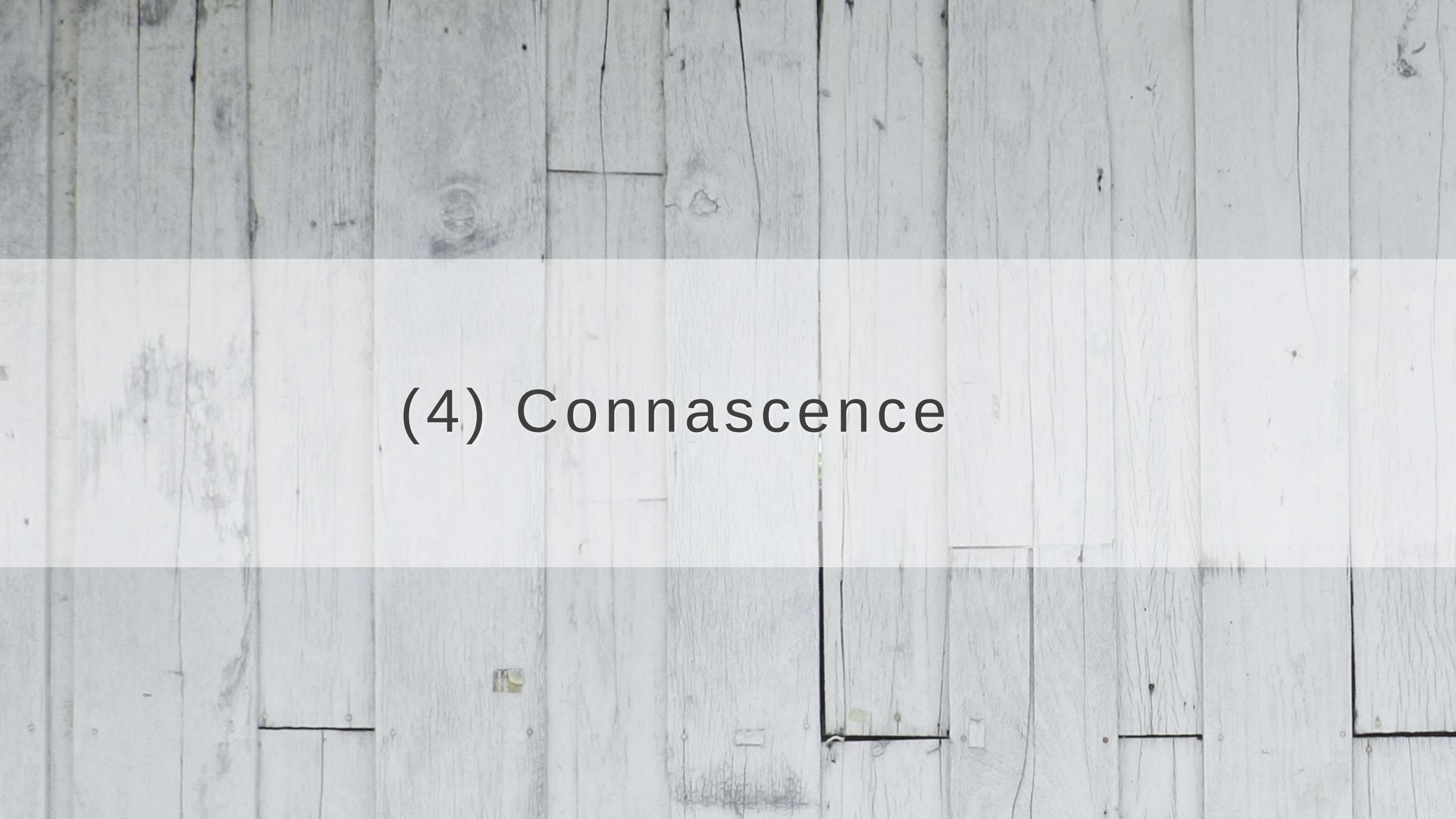
```
public static void addData(ArrayList input,String addTable)throws Exception
{
    ArrayList<Librarian> ALibrarian=new ArrayList<Librarian>();
    ArrayList<Member> AMember=new ArrayList<Member>();
    ArrayList<Ebook> AEbook=new ArrayList<Ebook>();
    ArrayList<PaperBook> APaperBook=new ArrayList<PaperBook>();
    Connection conn = null;
        try {
                Class.forName("com.mysql.jdbc.Driver");
                String datasource="jdbc:mysql://localhost/library?user=kendy&password=ken033580964";
                conn = DriverManager.getConnection(datasource);
                System.out.println("成功");
                Statement st = conn.createStatement();
                if(addTable.equals("librarian"))
                {
                    Librarian storeLibrarian=new Librarian();
                    ALibrarian=input;
                    storeLibrarian=ALibrarian.get(0);
                    String SQL = String.format("INSERT INTO librarian VALUES ('%s', '%s')",toreLibrarian.getlibrairanID(),s
                    st.execute(SQL);
                    st.close();
                }
                else if(addTable.equals("member"))
                {
                    Member storeMember=new Member();
                    AMember=input;
                    storeMember=AMember.get(0);
                    boolean i=storeMember.getright();
                    int x;
                    if(i)

        else if(addTable.equals("member"))
        {
            Member storeMember=new Member();
            AMember=input;
            storeMember=AMember.get(0);
            boolean i=storeMember.getright();
            int x;
            if(i)
            {
                x=1;
            }
            else
            {
                x=0;
            }
            System.out.println(storeMember.getmemberID()+storeMember.getmemberPassword()+storeMember.getmemberemail()+storeMember
            String SQL = String.format("INSERT INTO member (memberID,memberPassword,memberName,memberRepublicofChinaNationalID,e
                    ,storeMember.getmemberID()
                    ,storeMember.getmemberPassword()
                    ,storeMember.getmemberName()
                    ,storeMember.memberRepublicofChinaNationalID()
                    ,storeMember.getmemberemail()
                    ,storeMember.getnumberOfBorrowBook()
                    ,storeMember.getnumberOfOverdueBook()
                    ,storeMember.getnumberOfNoticeBook()
                    ,x
                    );
            st.execute(SQL);
            st.close();
        }
        else if(addTable.equals("paperbook"))
```

3-3. Logical Chhesion-2

```
        st.execute(SQL);
        st.close();
    }
    else if(addTable.equals("paperbook"))
    {
        PaperBook storePaperBook=new PaperBook();
        APaperBook=input;
        storePaperBook=APaperBook.get(0);
        int x=0;
        String sql = String.format("SELECT MAX(bookID) FROM paperbook");
        st.execute(sql);
        System.out.println("執行成功");
        ResultSet rs=st.getResultSet();
        System.out.println(rs);
        while(rs.next())
        {
            x=rs.getInt(1);
        }
        x=x+1;
        InitialGUI.setmaxID(x);
        PreparedStatement pstmt;
        String SQL = "INSERT INTO paperbook (bookID,bookTitle,author,publisher,publicationDate,summary,state,bo
        pstmt = conn.prepareStatement(SQL,Statement.RETURN_GENERATED_KEYS);
        pstmt.setInt(1,x);
        pstmt.setString(2,storePaperBook.getbookTitle());
        pstmt.setString(3,storePaperBook.getauthor());
        pstmt.setString(4,storePaperBook.getpublisher());
        pstmt.setString(5,storePaperBook.getpublicationDate());
        pstmt.setString(6,storePaperBook.getsummary());
```

```
    else if(addTable.equals("ebook"))
    {
        int rid;
        PreparedStatement pstmt ;
        Ebook storeEbook=new Ebook();
        AEbook=input;
        storeEbook=AEbook.get(0);
        String sql = String.format("SELECT MAX(bookID) FROM ebook");
        st.execute(sql);
        ResultSet rs=st.getResultSet();
        int x=0;
        while(rs.next())
        {
            x=rs.getInt(1);
        }
        x=x+1;
        InitialGUI.setmaxID(x);
        System.out.println(storeEbook.getbookTitle()+storeEbook.getauthor()+storeEbook.getpublisher()+storeEbook.getpublicati
        String SQL = "INSERT INTO ebook (bookID,bookTitle,author,publisher,publicationDate,summary,bookContext,bookType) VALU
        pstmt = conn.prepareStatement(SQL,Statement.RETURN_GENERATED_KEYS);
        pstmt.setInt(1,x);
        pstmt.setString(2,storeEbook.getbookTitle());
        pstmt.setString(3,storeEbook.getauthor());
        pstmt.setString(4,storeEbook.getpublisher());
        pstmt.setString(5,storeEbook.getpublicationDate());
        pstmt.setString(6,storeEbook.getsummary());
        pstmt.setString(7,storeEbook.getbookContext());
        pstmt.setString(8,storeEbook.getbookType());
        pstmt.execute();
```


```
    else if(addTable.equals("ebook"))
    {
        int rid;
        PreparedStatement pstmt ;
        Ebook storeEbook=new Ebook();
        AEbook=input;
        storeEbook=AEbook.get(0);
        String sql = String.format("SELECT MAX(bookID) FROM ebook");
        st.execute(sql);
        ResultSet rs=st.getResultSet();
        int x=0;
        while(rs.next())
        {
            x=rs.getInt(1);
        }
        x=x+1;
        InitialGUI.setmaxID(x);
        System.out.println(storeEbook.getbookTitle()+storeEbook.getauthor()+storeEbook.getpublisher()+storeEbook.getpublicati
        String SQL = "INSERT INTO ebook (bookID,bookTitle,author,publisher,publicationDate,summary,bookContext,bookType) VALU
        pstmt = conn.prepareStatement(SQL,Statement.RETURN_GENERATED_KEYS);
        pstmt.setInt(1,x);
        pstmt.setString(2,storeEbook.getbookTitle());
        pstmt.setString(3,storeEbook.getauthor());
        pstmt.setString(4,storeEbook.getpublisher());
        pstmt.setString(5,storeEbook.getpublicationDate());
        pstmt.setString(6,storeEbook.getsummary());
        pstmt.setString(7,storeEbook.getbookContext());
        pstmt.setString(8,storeEbook.getbookType());
        pstmt.execute();
```



(4) Connascence

4-1. Type of Class Connascence

class

```
public class LibrarianGUI extends Application{
    Stage stage=new Stage();
    public void start(Stage primaryStage) {
        try {
        FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("LibrarianGUI.fxml"));
            Parent root = (Parent) fxmlLoader.load();
            Scene scene = new Scene(root);
            primaryStage.setTitle("LibrarianGUI");
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
                e.printStackTrace();
            }
        }
    public static void main(String[] args) {

            launch(args);

        }
    public void  showWindow()
    {
    start(stage);
    }
}
```

4-2. Name Connascence

class Time

```
        Timer timer = new Timer();
        CheckOverdueBook task = new CheckOverdueBook();

        timer.schedule(task, date, PERIOD_DAY);
    }
```

4-3. Convention Connascence

class

```
public void borrowBookButtonClick(ActionEvent event) {
    JFrame borrowFrame = new JFrame("借書介面");
    JButton b1 = new JButton("借書");
    JTextField ID = new JTextField(15);
    JPanel p = new JPanel();
    p.add(new JLabel("輸入書的ID"));
    p.add(ID);
    p.add(b1);
    borrowFrame.add(p);
    borrowFrame.pack();
    borrowFrame.setDefaultLookAndFeelDecorated(true);
    String input = JOptionPane.showInputDialog("請輸入borrowerID");
    if (input == "" || input == null) {
        System.out.print(input + "fff");
        // borrowFrame.setDefaultCloseOperation(borrowFrame.EXIT_ON_CLOSE);
        // borrowFrame.setVisible(false);
        borrowFrame.dispose();
    } else {
        borrowFrame.setVisible(true);
        b1.addActionListener(ActionEvent -> {
            ArrayList<Member> storeMember = new ArrayList<Member>();
            Member checkMember = new Member();
            String table = new String("searchpaperbook");
            try {
                storeMember = LibraryDBMgr.searchData(input, "member");
                checkMember = storeMember.get(0);
                ArrayList<PaperBook> storePaperBook = new ArrayList<PaperBook>();
                PaperBook checkPaperBook = new PaperBook();
                storePaperBook = LibraryDBMgr.searchData(ID.getText(), table);
                checkPaperBook = storePaperBook.get(0);
```

(5) CRC Card

5. CRC Card

Front

Class name: LibrarianController	**ID:** 1	**Type:** Concrete, Domain
Description: This class provides librarian to save and edit data of memberships, paper book, and e-book. It also can help librarian search book information and provide book service.	**Association Use Case:** Manage Paper Book Manage E-Book Manage Member Borrow Book Return Book Search Book	
Responsibilities:	**Collaborators:**	
addMemberButtonClick	PaperBook	
addEbookButtonClick	Ebook	
addBookButtonClick	Member	
editMemberButtonClick	LibrarianGUI	
editEbookButtonClick	SendEmail	
editBookButtonClick	LibraryDBMgr	
deleteMemberButtonClick	Search	
deleteEbookButtonClick		
searchBookButtonClick		
borrowBookButtonClick		
returnBookButtonClick		
updatePaperBookStateButtonClick		

5. CRC Card

Back

Attributes:			
addMemberButton	(1..1)	(Button)	
addEbookButton	(1..1)	(Button)	
addBookButton	(1..1)	(Button)	
editMemberButton	(1..1)	(Button)	
editEbookButton	(1..1)	(Button)	
editBookButton	(1..1)	(Button)	
deleteMemberButton	(1..1)	(Button)	
deleteEbookButton	(1..1)	(Button)	
searchBookButton	(1..1)	(Button)	
borrowBookButton	(1..1)	(Button)	
returnBookButton	(1..1)	(Button)	
updatePaperBookStateButton	(1..1)	(Button)	
f	(0..1)	(File)	{f = (File) Actionevent.getNewValue()}
Relationships:			
Generalization(a-kind-of):			
Aggregation(has-parts):			
Other Associations:			
Manage Paper Book Manage E-Book Manage Member Borrow Book Return Book Search Book			

Text File

LibrarianController class invariants: F = (File) Actionevent.getNewValue()

(6) Contract
Method specification Activity diagram

6-1. Contract

Method Name: run()	**Class Name:** CheckOverdueBook	**ID:** 1
Client(consumers): Time		

Associated Use Case:

Member

Description of Responsibilities:

We use class checkOverdueBook to calculate if the book that borrowed by member is overdue or not.

Arguments Received:

day:long

Pre-Conditions:

day=(now.getTime() - beginDate.getTime())/(24*60*60*1000)

Post-Conditions:

```
if(day<=3)
        bookState = ("notice");
        setnumberOfNoticeBook -1
        getnumberOfNoticeBook +1

    else if(day<0)
        bookState = ("overdue");
        getnumberOfNoticeBook -1
        getnumberOfOverdueBook +1
        setright = (false)
```

6-2. Method Specificatio

Method Name: Time()	Class Name: Time	ID:
Contract ID:	Programmer: Kendy	Data Due: 05/28/2018

Programming Language:

Java

Triggers/Events:

CheckOverdueBook task = new CheckOverdueBook();

Arguments Received: Data Type:	Notes:
long	Borrowed day minus today

Messages Sent & Argument Passed: ClassName.MethodName:	Data Type:	Notes:

Arguments Returned: Data Type:	Notes:
void	

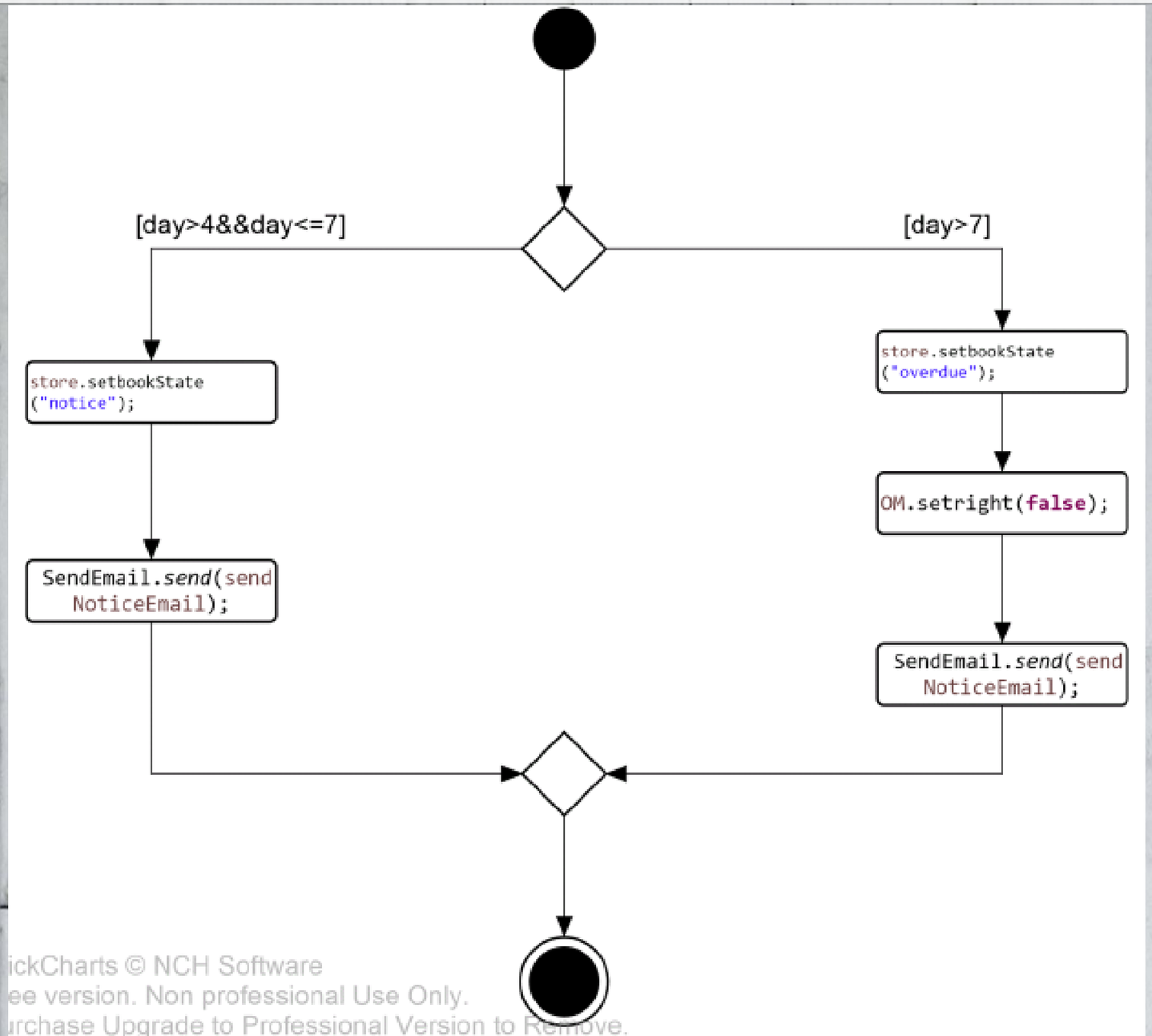
Algorithm Specification:

```
if(day<=3)
            {
             store.setbookState("notice");

             OM.setnumberOfNoticeBook(OM.getnumberOfBorrowBook()-1);
             OM.setnumberOfNoticeBook(OM.getnumberOfNoticeBook()+1);
             SEP.add(store);
             }
  else if(day<0)
            {
             store.setbookState("overdue");

             OM.setnumberOfNoticeBook(OM.getnumberOfNoticeBook()-1);
             OM.setnumberOfNoticeBook(OM.getnumberOfOverdueBook()+1);
             OM.setright(false);
            }
```

Misc.Notes:

None

6-3. Activity Diagram



(7) Data coupling
Cohesion
Convention connascence

7-1. Coupling(Data Coupling)

class MemberController

```
 3⊕ import java.sql.Timestamp;
20  public class MemberController {
21⊖     @FXML
22      private Button searchBookButton;
23⊖     @FXML
24      private Button borrowBookButton;
25⊖     public void searchBookButtonClick(ActionEvent event)
26      {
27          Search.main(null);
28      }
29⊖     @FXML
30      public void borrowBookButtonClick(ActionEvent event)
31      {
32          JFrame borrowFrame=new JFrame("借書介面");
33          JButton b1 = new JButton("借書");
34          JTextField ID = new JTextField(15);
35          JPanel p= new JPanel();
36          p.add(new JLabel("輸入書的ID"));
37          p.add(ID);
38          p.add(b1);
39          borrowFrame.add(p);
40          borrowFrame.pack();
41          borrowFrame.setVisible(true);
42          b1.addActionListener(ActionEvent->
43          {
44              Member checkMember = new Member();
45              checkMember=InitialGUI.getloginMember();
46              String table= new String("searchpaperbook");
47              try
48              {
49              ArrayList<PaperBook> storePaperBook=new ArrayList<PaperBook>();
50              PaperBook checkPaperBook=new PaperBook();
```

1

7-2. Cohesion(Function Cohesion)

class SendEmail

```
5  public class SendEmail {
6      public static void send(Member i) {
7          // Get properties object
8          String from = "kencs16358@gmail.com";
9          String password = "********";
10         String to = i.getmemberemail();
11         String sub = "";
12         String msg = "" + i.getmemberName() + "" + i.getnumberOfNoticeBook() + ""
13                 + i.getnumberOfOverdueBook() + ";
14         Properties props = new Properties();
15         props.put("mail.smtp.host", "smtp.gmail.com");
16         props.put("mail.smtp.socketFactory.port", "465");
17         props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
18         props.put("mail.smtp.auth", "true");
19         props.put("mail.smtp.port", "465");
20         // get Session
21         Session session = Session.getDefaultInstance(props, new javax.mail.Authenticator() {
22             protected PasswordAuthentication getPasswordAuthentication() {
23                 return new PasswordAuthentication(from, password);
24             }
25         });
26         // compose message
27         try {
28             MimeMessage message = new MimeMessage(session);
29             message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
30             message.setSubject(sub);
31             message.setText(msg);
32             // send message
33             Transport.send(message);
34             System.out.println("message sent successfully");
35         } catch (MessagingException e) {
36             throw new RuntimeException(e);
37         }
38     }
```

7-3. Connascence(Convention Connascence)

class PaperBook

```
2  import javafx.application.Application;
7  public class LibrarianGUI extends Application{
8      Stage stage=new Stage();
9      public void start(Stage primaryStage) {
10         try {
11         FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("LibrarianGUI.fxml"));
12            Parent root = (Parent) fxmlLoader.load();
13            Scene scene = new Scene(root);
14            primaryStage.setTitle("LibrarianGUI");
15            primaryStage.setScene(scene);
16            primaryStage.show();
17         } catch(Exception e) {
18                e.printStackTrace();
19            }
20         }
21     public static void main(String[] args) {
22
23            launch(args);
24
25         }
26     public void  showWindow()
27     {
28     start(stage);
29     }
30 }
```

(8) Referential integrity

8. Referential integrity

Primary Key

BookBorrowedRecord

Foreign Key

BookID	BookType	userID	userType	BorrowTime
1	PaperBook	M01	Member	107/02/01
2	PaperBook	M02	Member	107/03/01
3	PaperBook	M01	Member	107/04/04

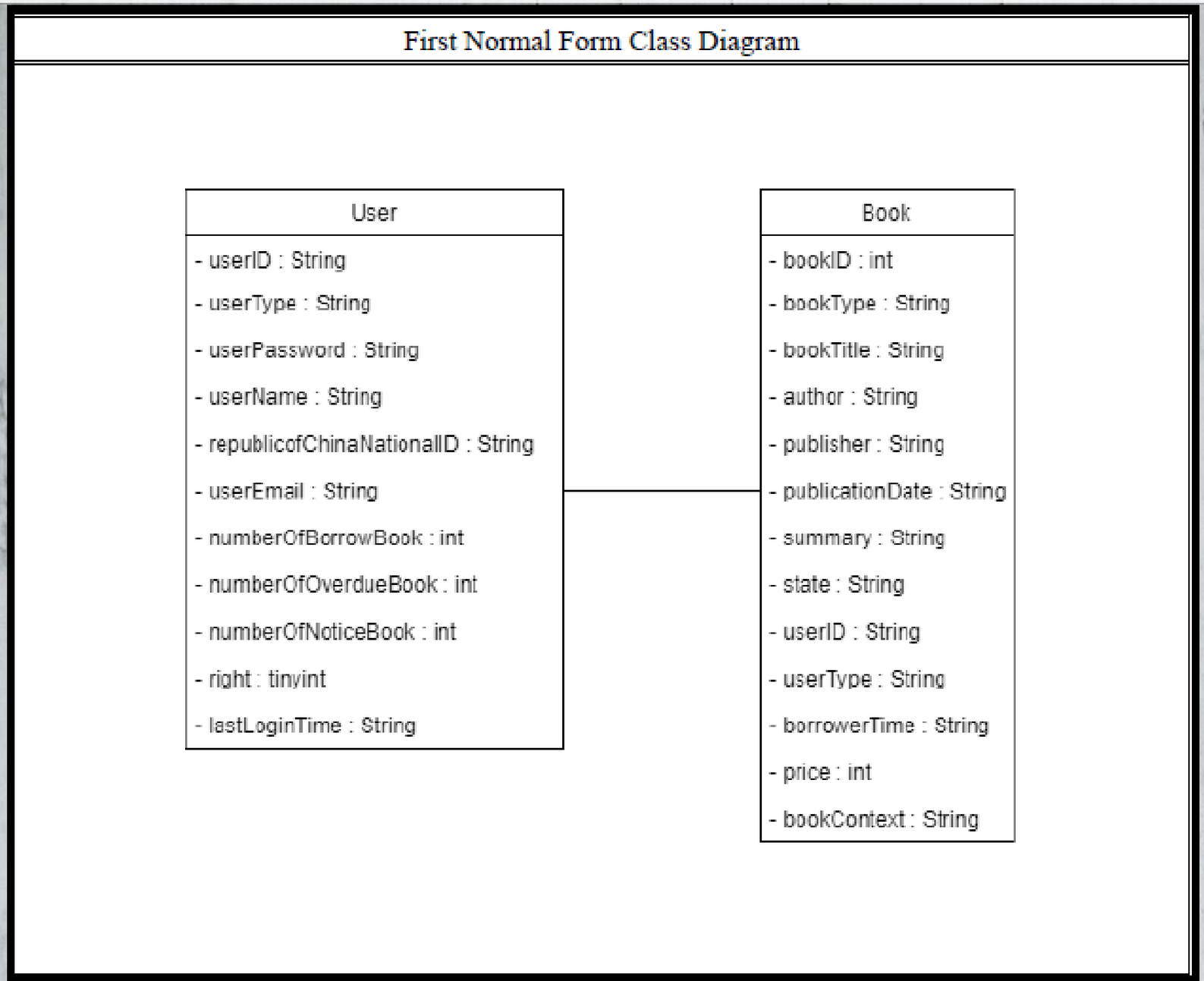
Member

Primary Key

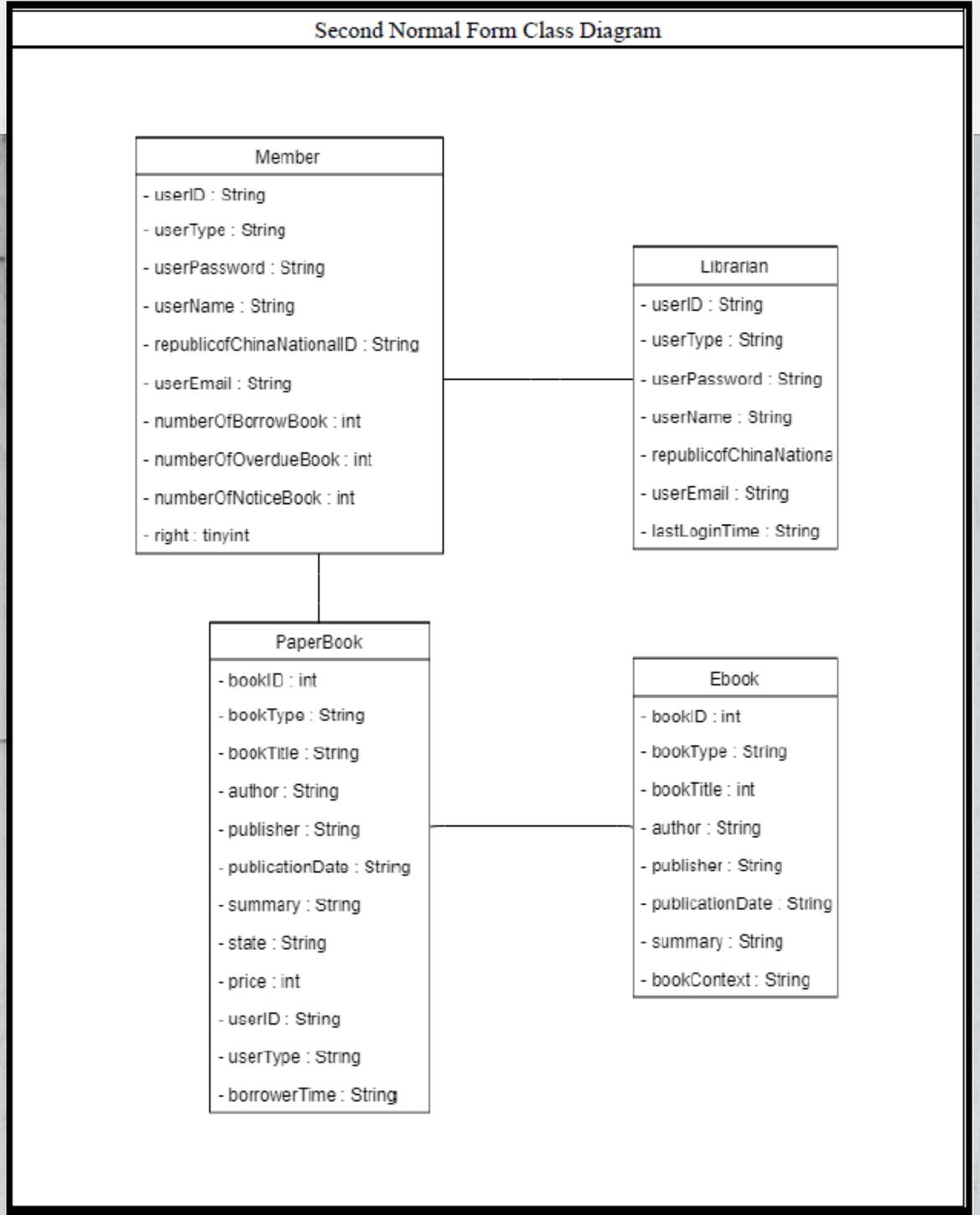
userID	userType	right
M01	Member	N
M02	Member	Y
M03	Member	Y
M05	Member	Y

(9) Normalizationions

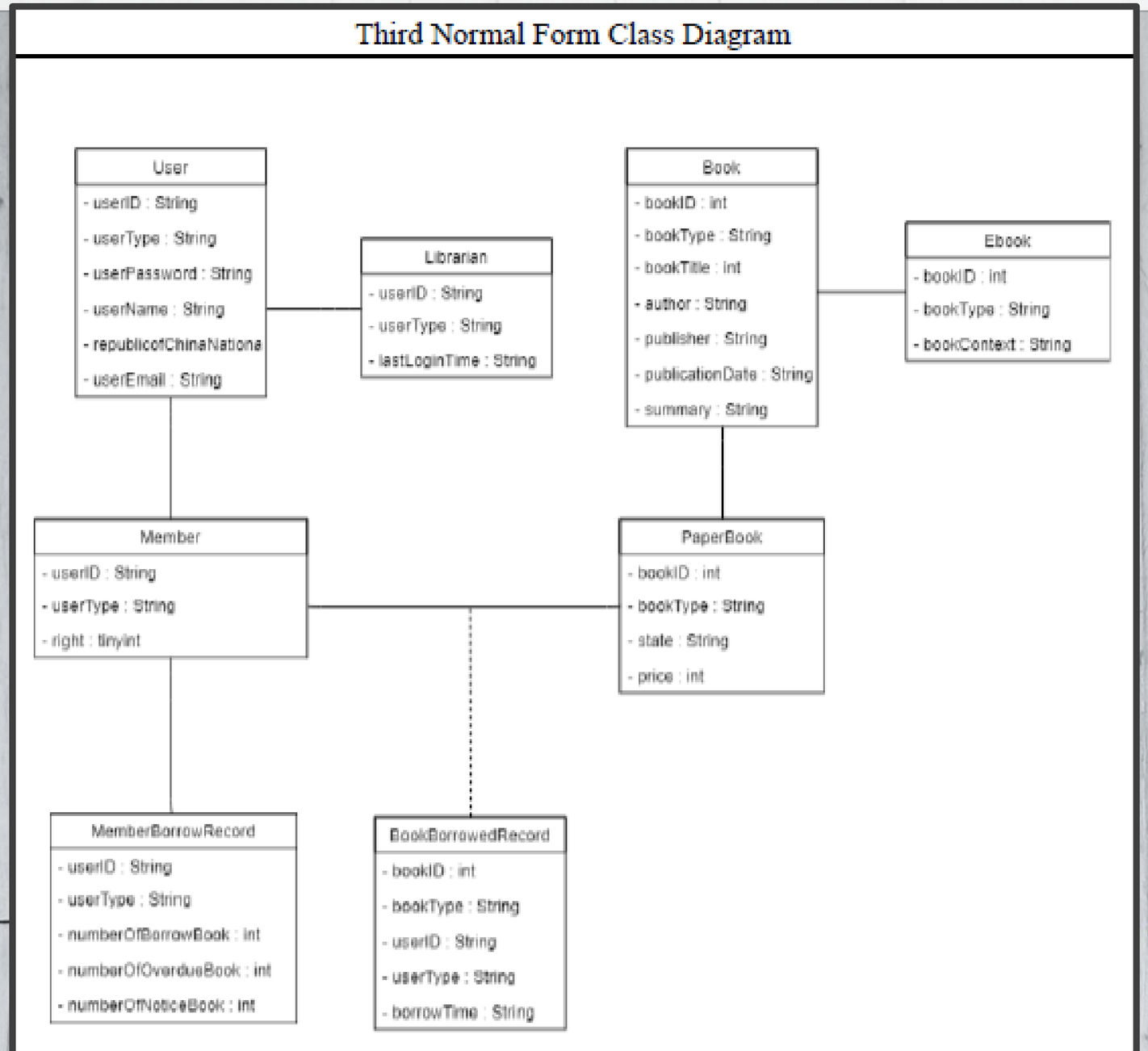
9. Class Diagram-1



9. Class Diagram-2



9. Class Diagram-3



9. Zero Normal Form

Book

bookID	bookType	bookTitle	author	publisher	publicationDate	summary	state	price	borrowerID
1	PaperBook	I'm No.1	oneno	Red	107/01	this book makes you become no.1	Overdue	111	M01
2	PaperBook	Second not bad	twowt	BANANA	107/02	don't always want be no.1, I'll tell you advantage of second	Borrowed	222	M02
1	Ebook	Third what ever	threerht	Zebra	107/03	third means you just behind two people, don't think to much			
3	PaperBook	Forth you better relex	fouruof	OK	107/04	If you want do more better but always get forth, you must get too much pressure to yourself. Try to relax.	Borrowed	333	M01
4	PaperBook	No fifth	fivevif	BANANA	107/01	No one care about fifth, just practice harder.	Availabe	444	
5	PaperBook	Sixth give up	sixis	BANANA	107/03	six is not a lucky number, give up will give you happy life	Missing	555	
2	Ebook	Lucky seven	oneno	Zebra	107/05	you must a luck guy to get this number, let me aupluse to you			
6	PaperBook	Super eight	sixis	OK	107/02	lying eight is unlimit, you are superman	Unavailable	121	
7	PaperBook	number nine	ninenin	Red	107/03	no no no no, just nine just a number, No MORE	Damaged	232	
8	PaperBook	Top ten fact	seveneves	BANANA	107/01	Fact no.1：If you want to know, borrow me first	Repaired	343	
9	PaperBook	Uncountable	twowt	OK	107/01	I can't count anymore, don't ask me the number behind ten.	Deregistered	10	

User

userID	userType	userPassword	userName	republicofChinaNationalID	userEmail	lastLoginTime	numberOfBorrowBook	numberOfOverdueBook	numberOfNoticeBook	right
M01	Member	1234	Lynn	A111333555	Lynn@mail		2	1	0	N
M02	Member	5678	Tim	R333555777	Tim@mail		1	0	1	Y
M01	Member	1234	Lynn	A111333555	Lynn@mail		2	1	0	N
M03	Member	8901	Ken	P555777999	Ken@mail		0	0	0	Y
M05	Member	2345	Smitevevejagermanjason	L777999111	Smit@mail		0	0	0	Y
L01	Librarian	9876	Omaiwa	I222444666	Omaiwa@mail	107/02/02 02:02				
L02	Librarian	5432	Mo sinde	U444666888	Mo sinde@mail	107/01/01 01:01				
L03	Librarian	1098	Iru	T666888000	Iru@mail	106/06/06 06:06				
L04	Librarian	7654	Nani	W888000222	Nani@mail	107/05/05 05:05				

borrowerType	borrowerTime	bookContext
Member	107/02/01	
Member	107/03/01	
		Third is good, don't mind.Third is good, don't mind.Third is good, don't mind…
Member	107/04/04	
		Seven is a good number.Seven is a good number.Seven is a good number…

9. First Normal Form

Book

bookID	bookType	bookTitle	author	publisher	publicationDate	summary
1	PaperBook	I'm No.1	oneno	Red	107/01	this book makes you become no.1
2	PaperBook	Second not bad	twowt	BANANA	107/02	don't always want be no.1, I'll tell you advantage of second
1	Ebook	Third what ever	threerht	Zebra	107/03	third means you just behind two people, don't think to much
3	PaperBook	Forth you better relex	fouruof	OK	107/04	If you want do more better but always get forth, you must get too much pressure to yourself. Try to relax.
4	PaperBook	No fifth	fivevif	BANANA	107/01	No one care about fifth, just practice harder.
5	PaperBook	Sixth give up	sixis	BANANA	107/03	six is not a lucky number, give up will give you happy life
2	Ebook	Lucky seven	oneno	Zebra	107/05	you must a luck guy to get this number, let me aupluse to you
6	PaperBook	Super eight	sixis	OK	107/02	lying eight is unlimit, you are superman
7	PaperBook	number nine	ninenin	Red	107/03	no no no no, just nine just a number, No MORE
8	PaperBook	Top ten fact	seveneves	BANANA	107/01	Fact no.1：If you want to know, borrow me first
9	PaperBook	Uncountable	twowt	OK	107/01	I can't count anymore, don't ask me the number behind ten.

state	price	borrowerID	borrwerType	borrowerTime	bookContext
Overdue	111	M01	Member	107/02/01	
Borrowed	222	M02	Member	107/03/01	
					Third is good, don't mind.Third is good, don't mind.Third is good, don't mind…
Borrowed	333	M01	Member	107/04/04	
Availabe	444				
Missing	555				
					Seven is a good number.Seven is a good number.Seven is a good number…
Unavailable	121				
Damaged	232				
Repaired	343				
Deregistered	10				

User

userID	userType	userPassword	userName	republicofChinaNationalID	userEmail	lastLoginTime	numberOfBorrowBook	numberOfOverdueBook	numberOfNoticeBook	right
M01	Member	1234	Lynn	A111333555	Lynn@mail		2	1	0	N
M02	Member	5678	Tim	R333555777	Tim@mail		1	0	1	Y
M03	Member	8901	Ken	P555777999	Ken@mail		0	0	0	Y
M05	Member	2345	Smitevevejagermanjason	L777999111	Smit@mail		0	0	0	Y
L01	Librarian	9876	Omai wa	I222444666	Omai wa@mail	107/02/02 02:02				
L02	Librarian	5432	Mo sinde	U444666888	Mo sinde@mail	107/01/01 01:01				
L03	Librarian	1098	Iru	T666888000	Iru@mail	106/06/06 06:06				
L04	Librarian	7654	Nani	W888000222	Nani@mail	107/05/05 05:05				

9. Second Normal Form

PaperBook

bookID	bookType	bookTitle	author	publisher	publicationDate	summary	state	price	borrowerID	borrowerType	borrowerTime
1	PaperBook	I'm No.1	oneno	Red	107/01	this book makes you become no.1	Overdue	111	M01	Member	107/02/01
2	PaperBook	Second not bad	twowt	BANANA	107/02	don't always want be no.1. I'll tell you advantage of second	Borrowed	222	M02	Member	107/03/01
3	PaperBook	Forth you better relex	fouruof	OK	107/04	If you want do more better but always get forth, you must get too much pressure to yourself. Try to relax.	Borrowed	333	M01	Member	107/04/04
4	PaperBook	No fifth	fivevif	BANANA	107/01	No one care about fifth, just practice harder.	Availabe	444			
5	PaperBook	Sixth give up	sixis	BANANA	107/03	six is not a lucky number, give up will give you happy life	Missing	555			
6	PaperBook	Super eight	sixis	OK	107/02	lying eight is unlimit, you are superman	Unavailable	121			
7	PaperBook	number nine	ninenin	Red	107/03	no no no no, just nine just a number, No MORE	Damaged	232			
8	PaperBook	Top ten fact	seveneves	BANANA	107/01	Fact no.1：If you want to know, borrow me first	Repaired	343			
9	PaperBook	Uncountable	twowt	OK	107/01	I can't count anymore, don't ask me the number behind ten.	Deregistered	10			

Ebook

bookID	bookType	bookTitle	author	publisher	publicationDate	summary	bookContext
1	Ebook	Third what ever	threerht	Zebra	107/03	third means you just behind two people, don't think to much	Third is good, don't mind.Third is good, don't mind.Third is good, don't mind…
2	Ebook	Lucky seven	oneno	Zebra	107/05	you must a luck guy to get this number, let me aupluse to you	Seven is a good number.Seven is a good number.Seven is a good number…

Member

userID	userType	userPassword	userName	republicofChinaNationalID	userEmail	numberOfBorrowBook	numberOfOverdueBook	numberOfNoticeBook	right
M01	Member	1234	Lynn	A111333555	Lynn@mail	2	1	0	N
M02	Member	5678	Tim	R333555777	Tim@mail	1	0	1	Y
M03	Member	8901	Ken	P555777999	Ken@mail	0	0	0	Y
M05	Member	2345	Smitevevejagermanjason	L777999111	Smit@mail	0	0	0	Y

Librarian

userID	userType	userPassword	userName	republicofChinaNationalID	userEmail	lastLoginTime
L01	Librarian	9876	Omai wa	I222444666	Omai wa@mail	107/02/02 02:02
L02	Librarian	5432	Mo sinde	U444666888	Mo sinde@mail	107/01/01 01:01
L03	Librarian	1098	Iru	T666888000	Iru@mail	106/06/06 06:06
L04	Librarian	7654	Nani	W888000222	Nani@mail	107/05/05 05:05

9. Third Normal Form

Book

bookID	bookType	bookTitle	author	publisher	publicationDate	summary
1	PaperBook	I'm No.1	oneno	Red	107/01	this book makes you become no.1
2	PaperBook	Second not bad	twowt	BANANA	107/02	don't always want be no.1, I'll tell you advantage of second
3	PaperBook	Forth you better relex	fouruof	OK	107/04	If you want do more better but always get forth, you must get too much pressure to yourself. Try to relax.
4	PaperBook	No fifth	fivevif	BANANA	107/01	No one care about fifth, just practice harder.
5	PaperBook	Sixth give up	sixis	BANANA	107/03	six is not a lucky number, give up will give you happy life
6	PaperBook	Super eight	sixis	OK	107/02	lying eight is unlimit, you are superman
7	PaperBook	number nine	ninenin	Red	107/03	no no no no, just nine just a number, No MORE
8	PaperBook	Top ten fact	seveneves	BANANA	107/01	Fact no.1 : If you want to know, borrow me first
9	PaperBook	Uncountable	twowt	OK	107/01	I can't count anymore, don't ask me the number behind ten.
1	Ebook	Third what ever	threerht	Zebra	107/03	third means you just behind two people, don't think to much
2	Ebook	Lucky seven	oneno	Zebra	107/05	you must a luck guy to get this number, let me auphuse to you

Member

userID	userType	right
M01	Member	N
M02	Member	Y
M03	Member	Y
M05	Member	Y

User

userID	userType	userPassword	userName	republicofChinaNationalID	userEmail
M01	Member	1234	Lynn	A111333555	Lynn@mail
M02	Member	5678	Tim	R333555777	Tim@mail
M03	Member	8901	Ken	P555777999	Ken@mail
M05	Member	2345	Smitevevejagermanjason	L777999111	Smit@mail
L01	Librarian	9876	Omai wa	I222444666	Omai wa@mail
L02	Librarian	5432	Mo sinde	U444666888	Mo sinde@mail
L03	Librarian	1098	Iru	T666888000	Iru@mail
L04	Librarian	7654	Nani	W888000222	Nani@mail

PaperBook

bookID	bookType	state	price
1	PaperBook	Overdue	111
2	PaperBook	Borrowed	222
3	PaperBook	Borrowed	333
4	PaperBook	Availabe	444
5	PaperBook	Missing	555
6	PaperBook	Unavailable	121
7	PaperBook	Damaged	232
8	PaperBook	Repaired	343
9	PaperBook	Deregistered	10

Ebook

bookID	bookType	bookContext
1	Ebook	Third is good, don't mind.Third is good, don't mind.Third is good, don't mind…
2	Ebook	Seven is a good number.Seven is a good number.Seven is a good number…

Librarian

userID	userType	lastLoginTime
L01	Librarian	107/02/02 02:02
L02	Librarian	107/01/01 01:01
L03	Librarian	106/06/06 06:06
L04	Librarian	107/05/05 05:05

BookBorrowedRecord

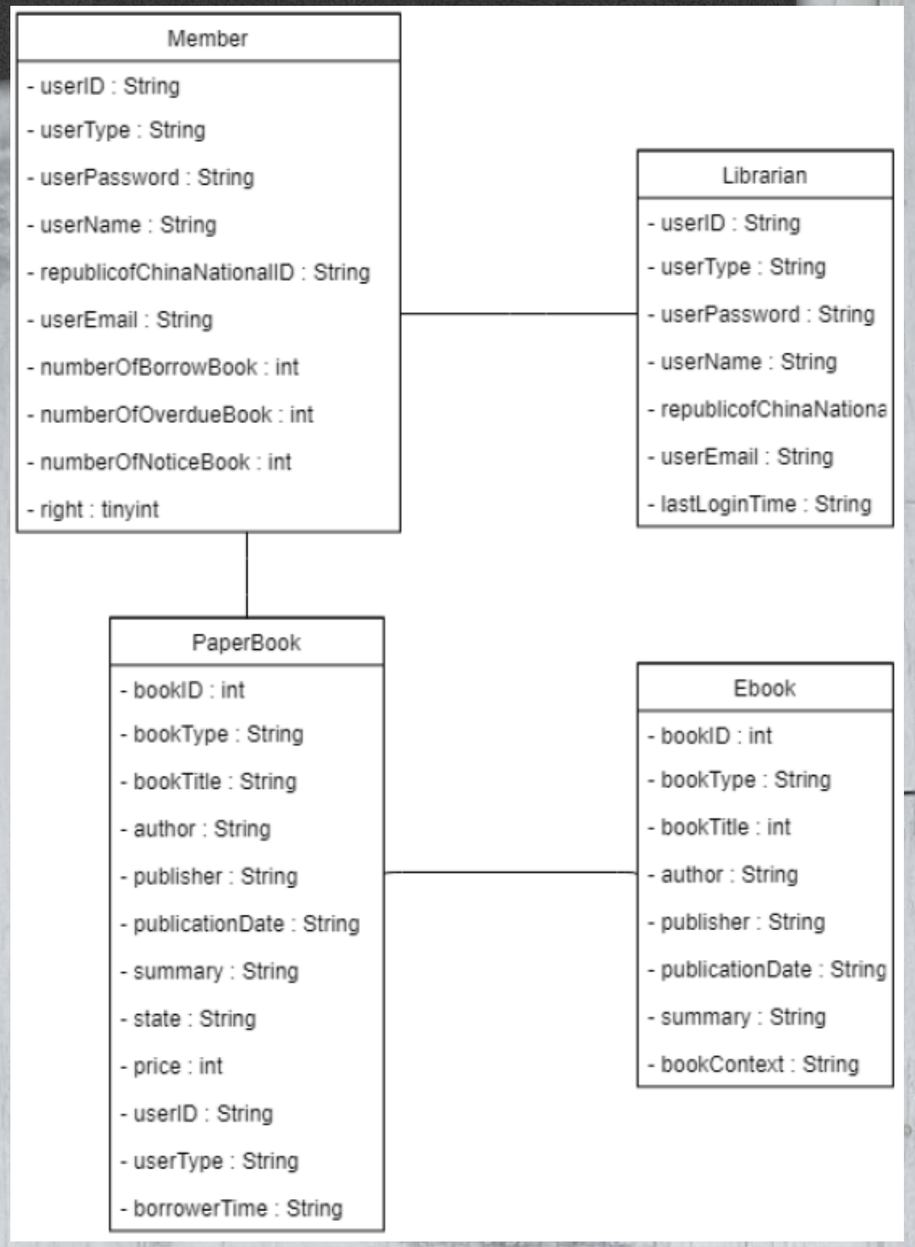
BookID	BookType	userID	userType	BorrowTime
1	PaperBook	M01	Member	107/02/01
2	PaperBook	M02	Member	107/03/01
3	PaperBook	M01	Member	107/04/04

MemberBorrowedRecord

userID	userType	numberOfBorrowBook	numberOfOverdueBook	numberOfNoticeBook
M01	Member	2	1	0
M02	Member	1	0	1
M03	Member	0	0	0
M05	Member	30	0	0

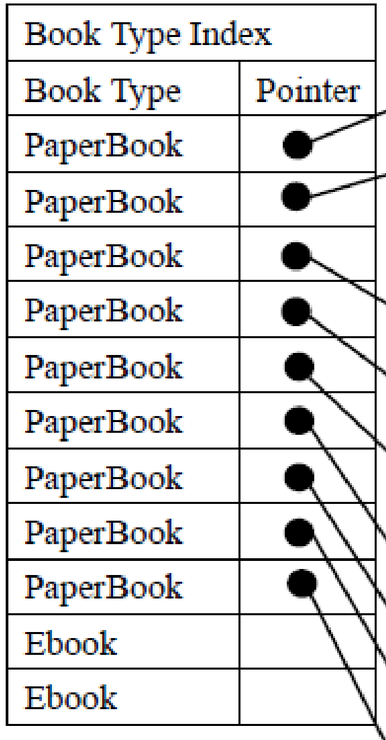
(10) Denormalize and New Class diagram

10. Class Diagram



(11) Inter-file clustering and Index strategies

11. Inter-file clustering and Index strategies-1



Book Type Index	
Book Type	Pointer
PaperBook	
PaperBook	
PaperBook	
PaperBook	
PaperBook	
PaperBook	
PaperBook	
PaperBook	
PaperBook	
Ebook	
Ebook	

Book

BookID	BookType	BookTitle	Author	Publisher	PublicationDate	Summary
1	PaperBook	I'm No.1	oneno	Red	107/01	this book makes you become no.1
2	PaperBook	Second not bad	twowt	BANANA	107/02	don't always want be no.1, I'll tell you advantage of second
1	Ebook	Third what ever	threerht	Zebra	107/03	third means you just behind two people, don't think to much
3	PaperBook	Forth you better relex	fouruof	OK	107/04	If you want do more better but always get forth, you must get too much pressure to yourself. Try to relax.
4	PaperBook	No fifth	fivevif	BANANA	107/01	No one care about fifth, just practice harder.
5	PaperBook	Sixth give up	sixis	BANANA	107/03	six is not a lucky number, give up will give you happy life
2	Ebook	Lucky seven	oneno	Zebra	107/05	you must a luck guy to get this number, let me aupluse to you
6	PaperBook	Super eight	sixis	OK	107/02	lying eight is unlimit, you are superman
7	PaperBook	number nine	ninenin	Red	107/03	no no no no, just nine just a number, No MORE
8	PaperBook	Top ten fact	seveneves	BANANA	107/01	Fact no.1：If you want to know, borrow me first
9	PaperBook	Uncountable	twowt	OK	107/01	I can't count anymore, don't ask me the number behind ten.

11. Inter-file clustering and Index strategies-2

Publisher Type Index	
Publisher Type	Pointer
BANANA	●
BANANA	●
BANANA	●
BANANA	●
Red	●
Red	
Zebra	●
Zebra	
OK	●
OK	
OK	

Book

BookID	BookType	BookTitle	Author	Publisher	PublicationDate	Summary
1	PaperBook	I'm No.1	oneno	Red	107/01	this book makes you become no.1
2	PaperBook	Second not bad	twowt	BANANA	107/02	don't always want be no.1, I'll tell you advantage of second
1	Ebook	Third what ever	threerht	Zebra	107/03	third means you just behind two people, don't think to much
3	PaperBook	Forth you better relex	fouruof	OK	107/04	If you want do more better but always get forth, you must get too much pressure to yourself. Try to relax.
4	PaperBook	No fifth	fivevif	BANANA	107/01	No one care about fifth, just practice harder.
5	PaperBook	Sixth give up	sixis	BANANA	107/03	six is not a lucky number, give up will give you happy life
2	Ebook	Lucky seven	oneno	Zebra	107/05	you must a luck guy to get this number, let me aupluse to you
6	PaperBook	Super eight	sixis	OK	107/02	lying eight is unlimit, you are superman
7	PaperBook	number nine	ninenin	Red	107/03	no no no no, just nine just a number, No MORE
8	PaperBook	Top ten fact	seveneves	BANANA	107/01	Fact no.1：If you want to know, borrow me first
9	PaperBook	Uncountable	twowt	OK	107/01	I can't count anymore, don't ask me the number behind ten.

Thank you!