

通販サイトにログインし 請求書をメールで送信

もくじ

論理構造 2 ページ、

ブラウザ画面の例を 3 ～ 9 ページ、

メールと PDF の例を 10 ～ 12 ページ、

Mukunashi_syoten_ver1 のソースコードを 13 ～ 73 ページ、

データベースの中身を 74 ～ 75 ページに書く。

11月11日

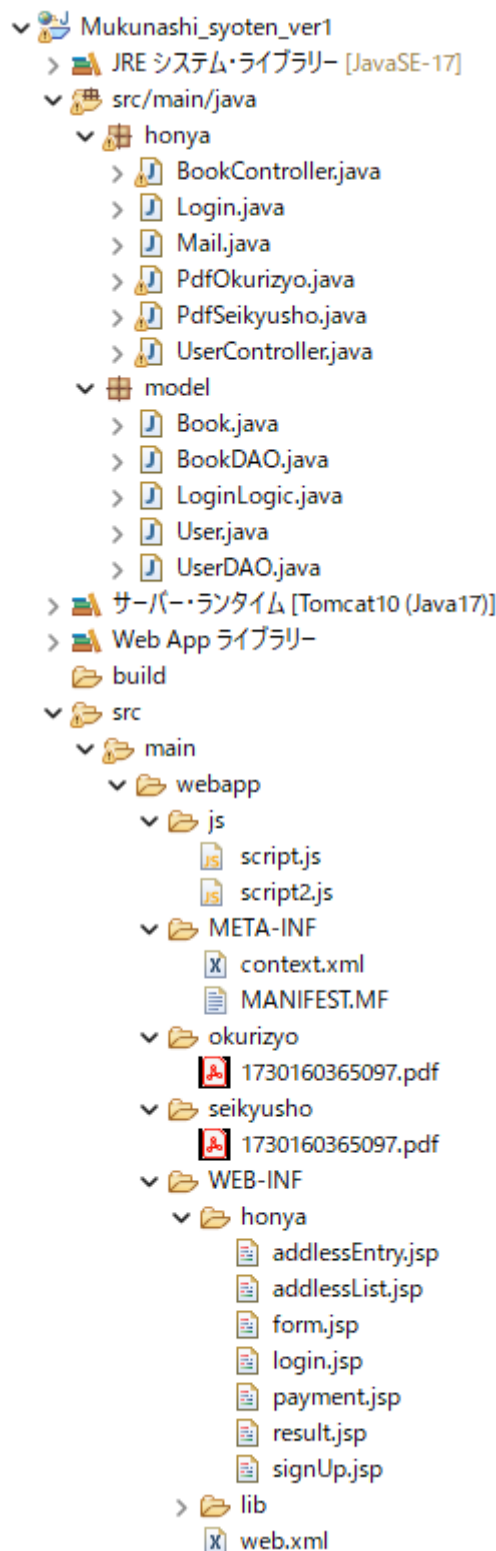
名前

みつやす
光安

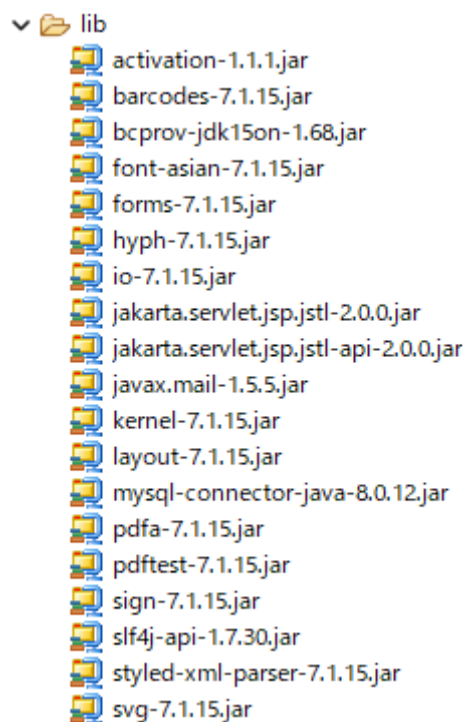
まさのり
聖徳

16

論理構造



lib の中身



ブラウザ画面の例

ブラウザで下記の URL を検索する。

http://localhost:8080/Mukunashi_syoten_ver1/BookController

form.jsp

本を刊行年月日が新しい順に表示する。購入したい本の購入冊数を入力して

「カートに入れる」を押すと、カートにあるものを確認する画面に移る。

棕梨書店タイムセール

書名	価格	出版社	刊行年月日	購入冊数	在庫数量
iphoneアプリ超入門	2200	日経BP	2020-02-28	<input type="text" value="2"/>	100
独習ASP.NET Webフォーム	3800	翔泳社	2020-02-17	<input type="text" value="0"/>	100
基礎からしっかり学ぶc#の教科書	2900	日経BP	2019-12-20	<input type="text" value="0"/>	100
はじめてのASP.NET Webフォームアプリ開発	3200	秀和システム	2019-10-12	<input type="text" value="1"/>	100
SQLデータ分析・活用入門	2600	ソシム	2019-09-12	<input type="text" value="0"/>	100
これから始めるVue.js実践入門	3990	SBクリエイティブ	2019-08-22	<input type="text" value="0"/>	100
はじめてのAndroidアプリ開発	3200	秀和システム	2019-08-10	<input type="text" value="1"/>	100
Androidアプリ開発の教科書	2850	翔泳社	2019-07-10	<input type="text" value="0"/>	100
独習Java新版	3150	翔泳社	2019-05-15	<input type="text" value="0"/>	100
JavaScript逆引きレシピ	2730	翔泳社	2018-10-15	<input type="text" value="2"/>	100
<input type="button" value="カートに入れる"/>					

購入冊数には数しか入力できない。負の数や小数や在庫数量を超える数を入力

した場合は、その瞬間に在庫数量の右に「エラー」と表示される。

result.jsp

form.jsp で「カートに入れる」を押すと移る。購入をやめたい本にチェックを入れる。その本をカートから取り除くには「チェックした本を購入取り消し」を押すとその本を取り除いたカートを表示し直す。送付先選択の画面に移るには「購入する」を押す。form.jsp に戻るには「戻る」をクリックする。

カートにあるもの

書名	価格	出版社	刊行年月日	購入冊数	購入取り消し
iphoneアプリ超入門	2200	日経BP	2020-02-28	2	<input type="checkbox"/>
はじめてのASP.NET Webフォームアプリ開発	3200	秀和システム	2019-10-12	1	<input checked="" type="checkbox"/>
はじめてのAndroidアプリ開発	3200	秀和システム	2019-08-10	1	<input type="checkbox"/>
JavaScript逆引きレシピ	2730	翔泳社	2018-10-15	2	<input checked="" type="checkbox"/>

チェックした本を購入取り消し

合計金額：16260円 [戻る](#)

購入する

カートに入れた購入冊数は「戻る」から form.jsp に戻ると購入冊数に入力されたままになっている。カートに本が一つも入っていない時は「購入する」ボタンは表示されない。

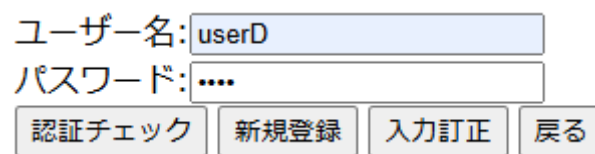
login.jsp

result.jsp で「購入」を押した時にまだログインしていない場合にここへ移る。

ユーザー名とパスワードを入力して「認証チェック」を押し、それが正しけれ

ば送付先選択の画面に移る。「新規登録」を押すと新規ユーザー登録画面に移る。「入力訂正」を押すと入力欄が空白になる。「戻る」を押すと result.jsp に戻る。

ログイン画面



ユーザー名:

パスワード:

入力したユーザー名とパスワードが間違っていた時に「認証チェック」を押すと、ボタンの下に「ログインに失敗しました」と表示され入力欄が空白になる。

addlessEntry.jsp

login.jsp で「新規登録」を押すところへ移る。ユーザー情報を全て入力してから「登録」を押すと、新規登録したアカウントで送付先選択の画面に移る。

「戻る」をクリックすると login.jsp に戻る。「キャンセル」をクリックすると入力欄が空白になる。

新規ユーザー登録

ユーザー名:
パスワード:
氏名:
〒: 住所:
メールアドレス:

入力欄に空白がある状態で「登録」をクリックすると、ボタンの下に「入力欄に空白があります」と表示されて入力欄が空白になる。

addlessList.jsp

result.jsp ですでにログインした状態で「購入する」を押すか login.jsp で「認証チェック」を押してログインに成功するか addlessEntry.jsp でユーザー情報を入力して「登録」を押すところへ移る。送付先を一つ選び「送付先決定」を押すと注文完了画面に移る。「新規追加」を押すと新規送付先登録画面に移る。「削除実行」を押すとチェックした送付先を削除した登録済み送付先を表示し直す。「戻る」を押すと result.jsp に戻る。

登録済み送付先

userD様ログイン中 [ログアウト](#)

送付先番号	名前と住所とメールアドレス	削除
	山田太郎	
<input checked="" type="radio"/> 1	103-9999,東京都東京市南町1-1-1 yamada@example.com	<input type="checkbox"/>
	鈴木浩二	
<input type="radio"/> 2	504-1111,神奈川県松戸市金町2-4-5 suzuki@example.com	<input type="checkbox"/>
	井上春子	
<input type="radio"/> 3	904-2222,埼玉県浦安市本町4-7-9 inoue@example.com	<input type="checkbox"/>
<div><input type="button" value="送付先決定"/> <input type="button" value="新規追加"/> <input type="button" value="削除実行"/> <input type="button" value="戻る"/></div>		

画面のタイトルの下にログイン中のユーザー名とログアウトして form.jsp に戻るリンクが表示される。これはログイン中なら form.jsp や result.jsp にも表示される。送付先は最大五件まで登録できる。送付先は一件も登録されていない時は「送付先決定」ボタンと「削除実行」ボタンは表示されない。送付先が五件登録されている時は「新規追加」ボタンは表示されない。送付先番号は最初は1が選ばれている。しかし、別の送付先を選んでから別の画面へ移った後にここへ戻ってくると、以前に選んだ送付先が選ばれたままになっている。もし選んだ送付先を削除したら、送付先は1が選ばれ直す。

addlessEntry.jsp

addlessList.jsp で「新規追加」を押すところへ移る。送付先情報を全て入力してから「登録」を押すと、登録した送付先が表示されている addlessList.jsp に移る。「戻る」をクリックすると addlessList.jsp に戻る。「キャンセル」をクリックすると入力欄が空白になる。

新規送付先登録

氏名:

〒: 住所:

メールアドレス:

入力欄に空白がある状態で「登録」をクリックすると、ボタンの下に「入力欄に空白があります」と表示されて入力欄が空白になる。

payment.jsp

addlessList.jsp で「送付先決定」を押すところへ移る。「本を選ぶ画面に戻る」をクリックするとカートの中身を削除して form.jsp に移る。「購入をキャンセルする」をクリックすると result.jsp に戻って購入をキャンセルするが、このボタンが表示されるのは payment.jsp に移ってから 5 秒間だけである。「5 秒以内なら注文をキャンセルできます」の文の中の数字は 1 秒ごとにカウントダウン

ンされて、0 になったらこの文は表示されなくなる。画面の下には合計金額と

送付先と注文を表示する。

注文完了しました。

userD様ログイン中 [ログアウト](#)

[本を選ぶ画面に戻る](#)

5秒以内なら注文をキャンセルできます

[購入をキャンセルする](#)

合計金額 : 16260円

送付先	名前	住所	メールアドレス		
	山田太郎	103-9999,東京都東京市南町1-1-1	yamada@example.com		
注文	書名		価格	出版社	刊行年月日 購入冊数
	iphoneアプリ超入門		2200	日経BP	2020-02-28 2
	はじめてのASP.NET Webフォームアプリ開発		3200	秀和システム	2019-10-12 1
	はじめてのAndroidアプリ開発		3200	秀和システム	2019-08-10 1
	JavaScript逆引きレシピ		2730	翔泳社	2018-10-15 2

「購入をキャンセルする」ボタンが表示されなくなった時点で注文が完了する。

その直後に請求書の PDF を添付したメールを、送り状の PDF を添付したメー

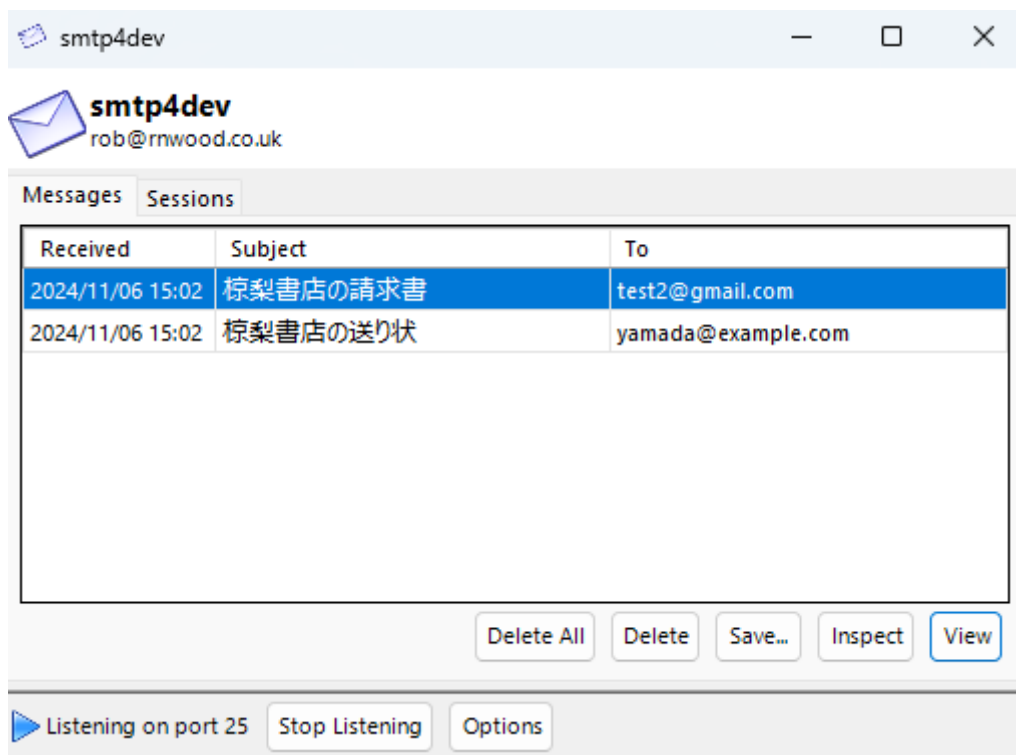
ルを、それぞれユーザーと送付先に送信します。そしてそれぞれの本の購入冊数

分在庫数量を減らす。

メールと PDF の例

メールの受信画面

ここでは smtp4.dev というダミーのメールサーバーのアプリで送信したメールを確認する。



メールに添付する PDF の名前は adlessList.jsp で「送付先決定」を押した時に UNIX エポックから経過したミリ秒にする。

請求書の本文

聖徳太子様へ棕梨書店から請求書です。

請求書の PDF

聖徳太子様



棕梨書店

書名	価格	購入冊数
iphoneアプリ超入門	2200	2
はじめてのASP.NET Webフォームアプリ開発	3200	1
はじめてのAndroidアプリ開発	3200	1
JavaScript逆引きレシピ	2730	2

合計金額：16260円



棕梨書店にアクセスするための下記 URL で QR コードを作成した。

http://localhost:8080/Mukunashi_syoten_ver1/BookController

バーコードは合計金額のすぐ下に表示される。

送り状の本文

山田太郎様へ棕梨書店から送り状です。

送り状の PDF

103-9999,東京都東京市南町1-1-1

2024/11/7

山田太郎様

下記は請求書のバーコードです

福岡市東区千早4丁目24-1



納品のご案内

拝啓平素は格別のご哀願を賜りまして誠にありがとうございます。

聖徳太子様より依頼された商品を送付いたしますので、ご査収のほどよろしくお願いいたします。

書名	購入冊数
iphoneアプリ超入門	2
はじめてのASP.NET Webフォームアプリ開発	1
はじめてのAndroidアプリ開発	1
JavaScript逆引きレシピ	2

以上

送り主 聖徳太子

ちなみに、バーコードの番号は「0123456789」で仮に固定している。

Mukunashi_syoten_ver1 のソースコード

form.jsp のソースコード

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      import="java.sql.*,javax.naming.*,javax.sql.*,java.text.*"%>
3  <%-- Book クラスとリストを扱うためのクラスをインポート --%>
4  <%@ page import="model.Book"%>
5  <%@ page import="java.util.List" %>
6  <%@ page import="model.User" %>
7  <!DOCTYPE html>
8  <html>
9  <head>
10 <meta charset="UTF-8">
11 <script src="js/script.js" charset="utf-8" defer></script>
12 <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
13 <title>棕梨書店タイムセール</title>
14 </head>
15 <body>
16 <h2>棕梨書店タイムセール</h2>
17 <%
18     User loginUser = (User)session.getAttribute("loginUser");
19     if(loginUser != null){
20         %>
21         <h4><%= loginUser.getUserId() + "様ログイン中" %><a style="margin-left: 60px"
22             href="UserController">ログアウト</a></h4>
23         <%} %>
24
25     <%-- ここから BookController にリンクする場合だけリクエストパラメータを追加 --%>
26     <form method="POST" action="BookController?action=done">
27         <table class="table">
28             <thead>
29                 <tr>
30                     <th>書名</th><th>価格</th><th>出版社</th><th>刊行年月日</th><th>購
31                     入冊数</th><th>在庫数量</th>
32                 </tr>
```

```

33 </thead>
34 <tbody>
35
36 <%
37 //以前に 1 以上と入力した購入冊数を表示するため
38 List<Book> list = (List<Book>)session.getAttribute("list");
39 //isbn と購入冊数の name に本ごとに 1 ずつ異なる cnt を付け加えて区別する
40 //listnum は list の要素番号
41 int cnt = 0, listnum=0;
42 Context context = new InitialContext();
43 DataSource ds = (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
44 try(Connection db = ds.getConnection()){
45     PreparedStatement ps = db.prepareStatement("SELECT * FROM book ORDER
46 BY published DESC");
47     ResultSet rs = ps.executeQuery();
48     while(rs.next()){
49         cnt++;
50     %>
51
52     <tr>
53         <td>
54             <input type="hidden" name="isbn<%=cnt %>"
55 value="<%=rs.getString("isbn") %>">
56             <%=rs.getString("title") %>
57         </td>
58         <td>
59             <%=rs.getInt("price") %>
60         </td>
61         <td>
62             <%=rs.getString("publish") %>
63         </td>
64         <td>
65             <%=rs.getString("published") %>
66         </td>
67         <td> <!--購入冊数は正の整数か 0 しか入力できない --%>

```

```

68         <!--データベースから表示している本の isbn と list の要素で
69 ある 1 冊以上購入する本の isbn が一致する時に以前の入力内容を表示 --%>
70         <input type="number" min="0"
71 max="<%=rs.getInt("zaiko") %>" name="num<%=cnt %>" size="5" value="<%= (list ==
72 null || list.size() == 0)? 0 : rs.getString("isbn").equals(list.get(listnum).getIsbn())?
73 list.get(listnum).getNum() : 0%>">
74     </td>
75     <td>
76         <%=rs.getInt("zaiko") %>
77     </td>
78     <td style="color: red;">
79         <!--赤色で購入冊数が在庫数量より多いか負の数か小数の場合
80 に script.js からエラーと表示する --%>
81     </td>
82 </tr>
83 <%
84     //list から購入冊数が 1 冊以上の isbn をデータベースから表示している
85     時に list の要素番号を増やす
86     if(list != null && list.size() > 0 &&
87 rs.getString("isbn").equals(list.get(listnum).getIsbn())){
88         listnum++;
89         //list の要素番号が list の範囲外にならないようにする
90         if(listnum == list.size()) listnum--;
91     }
92 }
93 }
94 %>
95 </tbody>
96 </table>
97 <!--cnt は画面に表示した本の数 --%>
98 <input type="hidden" name="cnt" value="<%=cnt %>">
99 <!--isbn と購入冊数と画面に表示した本の数を送信する --%>
100 <input type="submit" value="カートに入れる">
101 </form>
102 </body>
103 </html>

```

script.js のソースコード

```
1  //form.jsp に組み込む
2  'use strict';
3  //cnt は画面に表示した本の数
4  const cnt = $('input[name="cnt"]').val();
5  // 5 0 0 ミリ秒ごとに購入冊数に入力した数を確認する
6  setInterval(function(){
7      for(let i = 1; i <= +cnt; i++){
8          //表示した本ごとに在庫数が含まれる HTML の要素を配列（今回は要素
9  数 1 ）で取得
10         const td = $('tr:nth-of-type(' + i + ')>td:nth-of-type(6)');
11         //購入冊数に入力した数字が在庫数よりも多いか負の数か小数の場合
12         if(+$('`[name="num${i}"]`').val() > +td[0].innerText ||
13 +$('`[name="num${i}"]`').val() < 0 || +$('`[name="num${i}"]`').val() % 1 != 0){
14             //在庫数量の横にエラーと表示する
15             $('tr:nth-of-type(' + i + ')>td:nth-of-type(7)').text("*エラー");
16         }else{
17             //在庫数量の横に何も表示しない
18             $('tr:nth-of-type(' + i + ')>td:nth-of-type(7)').text("");
19         }
20     }
21 }, 500);
```

Book.java のソースコード

```
1  package model;
2
3  import java.io.Serializable;
4
5  public class Book implements Serializable {
6
7      private String isbn;
8      private String title;
9      private int price;
```



```
10     private String publish;
11     private String published;
12     private int num;
13
14     public Book() { /*コンストラクター*/ }
15
16     public String getIsbn() {
17         return isbn;
18     }
19
20     public void setIsbn(String isbn) {
21         this.isbn = isbn;
22     }
23
24     public String getTitle() {
25         return title;
26     }
27
28     public void setTitle(String title) {
29         this.title = title;
30     }
31
32     public int getPrice() {
33         return price;
34     }
35
36     public void setPrice(int price) {
37         this.price = price;
38     }
39
40     public String getPublish() {
41         return publish;
42     }
43
44     public void setPublish(String publish) {
45         this.publish = publish;
```

```

46         }
47
48     public String getPublished() {
49         return published;
50     }
51
52     public void setPublished(String published) {
53         this.published = published;
54     }
55
56     //購入冊数のゲッターとセッター
57     public int getNum() {
58         return num;
59     }
60
61     public void setNum(int num) {
62         this.num = num;
63     }
64
65 }

```

BookDAO.java のソースコード

```

1  package model;
2
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7
8  import javax.naming.Context;
9  import javax.naming.InitialContext;
10 import javax.naming.NamingException;
11 import javax.sql.DataSource;
12

```

```

13 public class BookDAO{
14
15     //本の種類別に購入したい本の isbn と購入冊数を引数にして、購入したい本のイ
16     ンスタンスを返す
17     public Book selectBook(String isbn, String num){
18         Book bok = new Book();
19         try {
20             Context context = new InitialContext();
21             DataSource ds =
22 (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
23             try(Connection db = ds.getConnection()){
24                 //isbn が一致する本の情報をデータベースから取得
25                 PreparedStatement ps = db.prepareStatement(
26                     "SELECT * FROM book WHERE
27 isbn = ?");
28                 ps.setString(1, isbn);
29                 ResultSet rs = ps.executeQuery();
30                 //レコードは 1 件なのでレコードポインタを一回だけ
31                 動かす
32                 rs.next();
33                 bok.setIsbn(rs.getString("isbn"));
34                 bok.setTitle(rs.getString("title"));
35                 bok.setPrice(rs.getInt("price"));
36                 bok.setPublish(rs.getString("publish"));
37                 bok.setPublished(rs.getString("published"));
38                 //request.getParameter()の戻り値である num は
39                 String 型なので int 型に変換する
40                 bok.setNum(Integer.parseInt(num));
41             }
42             } catch (NamingException | SQLException e) {
43                 e.printStackTrace();
44             }
45             return bok;
46         }
47
48     //購入を決定した本の種類のインスタンスを引数にする

```

```

49         public void reduceBook(Book bok) {
50             try {
51                 Context context = new InitialContext();
52                 DataSource ds =
53 (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
54                 try(Connection db = ds.getConnection()){
55                     //isbn が一致する本の在庫をデータベースから取得
56                     PreparedStatement ps = db.prepareStatement(
57                         "SELECT zaiko FROM book
58 WHERE isbn = ?");
59                     ps.setString(1, bok.getIsbn());
60                     ResultSet rs = ps.executeQuery();
61                     //レコードは 1 件なのでレコードポインタを一回だけ
62 動かす
63                     rs.next();
64                     //購入する本の在庫数から購入冊数を引いた値を新た
65 な在庫として取得
66                     int zaiko = rs.getInt("zaiko")- bok.getNum();
67                     //isbn が一致する本の在庫を新たな在庫に更新する
68                     ps = db.prepareStatement(
69                         "UPDATE book SET zaiko = ?
70 WHERE isbn = ?");
71                     ps.setInt(1, zaiko);
72                     ps.setString(2, bok.getIsbn());
73                     ps.executeUpdate();
74                 }
75             } catch (NamingException | SQLException e) {
76                 e.printStackTrace();
77             }
78         }
79     }

```

BookController.java のソースコード

```

1 package honya;

```

```

2
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.util.TimerTask;
7
8 import jakarta.servlet.ServletException;
9 import jakarta.servlet.annotation.WebServlet;
10 import jakarta.servlet.http.HttpServlet;
11 import jakarta.servlet.http.HttpServletRequest;
12 import jakarta.servlet.http.HttpServletResponse;
13 import jakarta.servlet.http.HttpSession;
14 import model.Book;
15 import model.BookDAO;
16 import model.User;
17
18 @WebServlet("/BookController")
19 public class BookController extends HttpServlet {
20
21     //http://localhost:8080/Mukunashi_syoten_ver1/BookController
22     //クライアントが URL で検索した場合や result.jsp か payment.jsp で「戻る」を
23     押した場合か login.jsp でキャンセルした場合
24     protected void doGet(HttpServletRequest request, HttpServletResponse response)
25     throws ServletException, IOException {
26         HttpSession session = request.getSession();
27
28         //payment.jsp で「戻る」を押した場合か login.jsp でキャンセルした場
29     合はログイン情報以外破棄
30         if(session.getAttribute("sum") != null) {
31             //購入した本のリストと合計金額とメール送信の TimerTask を
32     持つセッションを破棄する
33             session.removeAttribute("list");
34             session.removeAttribute("sum");
35             session.removeAttribute("task");
36         }
37         this.getServletContext().getRequestDispatcher("/WEB-

```

```

38 INF/honya/form.jsp").forward(request, response);
39     }
40
41     //form.jsp で「カートに入れる」を押した場合や result.jsp で「チェックした本を
42 購入取り消し」や「購入する」を押した場合
43     protected void doPost(HttpServletRequest request, HttpServletResponse
44 response) throws ServletException, IOException {
45         request.setCharacterEncoding("UTF-8");
46         HttpSession session = request.getSession();
47         //result.jsp でチェックした本の isbn
48         String[] remove = request.getParameterValues("remove");
49         //form.jsp からカートに入れた場合は中身が done そうでなければ null
50         String action = request.getParameter("action");
51
52         // String action が null ならその後の else if でエラーになってしまう
53         if(action == null) {
54             action = "no";
55         }
56
57         //result.jsp で一つでもチェックしていた場合
58         if(remove != null) {
59             //セッションスコープのリストを Book 型のリストにキャスト
60             List<Book> list = (List<Book>)session.getAttribute("list");
61             for(String r: remove) {
62                 for(Book bok: list) {
63                     //チェックした本をリストから見つける
64                     if(bok.getIsbn().equals(r)) {
65                         //チェックした本をリストから削除
66                         list.remove(list.indexOf(bok));
67                         break;
68                     }
69                 }
70             }
71             //セッションスコープのリストを作り直す
72             session.removeAttribute("list");
73             session.setAttribute("list", list);

```

```

74
75 //form.jsp からカートに入れた場合
76 }else if(action.equals("done")){
77     //cnt は form.jsp で表示した本の数
78     int cnt = Integer.parseInt(request.getParameter("cnt"));
79     //Book 型のリストを宣言
80     List<Book> list = new ArrayList<>();
81     BookDAO dao = new BookDAO();
82     //dao.selectBook(String isbn, String num)で本ごとに Bean を作
83     成して返す
84     for(int i = 1; i <= cnt; i++){
85         //購入冊数が 0 でも空白でもない本の場合
86         if((request.getParameter("num" + i)).indexOf("0") !=
87         0 && !request.getParameter("num" + i).equals("")) {
88             //Book 型のリストに購入する本の情報と冊数
89             を加える
90
91             list.add(dao.selectBook(request.getParameter("isbn" + i),
92             request.getParameter("num" + i)));
93         }
94     }
95     //買い物カゴはセッションスコープで作成
96     session.setAttribute("list", list);
97 }
98
99 //result.jsp で「購入する」 ボタンを押した場合
100 if(action.equals("pdf")) {
101     //セッションスコープからユーザー情報を取得
102     User loginUser = (User)session.getAttribute("loginUser");
103
104     //result.jsp で購入するボタンを押した直後
105     if(session.getAttribute("sum") == null) {
106         //合計金額をセッションスコープに入れる
107         session.setAttribute("sum",
108         request.getParameter("sum"));
109     }

```

```

110
111             //ログインしていない場合は addressList.jsp ではなく login.jsp
112     へフォワード
113             if(loginUser == null) {
114
115                 this.getServletContext().getRequestDispatcher("/WEB-
116     INF/honya/login.jsp").forward(request,response);
117             }else {
118
119                 this.getServletContext().getRequestDispatcher("/WEB-
120     INF/honya/addressList.jsp").forward(request,response);
121             }
122             //result.jsp で何もチェックせずに購入取り消し、または payment.jsp で
123     購入キャンセルを押した場合はすぐここへ。
124             //または login.jsp や addressList.jsp で戻るボタンをクリックした場合は
125     すぐここへ
126             }else{
127
128                 //addressList.jsp で戻るボタンをクリックした場合は選んだ送
129     付先の番号をセッションスコープに保存
130                 if(request.getParameter("send") != null) {
131                     session.setAttribute("send",
132     request.getParameter("send"));
133                 }
134
135                 //payment.jsp で「購入をキャンセル」を押した場合
136                 else if(session.getAttribute("task") != null) {
137                     //メールを送信する TimerTask を終了する
138                     TimerTask task =
139     (TimerTask)session.getAttribute("task");
140                     task.cancel();
141                 }
142                 this.getServletContext().getRequestDispatcher("/WEB-
143     INF/honya/result.jsp").forward(request,response);
144             }
145         }

```


result.jsp のソースコード

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      import="java.sql.*,javax.naming.*,javax.sql.*,java.text.*"%>
3  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
5  <%-- Book クラスとリストを扱うためのクラスをインポート --%>
6  <%@ page import="model.Book"%>
7  <%@ page import="java.util.List" %>
8  <%@ page import="model.User" %>
9  <!DOCTYPE html>
10 <html>
11 <head>
12 <meta charset="UTF-8">
13 <title>カートにあるもの</title>
14 </head>
15 <body>
16 <%
17     User loginUser = (User)session.getAttribute("loginUser");
18     if(loginUser != null){
19         %>
20         <h4><%= loginUser.getUserId() + "様ログイン中" %><a style="margin-left: 60px"
21             href="UserController">ログアウト</a></h4>
22         <%} %>
23         <h3>カートにあるもの</h3>
24
25         <form method="POST" action="BookController">
26             <table class="table">
27                 <thead>
28                     <tr>
29                         <th>書名</th><th>価格</th><th>出版社</th><th>刊行年月日</th><th>購
30                         入冊数</th><th>購入取り消し</th>
31                     </tr>

```

```

32 </thead>
33 <tbody>
34 <c:forEach var="item" items="${sessionScope['list']}">
35     <tr>
36         <td>${fn:escapeXml(item.title)}</td>
37         <td>${fn:escapeXml(item.price)}</td>
38         <td>${fn:escapeXml(item.publish)}</td>
39         <td>${fn:escapeXml(item.published)}</td>
40         <td>${fn:escapeXml(item.num)}</td>
41         <%--購入取り消しにチェックした本の isbn を送信する --%>
42         <td><input type="checkbox" name="remove"
43 value="${fn:escapeXml(item.isbn)}"></td>
44     </tr>
45 </c:forEach>
46 </tbody>
47 </table>
48
49 <input type="submit" value="チェックした本を購入取り消し">
50 <%--合計金額を計算するために Java の処理で、"list"オブジェクトを List(Book)でキャス
51 ト --%>
52 <% List<Book> list = (List<Book>)session.getAttribute("list");
53     int sum = 0;
54     //本ごとに価格×購入冊数を合計金額に加える
55     for(Book bok: list) sum += bok.getPrice() * bok.getNum();
56 %>
57 <%--form.jsp に戻ると購入する本の個数が入力されている、合計金額を青色で表示する -
58 -%>
59 <h3>合計金額 : <span style="color: blue;"><%=sum %></span>円 <a
60 href="BookController">戻る</a></h3>
61 </form>
62
63 <%
64 //本を一冊でも選んでいた場合に「購入する」 ボタンを表示する
65 if(sum > 0){
66 %>
67     <form method="POST" action="BookController?action=pdf">

```

```

68         <input type="hidden" name="sum" value="<%=sum %>">
69         <!--合計金額を送信する --%>
70         <input type="submit" value="購入する">
71     </form>
72 <%
73 }
74 %>
75 </body>
76 </html>

```

login.jsp のソースコード

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>ログイン画面</title>
8  </head>
9  <body>
10 <h1>ログイン画面</h1>
11 <form action="?" method="post">
12     ユーザー名:<input type="text" name="userId"><br>
13     パスワード:<input type="password" name="password"><br>
14     <input type="submit" name="index" value="認証チェック" formaction="Login">
15     <input type="submit" name="index" value="新規登録" formaction="Login">
16     <input type="submit" name="index" value="入力訂正" formaction="Login">
17     <!--name と value は何でも いい --%>
18     <input type="submit" name="index" value="戻る" formaction="BookController">
19     <% if(request.getAttribute("message") != null){%>
20         <p><%= request.getAttribute("message") %></p>
21     <%}%>
22 </form>
23 </body>

```

User.java のソースコード

```
1  package model;
2
3  import java.io.Serializable;
4
5  public class User implements Serializable {
6      private String userId;
7      private String password;
8      private String userName;
9      private String userAddress;
10     private String userMail;
11     //登録済み送付先件数
12     private int num;
13     private String name1;
14     private String address1;
15     private String mail1;
16     private String name2;
17     private String address2;
18     private String mail2;
19     private String name3;
20     private String address3;
21     private String mail3;
22     private String name4;
23     private String address4;
24     private String mail4;
25     private String name5;
26     private String address5;
27     private String mail5;
28     //決定した送付先
29     private String sendName;
30     private String sendAddress;
31     private String sendMail;
```

```

32
33     public User() {}
34     //ユーザーID とパスワードがログインするための主キー
35     public User(String userId, String password) {
36         this.userId = userId;
37         this.password = password;
38     }
39     //ユーザーを新規登録する時
40     public User(String userId, String password, String userName, String userAddress,
41 String userMail) {
42         this.userId = userId;
43         this.password = password;
44         this.userName = userName;
45         this.userAddress = userAddress;
46         this.userMail = userMail;
47     }
48
49     public String getUserId() {
50         return userId;
51     }
52     public String getPassword() {
53         return password;
54     }
55     public String getUserName() {
56         return userName;
57     }
58     public String getUserAddress() {
59         return userAddress;
60     }
61     public String getUserMail() {
62         return userMail;
63     }
64
65     public void setUserName(String userName) {
66         this.userName = userName;
67     }

```

```
68     public void setUserAddress(String userAddress) {
69         this.userAddress = userAddress;
70     }
71     public void setUserMail(String userMail) {
72         this.userMail = userMail;
73     }
74     public int getNum() {
75         return num;
76     }
77     public void setNum(int num) {
78         this.num = num;
79     }
80
81     public String getName1() {
82         return name1;
83     }
84     public String getAddress1() {
85         return address1;
86     }
87     public String getMail1() {
88         return mail1;
89     }
90     public String getName2() {
91         return name2;
92     }
93     public String getAddress2() {
94         return address2;
95     }
96     public String getMail2() {
97         return mail2;
98     }
99     public String getName3() {
100         return name3;
101     }
102     public String getAddress3() {
103         return address3;
```

```

104     }
105     public String getMail3() {
106         return mail3;
107     }
108     public String getName4() {
109         return name4;
110     }
111     public String getAddress4() {
112         return address4;
113     }
114     public String getMail4() {
115         return mail4;
116     }
117     public String getName5() {
118         return name5;
119     }
120     public String getAddress5() {
121         return address5;
122     }
123     public String getMail5() {
124         return mail5;
125     }
126
127     public void setName1(String name1) {
128         this.name1 = name1;
129     }
130     public void setAddress1(String address1) {
131         this.address1 = address1;
132     }
133     public void setMail1(String mail1) {
134         this.mail1 = mail1;
135     }
136     public void setName2(String name2) {
137         this.name2 = name2;
138     }
139     public void setAddress2(String address2) {

```

```

140         this.address2 = address2;
141     }
142     public void setMail2(String mail2) {
143         this.mail2 = mail2;
144     }
145     public void setName3(String name3) {
146         this.name3 = name3;
147     }
148     public void setAddress3(String address3) {
149         this.address3 = address3;
150     }
151     public void setMail3(String mail3) {
152         this.mail3 = mail3;
153     }
154     public void setName4(String name4) {
155         this.name4 = name4;
156     }
157     public void setAddress4(String address4) {
158         this.address4 = address4;
159     }
160     public void setMail4(String mail4) {
161         this.mail4 = mail4;
162     }
163     public void setName5(String name5) {
164         this.name5 = name5;
165     }
166     public void setAddress5(String address5) {
167         this.address5 = address5;
168     }
169     public void setMail5(String mail5) {
170         this.mail5 = mail5;
171     }
172
173     public String getSendName() {
174         return sendName;
175     }

```



```

176     public String getSendAddress() {
177         return sendAddress;
178     }
179     public String getSendMail() {
180         return sendMail;
181     }
182
183     public void setSendName(String sendName) {
184         this.sendName = sendName;
185     }
186     public void setSendAddress(String sendAddress) {
187         this.sendAddress = sendAddress;
188     }
189     public void setSendMail(String sendMail) {
190         this.sendMail = sendMail;
191     }
192
193     //登録済送付先を削除するために、後の番号の氏名と住所を前に詰める
194     public void update1() {
195         setName1(getName2());
196         setName2(getName3());
197         setName3(getName4());
198         setName4(getName5());
199         setName5("");
200         setAddress1(getAddress2());
201         setAddress2(getAddress3());
202         setAddress3(getAddress4());
203         setAddress4(getAddress5());
204         setAddress5("");
205         setMail1(getMail2());
206         setMail2(getMail3());
207         setMail3(getMail4());
208         setMail4(getMail5());
209         setMail5("");
210     }
211     public void update2() {

```

```

212         setName2(getName3());
213         setName3(getName4());
214         setName4(getName5());
215         setName5("");
216         setAddress2(getAddress3());
217         setAddress3(getAddress4());
218         setAddress4(getAddress5());
219         setAddress5("");
220         setMail2(getMail3());
221         setMail3(getMail4());
222         setMail4(getMail5());
223         setMail5("");
224     }
225     public void update3() {
226         setName3(getName4());
227         setName4(getName5());
228         setName5("");
229         setAddress3(getAddress4());
230         setAddress4(getAddress5());
231         setAddress5("");
232         setMail3(getMail4());
233         setMail4(getMail5());
234         setMail5("");
235     }
236     public void update4() {
237         setName4(getName5());
238         setName5("");
239         setAddress4(getAddress5());
240         setAddress5("");
241         setMail4(getMail5());
242         setMail5("");
243     }
244     public void update5() {
245         setName5("");
246         setAddress5("");
247         setMail5("");

```

```
248     }
249 }
```

UserDAO.java のソースコード

```
1  package model;
2
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7
8  import javax.naming.InitialContext;
9  import javax.naming.NamingException;
10 import javax.sql.DataSource;
11
12 public class UserDAO {
13     //ログインしたユーザの登録済送付先をデータベースから取り出し User インス
14     タンスに追加
15     public User setList(User loginUser){
16         try {
17             InitialContext context = new InitialContext();
18             DataSource ds =
19 (DataSource)context.lookup("java:comp/env/jdbc/abcpool");
20             try(Connection db = ds.getConnection()){
21                 PreparedStatement ps = db.prepareStatement(
22                     "SELECT * FROM user_t where
23     userId = ?");
24                 ps.setString(1, loginUser.getUserId());
25                 ResultSet rs = ps.executeQuery();
26                 //レコードは1件なのでレコードポインタを一回だけ
27     動かす
28                 rs.next();
29                 loginUser.setUserName(rs.getString("userName"));
30
```

```

31         loginUser.setUserAddress(rs.getString("userAddress"));
32         loginUser.setUserMail(rs.getString("userMail"));
33         loginUser.setName1(rs.getString("name1"));
34         loginUser.setName2(rs.getString("name2"));
35         loginUser.setName3(rs.getString("name3"));
36         loginUser.setName4(rs.getString("name4"));
37         loginUser.setName5(rs.getString("name5"));
38         loginUser.setAddress1(rs.getString("address1"));
39         loginUser.setAddress2(rs.getString("address2"));
40         loginUser.setAddress3(rs.getString("address3"));
41         loginUser.setAddress4(rs.getString("address4"));
42         loginUser.setAddress5(rs.getString("address5"));
43         loginUser.setMail1(rs.getString("mail1"));
44         loginUser.setMail2(rs.getString("mail2"));
45         loginUser.setMail3(rs.getString("mail3"));
46         loginUser.setMail4(rs.getString("mail4"));
47         loginUser.setMail5(rs.getString("mail5"));
48     }
49     }catch (NamingException | SQLException e) {
50         e.printStackTrace();
51     }
52     return loginUser;
53 }
54
55 //User インスタンスの情報を取り出しデータベースを更新
56 public void updateList(User loginUser) {
57     try {
58         InitialContext context = new InitialContext();
59         DataSource ds =
60 (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
61         try(Connection db = ds.getConnection()){
62             PreparedStatement ps = db.prepareStatement(
63                 "UPDATE user_t SET
64 num=?,name1=?,name2=?,name3=?,name4=?,name5=?,address1=?,address2=?,address3
65 =?,address4=?,address5=?,mail1=?,mail2=?,mail3=?,mail4=?,mail5=? where userId = ?");
66             ps.setInt(1, loginUser.getNum());

```

```

67         ps.setString(2, loginUser.getName1());
68         ps.setString(3, loginUser.getName2());
69         ps.setString(4, loginUser.getName3());
70         ps.setString(5, loginUser.getName4());
71         ps.setString(6, loginUser.getName5());
72         ps.setString(7, loginUser.getAddress1());
73         ps.setString(8, loginUser.getAddress2());
74         ps.setString(9, loginUser.getAddress3());
75         ps.setString(10, loginUser.getAddress4());
76         ps.setString(11, loginUser.getAddress5());
77         ps.setString(12, loginUser.getMail1());
78         ps.setString(13, loginUser.getMail2());
79         ps.setString(14, loginUser.getMail3());
80         ps.setString(15, loginUser.getMail4());
81         ps.setString(16, loginUser.getMail5());
82         ps.setString(17, loginUser.getUserId());
83         ps.executeUpdate();
84     }
85     }catch (NamingException | SQLException e) {
86         e.printStackTrace();
87     }
88 }
89 }

```

LoginLogic.java のソースコード

```

1  package model;
2
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7
8  import javax.naming.Context;
9  import javax.naming.InitialContext;

```

```

10  import javax.naming.NamingException;
11  import javax.sql.DataSource;
12
13  public class LoginLogic {
14      public boolean execute(User user) {
15          //ユーザ ID とパスワードがどちらも入力されている場合
16          if(user.getUserId()!=" " && user.getPassword()!=" ") {
17              try {
18                  Context context = new InitialContext();
19                  DataSource ds =
20  (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
21                  try(Connection db = ds.getConnection()){
22                      //データベースからユーザ ID に対応するパ
23  スワードを取得
24                      PreparedStatement ps =
25  db.prepareStatement(
26                      "SELECT password FROM
27  user_t where userId=?");
28                      ps.setString(1, user.getUserId());
29                      ResultSet rs = ps.executeQuery();
30                      //レコードは 1 件なのでレコードポインタを
31  一回だけ動かす
32                      rs.next();
33                      //ユーザ ID に対応するパスワードと
34  index.jsp から送られたパスワードが一致するならログイン
35
36                      if(user.getPassword().equals(rs.getString("password"))) {return true;}
37                      }
38                      } catch (NamingException | SQLException e) {
39                          e.printStackTrace();
40                      }
41                  }
42                  //ログインできるのならここは処理しない
43                  return false;
44              }
45      }

```

Login.java のソースコード

```
1  package honya;
2
3
4  import java.io.IOException;
5  import java.sql.Connection;
6  import java.sql.PreparedStatement;
7  import java.sql.SQLException;
8
9  import javax.naming.Context;
10 import javax.naming.InitialContext;
11 import javax.naming.NamingException;
12 import javax.sql.DataSource;
13
14 import jakarta.servlet.ServletException;
15 import jakarta.servlet.annotation.WebServlet;
16 import jakarta.servlet.http.HttpServlet;
17 import jakarta.servlet.http.HttpServletRequest;
18 import jakarta.servlet.http.HttpServletResponse;
19 import jakarta.servlet.http.HttpSession;
20 import model.LoginLogic;
21 import model.User;
22 import model.UserDAO;
23
24 @WebServlet("/Login")
25 public class Login extends HttpServlet {
26
27     //http://localhost:8080/Mukunashi_syoten_ver1/Login で検索した場合
28     protected void doGet(HttpServletRequest request, HttpServletResponse response)
29     throws ServletException, IOException {
30         //ログインしていれば送付先選択画面へフォワード
31         if(request.getSession().getAttribute("loginUser") != null) {
32             this.getServletContext().getRequestDispatcher("/WEB-
```

```

33 INF/honya/addlessList.jsp").forward(request,response);
34
35         //ログインしていなければログイン画面へフォワード
36         }else {
37             this.getServletContext().getRequestDispatcher("/WEB-
38 INF/honya/login.jsp").forward(request,response);
39         }
40     }
41
42     protected void doPost(HttpServletRequest request, HttpServletResponse
43 response) throws ServletException, IOException {
44         //リクエストパラメータの取得
45         request.setCharacterEncoding("UTF-8");
46         String index = request.getParameter("index");
47         HttpSession session = request.getSession();
48
49         //login.jsp でボタンの新規登録かキャンセルを選んだ場合
50         if(index.equals("新規登録") || index.equals("キャンセル")) {
51             this.getServletContext().getRequestDispatcher("/WEB-
52 INF/honya/signUp.jsp").forward(request,response);
53
54             //signUp.jsp でボタンの登録を選んだ場合
55             }else if(index.equals("登録")) {
56                 //入力欄のどれかが空白だった場合
57                 if(request.getParameter("userId")=="" ||
58 request.getParameter("password")=="" || request.getParameter("name")=="" ||
59 request.getParameter("code")=="" || request.getParameter("address")=="" ||
60 request.getParameter("mail")==") {
61                     //ログイン失敗時は signUp.jsp に戻って、「入力欄に
62 空白があります」と表示する
63                     request.setAttribute("message","入力欄に空白があり
64 ます");
65
66                     this.getServletContext().getRequestDispatcher("/WEB-
67 INF/honya/signUp.jsp").forward(request,response);
68

```



```

69             }else {
70                 try {
71                     Context context = new InitialContext();
72                     DataSource ds =
73 (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
74                     try(Connection db = ds.getConnection()){
75                         //データベースからユーザ ID に対
76 応するパスワードを取得
77                         PreparedStatement ps =
78 db.prepareStatement(
79                                     "INSERT INTO
80 user_t(userId,password,userName,userAddress,userMail,num) values(?,?,?,?,?,0)");
81                         ps.setString(1,
82 request.getParameter("userId"));
83                         ps.setString(2,
84 request.getParameter("password"));
85                         ps.setString(3,
86 request.getParameter("name"));
87                         //郵便番号と住所をまとめてデータ
88 ベースに userAddress として登録する
89                         ps.setString(4,
90 request.getParameter("code") + "," + request.getParameter("address"));
91                         ps.setString(5,
92 request.getParameter("mail"));
93                         ps.executeUpdate();
94                     }
95                 } catch (NamingException | SQLException e) {
96                     e.printStackTrace();
97                 }
98                 //新規登録するユーザーの ID とパスワードと名前と
99 住所とメールアドレスで User インスタンスを作り、セッションスコープに保存
100                 User loginUser = new
101 User(request.getParameter("userId"), request.getParameter("password"),
102 request.getParameter("name"), request.getParameter("code") + "," +
103 request.getParameter("address"), request.getParameter("mail"));
104                 session.setAttribute("loginUser", loginUser);

```

```

105
106         this.getServletContext().getRequestDispatcher("/WEB-
107 INF/honya/addlessList.jsp").forward(request,response);
108     }
109
110         //login.jsp でボタンの認証チェックを選んだ場合
111     }else if(index.equals("認証チェック")){
112         String userId = request.getParameter("userId");
113         String password = request.getParameter("password");
114         //User インスタンス(ユーザー情報)の生成
115         User user = new User(userId, password);
116         //ログイン処理
117         LoginLogic loginLogic = new LoginLogic();
118         boolean isLogin = loginLogic.execute(user);
119
120         //ログイン成功時の処理
121         if(isLogin) {
122             //User インスタンスにユーザ ID ごとの登録済送付先
123             を追加
124             UserDao dao = new UserDao();
125             user = dao.setList(user);
126             //ユーザ情報をセッションスコープに保存
127             session.setAttribute("loginUser", user);
128             //ログイン中なら請求書とメールを作成する
129
130             this.getServletContext().getRequestDispatcher("/WEB-
131 INF/honya/addlessList.jsp").forward(request,response);
132
133         }else {
134             //ログイン失敗時は login.jsp に戻って、「ログインに
135             失敗しました」と表示する
136             request.setAttribute("message", "ログインに失敗しま
137             した");
138
139             this.getServletContext().getRequestDispatcher("/WEB-
140 INF/honya/login.jsp").forward(request,response);

```

```

141         }
142
143         //signUp.jsp でボタンの戻るを選んだ場合か login.jsp でボタンのキャン
144 セルを選んだ場合
145         }else if(index.equals("戻る") || index.equals("入力訂正")) {
146             this.getServletContext().getRequestDispatcher("/WEB-
147 INF/honya/login.jsp").forward(request,response);
148         }
149     }
150 }

```

signUp.jsp のソースコード

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>新規ユーザー登録</title>
8  </head>
9  <body>
10 <h1>新規ユーザー登録</h1>
11 <form method="post" action="?">
12 ユーザー名:<input type="text" name="userId"><br>
13 パスワード:<input type="password" name="password"><br>
14 氏名 : <input type="text" name="name"><br>
15 〒 : <input type="text" name="code">住所 : <input type="text" name="address"
16 size=30><br>
17 メールアドレス : <input type="text" name="mail"><br>
18 <button type="submit" name="index" value="登録" formaction="Login">登録</button>
19 <button type="submit" name="index" value="戻る" formaction="Login">戻る</button>
20 <button type="submit" name="index" value="キャンセル" formaction="Login">キャンセ
21 ル</button>
22 <!--入力欄に空白がある状態で登録ボタンを押したら「入力欄に空白があります」と表示

```

```

23  する --%>
24  <% if(request.getAttribute("message") != null){%>
25      <p><%= request.getAttribute("message")%></p>
26  <%}%>
27  </body>
28  </html>

```

adlessList.jsp のソースコード

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      import="java.sql.*,javax.naming.*,javax.sql.*,java.text.*"%>
3  <%@ page import="model.User"%>
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title>登録済み送付先</title>
9  </head>
10 <body>
11 <h1>登録済み送付先</h1>
12 <form method="POST" action="?">
13 <!--adlessList.jsp にフォワードされる時はすでにログインしている --%>
14 <% User loginUser = (User)session.getAttribute("loginUser");%>
15 <h4><%= loginUser.getUserId() + "様ログイン中" %>
16 <a style="margin-left: 30px" href="UserController">ログアウト</a></h4>
17
18 <table class="table">
19 <thead>
20 <tr>
21     <th>送付先番号</th><th>名前と住所とメールアドレス</th><th>削除</th>
22 </tr>
23 </thead>
24 <tbody>
25 <%
26  int num;

```

```

27 Context context = new InitialContext();
28 DataSource ds = (DataSource)context.lookup("java:comp/env/jdbc/abcdpool");
29 try(Connection db = ds.getConnection()){
30     //ログインしているユーザ ID の情報をデータベースから取得
31     PreparedStatement ps = db.prepareStatement("SELECT * FROM user_t where
32     userId = ?");
33     ps.setString(1, loginUser.getUserId());
34     ResultSet rs = ps.executeQuery();
35     //レコードは 1 件なのでレコードポインタを一回だけ動かす
36     rs.next();
37     //num は登録済み送付先の件数
38     num = rs.getInt("num");
39     //send は前回選んだ送付先の番号
40     String send = (String)session.getAttribute("send");
41     //i は送付先番号になる
42     for(int i = 1; i <= num; i++){
43         %>
44
45         <!--送付先の一件目もしくは前回選んだ送付先にだけラジオボタンのチェックを
46         入れる --%>
47         <tr><% if((i == 1 && send == null) || (send != null && Integer.parseInt(send)
48         == i)){ %>
49             <td rowspan="3"><input type="radio" name="send" value="<%=i %>"
50             checked><%=i %></td>
51             <% }else{ %>
52             <td rowspan="3"><input type="radio" name="send"
53             value="<%=i %>"><%=i %></td>
54             <% } %>
55             <!--送付先の氏名と住所はデータベースで列名の後ろに番号を 1 から 5
56             まで付けている --%>
57             <td><%=rs.getString("name"+i) %></td>
58             <td rowspan="3"><input type="checkbox" name="delete"
59             value="<%=i %>"></td>
60         </tr>
61         <tr><td><%=rs.getString("address" + i) %></td></tr>
62         <tr><td><%=rs.getString("mail" + i) %></td></tr>

```

```

63  <% }
64  } %>
65  </tbody>
66  </table>
67
68  <!--登録済送付先の件数である num も表示せずに送信する --%>
69  <input type="hidden" name="num" value="<%=num %>">
70  <% if(num > 0){%>
71      <button type="submit" name="controller" value="送付先決定"
72  formaction="UserController">送付先決定</button>
73  <% }
74  if(num < 5){ %>
75      <button type="submit" name="controller" value="新規追加"
76  formaction="UserController">新規追加</button>
77  <% }
78  if(num > 0){ %>
79      <button type="submit" name="controller" value="削除実行"
80  formaction="UserController">削除実行</button>
81  <% } %>
82  <!--name と value は何でも いい --%>
83  <button type="submit" name="controller" value="戻る" formaction="BookController">戻
84  る</button>
85  </form>
86  </body>
87  </html>

```

UserController.java のソースコード

```

1  package honya;
2
3  import java.io.IOException;
4  import java.util.Date;
5  import java.util.List;
6  import java.util.Timer;
7  import java.util.TimerTask;

```

```

8
9  import jakarta.servlet.ServletException;
10 import jakarta.servlet.annotation.WebServlet;
11 import jakarta.servlet.http.HttpServlet;
12 import jakarta.servlet.http.HttpServletRequest;
13 import jakarta.servlet.http.HttpServletResponse;
14 import jakarta.servlet.http.HttpSession;
15 import model.Book;
16 import model.BookDAO;
17 import model.User;
18 import model.UserDAO;
19
20 @WebServlet("/UserController")
21 public class UserController extends HttpServlet {
22
23     // 「ログアウトしてトップ画面に戻る」のリンクをクリックした場合
24     protected void doGet(HttpServletRequest request, HttpServletResponse response)
25     throws ServletException, IOException {
26         // セッションスコープを破棄
27         request.getSession().invalidate();
28         // index.jsp にフォワード
29         this.getServletContext().getRequestDispatcher("/WEB-INF/honya/form.jsp").forward(request, response);
30     }
31
32
33     protected void doPost(HttpServletRequest request, HttpServletResponse
34 response) throws ServletException, IOException {
35         request.setCharacterEncoding("UTF-8");
36         String controller = request.getParameter("controller");
37         HttpSession session = request.getSession();
38         // ログインしたユーザの個人情報をセッションスコープから取得
39         User loginUser = (User)session.getAttribute("loginUser");
40
41         // addressList.jsp で戻るボタンをクリックした場合は選んだ送付先の番号をセッションスコープに保存
42         if(request.getParameter("send") != null) {

```

```

44         session.setAttribute("send", request.getParameter("send"));
45     }
46     UserDao dao = new UserDao();
47
48     //addressList.jsp でボタンの送付先決定を選ぶ
49     if(controller.equals("送付先決定")){
50         //送付先番号の名前と住所を注文完了画面へ送る
51         String send = request.getParameter("send");
52         switch(send) {
53             case "1" -> {
54
55                 loginUser.setSendName(loginUser.getName1());
56
57                 loginUser.setSendAddress(loginUser.getAddress1());
58                 loginUser.setSendMail(loginUser.getMail1());
59             }
60             case "2" -> {
61
62                 loginUser.setSendName(loginUser.getName2());
63
64                 loginUser.setSendAddress(loginUser.getAddress2());
65                 loginUser.setSendMail(loginUser.getMail2());
66             }
67             case "3" -> {
68
69                 loginUser.setSendName(loginUser.getName3());
70
71                 loginUser.setSendAddress(loginUser.getAddress3());
72                 loginUser.setSendMail(loginUser.getMail3());
73             }
74             case "4" -> {
75
76                 loginUser.setSendName(loginUser.getName4());
77
78                 loginUser.setSendAddress(loginUser.getAddress4());
79                 loginUser.setSendMail(loginUser.getMail4());

```



```

80         }
81         case "5" -> {
82
83             loginUser.setSendName(loginUser.getName5());
84
85             loginUser.setSendAddress(loginUser.getAddress5());
86             loginUser.setSendMail(loginUser.getMail5());
87         }
88     }
89     BookDAO Bdao = new BookDAO();
90     List<Book> list = (List<Book>)session.getAttribute("list");
91     for(Book bok: list) {
92         //購入する本の在庫数から購入冊数を引く
93         Bdao.reduceBook(bok);
94     }
95     //UNIX エポックから経過したミリ秒を PDF の名前とする
96     long unix = new Date().getTime();
97     //請求書を作るクラスのメソッドを実行
98     new PdfSeikyusho().doGet(request, response, unix);
99     //送り状を作るクラスのメソッドを実行
100    new PdfOkurizyo().doGet(request, response, unix);
101    //請求書付きのメールを送信するためのクラスのメソッドを使
102    うためにオブジェクトを作成
103    Mail mail = new Mail();
104    //初期化パラメータを読み込み
105    mail.init(getServletConfig());
106    Timer timer = new Timer(false);
107    TimerTask task = new TimerTask() {
108        @Override
109        public void run() {
110            try {
111                //請求書付きのメールを送信する
112                mail.doGet(request, response, unix,
113                loginUser);
114                //購入した本のリストと合計金額と
115                メール送信の TimerTask を持つセッションを破棄する

```

```

116             session.removeAttribute("list");
117             session.removeAttribute("sum");
118             session.removeAttribute("task");
119         } catch (ServletException | IOException e) {
120             e.printStackTrace();
121         }
122     }
123 };
124 //請求書付きのメールを送信するのを 5 秒後に実行する
125 timer.schedule(task, 5000);
126 session.setAttribute("task", task);
127 this.getServletContext().getRequestDispatcher("/WEB-
128 INF/honya/payment.jsp").forward(request,response);
129
130 //addlessList.jsp でボタンの新規追加を選ぶか addlessEntry.jsp でボタン
131 の入力訂正を選ぶ
132 }else if(controller.equals("新規追加") || controller.equals("入力訂正")) {
133     this.getServletContext().getRequestDispatcher("/WEB-
134 INF/honya/addlessEntry.jsp").forward(request,response);
135
136 //addlessList.jsp でボタンの削除実行を選ぶか addlessEntry.jsp でボタン
137 の戻るを選ぶ
138 }else if(controller.equals("削除実行") || controller.equals("戻る")) {
139     //addlessList.jsp で削除にチェックを入れた送付先
140     String[] delete = request.getParameterValues("delete");
141
142     //addlessList.jsp で一つでもチェックしていた場合
143     if(delete != null) {
144         //削除する送付先の番号の配列を昇順から降順に並べ
145         替えた配列を作る
146         String[] reverseDelete = new String[delete.length];
147         for(int i = 0; i < delete.length; i++) {
148             reverseDelete[i] = delete[delete.length - i - 1];
149         }
150         //num は現在の登録済み送付先件数
151         int num =

```

```

152 Integer.parseInt(request.getParameter("num"));
153 //send は選んだ送付先の番号
154 int send =
155 Integer.parseInt(request.getParameter("send"));
156 for(String del:reverseDelete) {
157 //登録済み送付先件数を一つ減らす
158 loginUser.setNum(--num);
159 //チェックした送付先番号の大きいほうから
160 氏名と住所を消して、後の番号の氏名と住所を前に詰める
161 switch(del) {
162     case "1" -> loginUser.update1();
163     case "2" -> loginUser.update2();
164     case "3" -> loginUser.update3();
165     case "4" -> loginUser.update4();
166     case "5" -> loginUser.update5();
167 }
168 //選んだ送付先より番号が小さい送付先を削
169 除した場合は番号を減らす
170 if(Integer.parseInt(del) < send) {
171     send--;
172 //選んだ送付先と番号が同じ送付先を削除し
173 た場合は番号を 1 にする
174 }else if(Integer.parseInt(del) == send) {
175     send = 1;
176 }
177 }
178 //セッションスコープの送付先番号を更新する
179 session.setAttribute("send", Integer.toString(send));
180 //ログインしたユーザの個人情報をデータベースで更
181 新する
182 dao.updateList(loginUser);
183 }
184 this.getServletContext().getRequestDispatcher("/WEB-
185 INF/honya/addressList.jsp").forward(request,response);
186 }
187

```

```

188         //addlessEntry.jsp でボタンの登録を選ぶ
189         else if(controller.equals("登録")) {
190
191             //入力欄のどれかが空白だった場合
192             if(request.getParameter("name")=="" ||
193 request.getParameter("code")=="" || request.getParameter("address")=="" ||
194 request.getParameter("mail")==") {
195                 //ログイン失敗時は signUp.jsp に戻って、「入力欄に
196 空白があります」と表示する
197                 request.setAttribute("message", "入力欄に空白があり
198 ます");
199
200         this.getServletContext().getRequestDispatcher("/WEB-
201 INF/honya/addlessEntry.jsp").forward(request,response);
202
203         }else {
204             //登録済み送付先件数を一つ増やす
205             int num =
206 Integer.parseInt(request.getParameter("num"))+1;
207             loginUser.setNum(num);
208             //ログインしたユーザのインスタンスで登録済み送付
209 先件数の番号の名前と住所を更新する
210             switch(num) {
211                 case 1 -> {
212
213                     loginUser.setName1(request.getParameter("name"));
214                                     //郵便番号と住所をまとめてユーザ
215 インスタンスに address として保存する
216
217                     loginUser.setAddress1(request.getParameter("code") + "," +
218 request.getParameter("address"));
219
220                     loginUser.setMail1(request.getParameter("mail"));
221                 }
222                 case 2 -> {
223

```

```

224         loginUser.setName2(request.getParameter("name"));
225
226         loginUser.setAddress2(request.getParameter("code") + "," +
227 request.getParameter("address"));
228
229         loginUser.setMail2(request.getParameter("mail"));
230     }
231     case 3 -> {
232
233         loginUser.setName3(request.getParameter("name"));
234
235         loginUser.setAddress3(request.getParameter("code") + "," +
236 request.getParameter("address"));
237
238         loginUser.setMail3(request.getParameter("mail"));
239     }
240     case 4 -> {
241
242         loginUser.setName4(request.getParameter("name"));
243
244         loginUser.setAddress4(request.getParameter("code") + "," +
245 request.getParameter("address"));
246
247         loginUser.setMail4(request.getParameter("mail"));
248     }
249     case 5 -> {
250
251         loginUser.setName5(request.getParameter("name"));
252
253         loginUser.setAddress5(request.getParameter("code") + "," +
254 request.getParameter("address"));
255
256         loginUser.setMail5(request.getParameter("mail"));
257     }
258 }
259 //ログインしたユーザの個人情報をデータベースで更

```

```

260 新する
261
262
263         this.getServletContext().getRequestDispatcher("/WEB-
264 INF/honya/addlessList.jsp").forward(request,response);
265     }
266 }
267 }
268 }

```

addlessEntry.jsp のソースコード

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2    pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>新規送付先登録</title>
8  </head>
9  <body>
10 <h1>新規送付先登録</h1>
11 <form method="POST" action="?">
12 氏名:<input type="text" name="name"><br>
13 〒 : <input type="text" name="code">住所 : <input type="text" name="address"
14 size=30><br>
15 メールアドレス:<input type="text" name="mail"><br>
16 <button type="submit" name="controller" value="登録" formaction="UserController">登
17 録</button>
18 <button type="submit" name="controller" value="戻る" formaction="UserController">戻
19 る</button>
20 <button type="submit" name="controller" value="入力訂正"
21 formaction="UserController">入力訂正</button>
22 <%--num は登録済み送付先件数 --%>
23 <input type="hidden" name="num" value="<%=request.getParameter("num") %>">

```

```
24 </form>
25 <!--入力欄に空白がある状態で登録ボタンを押したら「入力欄に空白があります」と表示
26 する --%>
27 <% if(request.getAttribute("message") != null){%>
28     <p><%= request.getAttribute("message")%></p>
29 <%}%>
30 </body>
31 </html>
```

PdfSeikyusho.java のソースコード

```
1 package honya;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.util.List;
6
7 import com.itextpdf.barcodes.Barcode128;
8 import com.itextpdf.barcodes.BarcodeQRCode;
9 import com.itextpdf.kernel.colors.Color;
10 import com.itextpdf.kernel.colors.ColorConstants;
11 import com.itextpdf.kernel.colors.DeviceRgb;
12 import com.itextpdf.kernel.font.PdfFont;
13 import com.itextpdf.kernel.font.PdfFontFactory;
14 import com.itextpdf.kernel.geom.Rectangle;
15 import com.itextpdf.kernel.pdf.PdfDocument;
16 import com.itextpdf.kernel.pdf.PdfPage;
17 import com.itextpdf.kernel.pdf.PdfWriter;
18 import com.itextpdf.kernel.pdf.canvas.PdfCanvas;
19 import com.itextpdf.kernel.pdf.xobject.PdfFormXObject;
20 import com.itextpdf.layout.Canvas;
21 import com.itextpdf.layout.Document;
22 import com.itextpdf.layout.element.Cell;
23 import com.itextpdf.layout.element.Image;
24 import com.itextpdf.layout.element.Paragraph;
```

```

25 import com.itextpdf.layout.element.Table;
26 import com.itextpdf.layout.element.Text;
27 import com.itextpdf.layout.property.HorizontalAlignment;
28 import com.itextpdf.layout.property.TextAlignment;
29
30 import jakarta.servlet.ServletException;
31 import jakarta.servlet.annotation.WebServlet;
32 import jakarta.servlet.http.HttpServlet;
33 import jakarta.servlet.http.HttpServletRequest;
34 import jakarta.servlet.http.HttpServletResponse;
35 import jakarta.servlet.http.HttpSession;
36 import model.Book;
37 import model.User;
38
39 @WebServlet("/PdfSeikyusho")
40 //result.jsp で購入するボタンを押した時にメールに添付する請求書を PDF で作成
41 public class PdfSeikyusho extends HttpServlet {
42     //unix は UNIX エポックから経過したミリ秒であり保存した PDF の名前
43     protected void doGet(HttpServletRequest request, HttpServletResponse response,
44 long unix) throws ServletException, IOException {
45         //作成した PDF を動的 Web プロジェクトに保存するために、プロパテ
46 ィーのロケーションで File クラスのオブジェクトを作成
47         PdfDocument pdf = new PdfDocument(
48             new PdfWriter(new
49 File("../workspace20240415/Mukunashi_syoten_ver1/src/main/webapp/seikyusho/" + unix
50 + ".pdf")));
51         Document doc = new Document(pdf);
52         PdfFont font = PdfFontFactory.createFont(
53             "HeiseiKakuGo-W5", "UniJIS-UCS2-H");
54         doc.setFont(font);
55         HttpSession session = request.getSession();
56         User loginUser = (User)session.getAttribute("loginUser");
57         //ユーザーの名前を PDF に書く
58         doc.add(new Paragraph(loginUser.getUserName() + " 様
59 ").setFontSize(24)
60             .add(new Text("

```

棕梨


```

61     書店").setFontSize(20));
62         Table table = new Table(new float[] {255, 50, 70});
63         String[] headers = {"書名", "価格", "購入冊数"};
64         for(String header : headers) {
65             Cell cell = new Cell();
66             cell.setFontSize(12)
67                 .setBold()
68
69             .setHorizontalAlignment(HorizontalAlignment.CENTER)
70                 .setTextAlignment(TextAlignment.CENTER)
71                 .add(new Paragraph(header));
72             table.addCell(cell);
73         }
74         //セッションスコープから購入する本のリストと合計金額をキャスト
75         List<Book> list = (List<Book>)session.getAttribute("list");
76         String sum = (String)session.getAttribute("sum");
77         //購入する本の種類ごとに表の一行で情報を表示する
78         for(Book bok: list) {
79             //クラス型の配列は要素数の宣言とクラスの宣言を別々にする
80             Cell[] dcell = new Cell[3];
81             for(int i = 0; i < 3; i++) dcell[i] = new Cell();
82             dcell[0].add(new Paragraph(bok.getTitle()));
83             dcell[1].add(new Paragraph(String.valueOf(bok.getPrice())));
84             dcell[2].add(new Paragraph(String.valueOf(bok.getNum())));
85             //一行分のセル三つをテーブルに加える
86             for(int j = 0; j < 3; j++) table.addCell(dcell[j]);
87         }
88         doc.add(table);
89         doc.add(new Paragraph("合計金額 : ").setFontSize(24)
90             .add(
91                 //合計金額の数値の Text だけ青色にする
92                 new Text(sum)
93                 .setFontColor(new DeviceRgb(0, 0, 255))
94                 ).add(new Text("円")
95             ));
96         //「0123456789」と書かれたバーコードを表の下に作成する

```

```

97         doc.add(createBarcode("0123456789", pdf));
98         // 棕梨書店の URL で QR コードを作成
99
100        createQRcode("http://localhost:8080/Mukunashi_syoten_ver1/BookController",p
101 df);
102        doc.close();
103    }
104
105    // バーコードの下に入れる文字と請求書の PDF を引数にする
106    private static Cell createBarcode(String code, PdfDocument pdfDoc) {
107        Barcode128 barcode = new Barcode128(pdfDoc);
108        barcode.setCode(code);
109        Color myColor = new DeviceRgb(255, 100, 50);
110        PdfFormXObject barcodeObject = barcode.createFormXObject(myColor,
111 null, pdfDoc);
112        Cell cell = new Cell().add(new Image(barcodeObject));
113        cell.setPaddingTop(000);
114        cell.setPaddingRight(200);
115        cell.setPaddingBottom(000);
116        cell.setPaddingLeft(200);
117        return cell;
118    }
119
120    // QL コードが表す URL と請求書の PDF を引数にする
121    private static void createQRcode(String url, PdfDocument pdfDocument) {
122        PdfPage pdfPage = pdfDocument.getFirstPage();
123        PdfCanvas pdfCanvas = new PdfCanvas(pdfPage);
124        Rectangle rect = new Rectangle(250, 760, 50, 50);
125
126        BarcodeQRCode barcodeQRCode = new BarcodeQRCode(url);
127        PdfFormXObject pdfFormXObject =
128
129        barcodeQRCode.createFormXObject(ColorConstants.BLACK, pdfDocument);
130        Image qrCodeImage =
131            new
132        Image(pdfFormXObject).setWidth(rect.getWidth());

```

```

133
134         setHeight(rect.getHeight());
135
136         Canvas qrCanvas = new Canvas(pdfCanvas, rect);
137         qrCanvas.add(qrCodeImage);
138         qrCanvas.close();
139     }
140 }

```

PdfOkurizyo.java のソースコード

```

1  package honya;
2
3  import java.io.File;
4  import java.io.IOException;
5  import java.time.LocalDate;
6  import java.time.format.DateTimeFormatter;
7  import java.util.List;
8
9  import com.itextpdf.barcodes.Barcode128;
10 import com.itextpdf.kernel.colors.Color;
11 import com.itextpdf.kernel.colors.DeviceRgb;
12 import com.itextpdf.kernel.font.PdfFont;
13 import com.itextpdf.kernel.font.PdfFontFactory;
14 import com.itextpdf.kernel.pdf.PdfDocument;
15 import com.itextpdf.kernel.pdf.PdfWriter;
16 import com.itextpdf.kernel.pdf.xobject.PdfFormXObject;
17 import com.itextpdf.layout.Document;
18 import com.itextpdf.layout.element.Cell;
19 import com.itextpdf.layout.element.Image;
20 import com.itextpdf.layout.element.Paragraph;
21 import com.itextpdf.layout.element.Table;
22 import com.itextpdf.layout.element.Text;
23 import com.itextpdf.layout.property.HorizontalAlignment;
24 import com.itextpdf.layout.property.TextAlignment;

```

```

25
26 import jakarta.servlet.ServletException;
27 import jakarta.servlet.annotation.WebServlet;
28 import jakarta.servlet.http.HttpServlet;
29 import jakarta.servlet.http.HttpServletRequest;
30 import jakarta.servlet.http.HttpServletResponse;
31 import jakarta.servlet.http.HttpSession;
32 import model.Book;
33 import model.User;
34
35 @WebServlet("/Okurizyo")
36 //result.jsp で購入するボタンを押した時にメールに添付する送り状を PDF で作成
37 public class PdfOkurizyo extends HttpServlet {
38     //unix は UNIX エポックから経過したミリ秒であり保存した PDF の名前
39     protected void doGet(HttpServletRequest request, HttpServletResponse response,
40 long unix) throws ServletException, IOException {
41         //作成した PDF を動的 Web プロジェクトに保存するために、プロパテ
42 ィーのロケーションで File クラスのオブジェクトを作成
43         PdfDocument pdf = new PdfDocument(
44             new PdfWriter(new
45 File("../workspace20240415/Mukunashi_syoten_ver1/src/main/webapp/okurizyo/" + unix
46 + ".pdf")));
47         Document doc = new Document(pdf);
48         PdfFont font = PdfFontFactory.createFont(
49             "HeiseiKakuGo-W5", "UniJIS-UCS2-H");
50         doc.setFont(font);
51         HttpSession session = request.getSession();
52         User loginUser = (User)session.getAttribute("loginUser");
53         //日付を「xxxx/xx/xx」と出力する
54         var dateFormat = DateTimeFormatter.ofPattern("y/M/d");
55         //郵便番号と住所と日付を PDF に書く。
56         doc.add(new Paragraph(loginUser.getSendAddress()).setFontSize(20)
57             .add(new Text("
58 "+LocalDate.now().format(dateFormat))));
59         //送付先の名前を PDF に書く
60         doc.add(new Paragraph(loginUser.getSendName() + "様

```

```

61 ").setFontSize(20));
62         doc.add(new Paragraph("下記は請求書のバーコードです
63 ").setFontSize(16)
64             .add(new Text("                福岡市東区千早4丁目2-4
65 -1"))));
66         //「0123456789」と書かれたバーコードをユーザー名の下に作成する
67         doc.add(createBarcode("0123456789", pdf));
68         //文字を中央に表示する
69         doc.add(new Paragraph("納品のご案内
70 ").setFontSize(24).setTextAlignment(TextAlignment.CENTER));
71         doc.add(new Paragraph("拝啓平素は格別のご哀願を賜りまして誠にあり
72 かとございます。").setFontSize(11));
73         //注文したユーザーの名前をPDFに書く
74         doc.add(new Paragraph(loginUser.getUserName() + "様より依頼された
75 商品を送付いたしますので、ご査収のほどよろしくお願いいたします。
76 ").setFontSize(11));
77         Table table = new Table(new float[] {255, 70});
78         String[] headers = {"書名", "購入冊数"};
79         for(String header : headers) {
80             Cell cell = new Cell();
81             cell.setFontSize(12)
82                 .setBold()
83
84             .setHorizontalAlignment(HorizontalAlignment.CENTER)
85                 .setTextAlignment(TextAlignment.CENTER)
86                 .add(new Paragraph(header));
87             table.addCell(cell);
88         }
89         //セッションスコープから購入する本のリストをキャスト
90         List<Book> list = (List<Book>)session.getAttribute("list");
91         //購入する本の種類ごとに表の一行で情報を表示する
92         for(Book bok: list) {
93             //クラス型の配列は要素数の宣言とクラスの宣言を別々にする
94             Cell[] dcell = new Cell[2];
95             for(int i = 0; i < 2; i++) dcell[i] = new Cell();
96             dcell[0].add(new Paragraph(bok.getTitle()));

```

```

97         dcell[1].add(new Paragraph(String.valueOf(bok.getNum())));
98         //一行分のセル二つをテーブルに加える
99         for(int j = 0; j < 2; j++) table.addCell(dcell[j]);
100     }
101     doc.add(table);
102     doc.add(new Paragraph("以上
103 り主 "+loginUser.getUserName()).setFontSize(16));
104     doc.close();
105 }
106
107 //バーコードの下に入れる文字と請求書の PDF を引数にする
108 private static Cell createBarcode(String code, PdfDocument pdfDoc) {
109     Barcode128 barcode = new Barcode128(pdfDoc);
110     barcode.setCode(code);
111     Color myColor = new DeviceRgb(255, 100, 50);
112     PdfFormXObject barcodeObject = barcode.createFormXObject(myColor,
113 null, pdfDoc);
114     Cell cell = new Cell().add(new Image(barcodeObject));
115     cell.setPaddingTop(20);
116     cell.setPaddingRight(200);
117     cell.setPaddingBottom(000);
118     cell.setPaddingLeft(70);
119     return cell;
120 }
121 }

```

送

Mail.java のソースコード

```

1 package honya;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.io.UnsupportedEncodingException;
6 import java.util.Properties;
7

```

```

8  import javax.activation.DataHandler;
9  import javax.activation.FileDataSource;
10 import javax.mail.Authenticator;
11 import javax.mail.Message;
12 import javax.mail.MessagingException;
13 import javax.mail.PasswordAuthentication;
14 import javax.mail.Session;
15 import javax.mail.Transport;
16 import javax.mail.internet.InternetAddress;
17 import javax.mail.internet.MimeBodyPart;
18 import javax.mail.internet.MimeMessage;
19 import javax.mail.internet.MimeMultipart;
20 import javax.mail.internet.MimeUtility;
21
22 import jakarta.servlet.ServletConfig;
23 import jakarta.servlet.ServletContext;
24 import jakarta.servlet.ServletException;
25 import jakarta.servlet.http.HttpServlet;
26 import jakarta.servlet.http.HttpServletRequest;
27 import jakarta.servlet.http.HttpServletResponse;
28 import model.User;
29
30 public class Mail extends HttpServlet {
31     // メールサーバーの情報を格納するためのプライベート変数
32     private String host = null;
33     private String port = null;
34     private String user = null;
35     private String password = null;
36     private String charset = null;
37
38     //サーブレットの初期化時に初期化パラメータを読み込み
39     @Override
40     //config はサーブレット初期化パラメーターにアクセスする手段を提供
41     public void init(ServletConfig config) throws ServletException{
42         //HttpServlet クラスでもともと定義された init メソッドを実行
43         super.init(config);

```

```

44         //app はユーザー間で共有可能な情報
45         ServletContext app = config.getServletContext();
46         //web.xml の<param-name>から<param-value>を取得
47         this.host = app.getInitParameter("smtp.host");
48         this.port = app.getInitParameter("smtp.port");
49         //user と password はパスワード認証のためのユーザー名とパスワード
50         this.user = app.getInitParameter("smtp.user");
51         this.password = app.getInitParameter("smtp.password");
52         //charset はメールの文章の文字符号化方式
53         this.charset = app.getInitParameter("smtp.charset");
54     }
55
56     //unix はUNIX エポックから経過したミリ秒であり保存する PDF の名前、logiUser
57     はログインしたユーザーのインスタンス
58     protected void doGet(HttpServletRequest request, HttpServletResponse response,
59     long unix, User loginUser) throws ServletException, IOException {
60         //送信時のパラメータ情報を設定
61         Properties props = new Properties();
62         //メールサーバーのホスト名
63         props.setProperty("mail.smtp.host", this.host);
64         //メールサーバーのポート番号
65         props.setProperty("mail.smtp.port", this.port);
66         //ユーザー認証を有効にするか
67         props.setProperty("mail.smtp.auth", "true");
68         //connectiontimeout はサーバーとの接続が確立するまでに許容する時間
69         props.setProperty("mail.smtp.connectiontimeout", "60000");
70         //timeout はサーバーが送信するのに許容する時間
71         props.setProperty("mail.smtp.timeout", "60000");
72         //メールサーバーへのセッションを確立
73         Session session = Session.getInstance(props, new Authenticator() {
74             //パスワード認証のための情報のオブジェクトで、認証クラスの
75             オブジェクトを作成し、セッションオブジェクトに組み込む
76             protected PasswordAuthentication getPasswordAuthentication()
77             {
78                 return new PasswordAuthentication(user, password);
79             }
79         });

```



```

80         });
81         //請求書を送るメールを作成
82         try {
83             MimeMessage message = new MimeMessage(session);
84             //MIME マルチパートを準備
85             MimeMultipart multipart = new MimeMultipart();
86
87             //テキスト本体を設定
88             MimeBodyPart part1 = new MimeBodyPart();
89             //テキストを設定
90             part1.setText(loginUser.getUserName() + "様へ棕梨書店から請
91 求書です。", charset);
92             //個別のデータが入った MimeBodyPart を MimeMultipart に梱
93 包する
94             multipart.addBodyPart(part1);
95
96             //添付ファイルを準備
97             MimeBodyPart part2 = new MimeBodyPart();
98             //データを取得するためのハンドラーを設定
99             DataHandler handler = new DataHandler(
100                 //動的 Web プロジェクトに保存した PDF を
101                 添付するために、プロパティのロケーションで File クラスのオブジェクトを作成
102                 new FileDataSource(new
103                 File("../workspace20240415/Mukunashi_syoten_ver1/src/main/webapp/seikyusho/" + unix
104                 + ".pdf")));
105             //データを設定
106             part2.setDataHandler(handler);
107             //添付ファイルに画像の名前を設定
108
109             part2.setFileName(MimeUtility.encodeWord(handler.getName()));
110             multipart.addBodyPart(part2);
111
112             //送信先／元を設定
113             message.setFrom(new InternetAddress(
114                 "mukunashi@example.com", " 棕 梨 書 店 ",
115             charset));

```

```

116         message.addRecipient(Message.RecipientType.TO, new
117     InternetAddress(
118         loginUser.getUserMail(),
119     loginUser.getUserName(), charset));
120         message.setSubject("棕梨書店の請求書", charset);
121         //メール本体にマルチパートを設定
122         message.setContent(multipart);
123         Transport.send(message);
124     }catch(MessagingException | UnsupportedEncodingException e) {
125         e.printStackTrace();
126     }
127
128     //送り状を送るメールを作成
129     try {
130         MimeMessage message = new MimeMessage(session);
131         //MIME マルチパートを準備
132         MimeMultipart multipart = new MimeMultipart();
133
134         //テキスト本体を設定
135         MimeBodyPart part1 = new MimeBodyPart();
136         //テキストを設定
137         part1.setText(loginUser.getSendName() + "様へ棕梨書店から送
138     り状です。", charset);
139         //個別のデータが入った MimeBodyPart を MimeMultipart に梱
140     包する
141         multipart.addBodyPart(part1);
142
143         //添付ファイルを準備
144         MimeBodyPart part2 = new MimeBodyPart();
145         //データを取得するためのハンドラーを設定
146         DataHandler handler = new DataHandler(
147             //動的 Web プロジェクトに保存した PDF を
148     添付するために、プロパティのロケーションで File クラスのオブジェクトを作成
149             new FileDataSource(new
150     File("../workspace20240415/Mukunashi_syoten_ver1/src/main/webapp/okurizyo/" + unix
151     + ".pdf"))));

```

```

152         //データを設定
153         part2.setDataHandler(handler);
154         //添付ファイルに画像の名前を設定
155
156         part2.setFileName(MimeUtility.encodeWord(handler.getName()));
157         multipart.addBodyPart(part2);
158
159         //送信先／元を設定
160         message.setFrom(new InternetAddress(
161             "mukunashi@example.com", "  棕  梨  書  店  ",
162             charset));
163         message.addRecipient(Message.RecipientType.TO,      new
164             InternetAddress(
165                 loginUser.getSendMail(),
166                 loginUser.getSendName(), charset));
167         message.setSubject("棕梨書店の送り状", charset);
168         //メール本体にマルチパートを設定
169         message.setContent(multipart);
170         Transport.send(message);
171     }catch(MessagingException | UnsupportedEncodingException e) {
172         e.printStackTrace();
173     }
174 }
175 }

```

payment.jsp のソースコード

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <%@ page import="java.util.TimerTask" %>
4  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
6  <%@ page import="model.User"%>
7  <!DOCTYPE html>
8  <html>

```

```

9    <head>
10   <meta charset="UTF-8">
11   <script src="js/script2.js" charset="utf-8" defer></script>
12   <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
13   <title>注文完了</title>
14   </head>
15   <body>
16   <%--請求書を PDF で表示するが、ブラウザの「戻る」をクリックすればここに戻る -
17   -%>
18   <h1>注文完了しました。</h1>
19   <% User loginUser = (User)session.getAttribute("loginUser");%>
20   <h4><%= loginUser.getUserId() + "様ログイン中" %>
21   <a style="margin-left: 30px" href="UserController">ログアウト</a></h4>
22   <%--result.jsp に戻ったら購入冊数はすべて 0 になる --%>
23   <a href="BookController">本を選ぶ画面に戻る</a>
24
25   <em><%--script2.jsp で「5 秒以内なら注文をキャンセルできます」の 5 をカウントダウ
26   ン --%></em>
27   <form method="POST" action="BookController">
28   <span><%--script2.jsp で「購入をキャンセルする」ボタンを表示するが、このボタンは
29   五秒後に消える --%></span>
30   <hr>
31   <%--合計金額を改めて表示する --%>
32   <strong>合計金額：<%=session.getAttribute("sum")%>円</strong><br>
33
34   <%--送付先を改めて表示する --%>
35   <table class="table">
36   <thead>
37   <tr>
38       <th>送付先</th><th>名前</th><th>住所</th><th>メールアドレス</th>
39   </tr>
40   </thead>
41   <tbody>
42   <tr>
43       <td></td><td><%=loginUser.getSendName() %></td><td><%=loginUser.ge
44   tSendAddress() %></td><td><%=loginUser.getSendMail() %></td>

```

```

45     </tr>
46 </tbody>
47 </table>
48
49 <%--購入した本のリストを改めて表示する --%>
50 <table class="table">
51 <thead>
52 <tr>
53     <th>注文</th><th>書名</th><th>価格</th><th>出版社</th><th>刊行年月
54 日</th><th>購入冊数</th>
55 </tr>
56 </thead>
57 <tbody>
58 <c:forEach var="item" items="${sessionScope['list']}">
59     <tr>
60         <td></td>
61         <td>${fn:escapeXml(item.title)}</td>
62         <td>${fn:escapeXml(item.price)}</td>
63         <td>${fn:escapeXml(item.publish)}</td>
64         <td>${fn:escapeXml(item.published)}</td>
65         <td>${fn:escapeXml(item.num)}</td>
66     </tr>
67 </c:forEach>
68 </tbody>
69 </table>
70 </form>
71 </body>
72 </html>

```

script2.js のソースコード

```

1 //payment.jsp に組み込む
2 'use strict';
3 $('span').html("<input type='¥'submit¥" id='¥'btn¥" value='¥'購入をキャンセルする¥">");
4 $('em').html(`<br>5 秒以内なら注文をキャンセルできます`);

```

```

5   let count = 4;
6
7   //payment.jsp を表示してから、5 秒から 1 秒までカウントダウンを表示する
8   const timerID = setInterval(function(){
9       //0 秒になったら何も表示しないでカウントダウンを止める
10      if(count == 0){
11          $('em').html("");
12          return clearInterval(timerID);
13      }
14      $('em').html(`<br>${count}秒以内なら注文をキャンセルできます`);
15      count--;
16  },
17  1000
18 );
19
20 //メールが送信される、payment.jsp を表示してから五秒後に「購入をキャンセルする」
21 ボタンが表示されなくなる
22 setTimeout(
23     function(){
24         $('span').html("<input type='hidden' id='btn' value='購入をキャ
25 ンセルする'>");
26     },
27     5000
28 );

```

web.xml のソースコード

```

1   <?xml version="1.0" encoding="UTF-8"?>
2   <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="https://jakarta.ee/xml/ns/jakartaee"
4   xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
5   https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd" id="WebApp_ID" version="5.0">
6       <display-name>basket_canwait_login</display-name>
7       <context-param>
8           <param-name>book.title</param-name>

```

```

9      <param-value>独習 JSP & サーブレット</param-value>
10    </context-param>
11    <context-param>
12      <param-name>smtp.host</param-name>
13      <param-value>localhost</param-value>
14    </context-param>
15    <context-param>
16      <param-name>smtp.port</param-name>
17      <param-value>25</param-value>
18    </context-param>
19    <context-param>
20      <param-name>smtp.user</param-name>
21      <param-value>xxxxx</param-value>
22    </context-param>
23    <context-param>
24      <param-name>smtp.password</param-name>
25      <param-value>xxxxx</param-value>
26    </context-param>
27    <context-param>
28      <param-name>smtp.charset</param-name>
29      <param-value>ISO-2022-JP</param-value>
30    </context-param>
31    <display-name>book_basket_mail</display-name>
32    <welcome-file-list>
33      <welcome-file>index.html</welcome-file>
34      <welcome-file>index.jsp</welcome-file>
35      <welcome-file>index.htm</welcome-file>
36      <welcome-file>default.html</welcome-file>
37      <welcome-file>default.jsp</welcome-file>
38      <welcome-file>default.htm</welcome-file>
39    </welcome-file-list>
40    <servlet>
41      <description></description>
42      <display-name>BookController</display-name>
43      <servlet-name>BookController</servlet-name>
44      <servlet-class>honya.BookController</servlet-class>

```

```

45     </servlet>
46     <servlet>
47         <description></description>
48         <display-name>PdfSeikyusho</display-name>
49         <servlet-name>PdfSeikyusho</servlet-name>
50         <servlet-class>honya.PdfSeikyusho</servlet-class>
51     </servlet>
52     <servlet>
53         <description></description>
54         <display-name>Mail</display-name>
55         <servlet-name>Mail</servlet-name>
56         <servlet-class>honya.Mail</servlet-class>
57     </servlet>
58     <servlet>
59         <description></description>
60         <display-name>UserController</display-name>
61         <servlet-name>UserController</servlet-name>
62         <servlet-class>servlet.UserController</servlet-class>
63     </servlet>
64     <servlet>
65         <description></description>
66         <display-name>Login</display-name>
67         <servlet-name>Login</servlet-name>
68         <servlet-class>servlet.Login</servlet-class>
69     </servlet>
70     <servlet>
71         <description></description>
72         <display-name>PdfOkurizyo</display-name>
73         <servlet-name>PdfOkurizyo</servlet-name>
74         <servlet-class>honya.PdfOkurizyo</servlet-class>
75     </servlet>
76     <servlet-mapping>
77         <servlet-name>PdfOkurizyo</servlet-name>
78         <url-pattern>/PdfOkurizyo</url-pattern>
79     </servlet-mapping>
80 </web-app>

```


context.xml のソースコード

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE configuration>
3  <Context reloadable="true"> <Resource name="jdbc/abcdpool" auth="Container"
4  type="javax.sql.DataSource" driverClassName="com.mysql.cj.jdbc.Driver"
5  url="jdbc:mysql://localhost:3306/basic?allowPublicKeyRetrieval=true&useSSL=false
6  &useUnicode=true&characterEncoding=UTF-8&serverTimezone=JST"
7  username="root" password="rootroot" maxTotal="4" maxWaitMillis="5000" maxIdle="2"
8  validationQuery="SELECT count(*) from usr"> </Resource>
9  </Context>
10 <!-- 光安聖徳 16 -->
```

データベースの中身

book テーブル

book テーブルには ISBN を主キーとして、form.jsp で表示する商品の書名、価格、出版社、刊行年月日、在庫数量が入っている。

フィールド

Field	Type	Null	Key	Default	Extra
isbn	char(17)	NO	PRI	NULL	
title	varchar(100)	YES		NULL	
price	int	YES		NULL	
publish	varchar(20)	YES		NULL	
published	date	YES		NULL	
zaiko	int	YES		NULL	

データ

isbn	title	price	publish	published	zaiko
978-4-7980-5759-0	はじめてのAndroidアプリ開発	3200	秀和システム	2019-08-10	99
978-4-7980-5804-7	はじめてのASP.NET Webフォームアプリ開発	3200	秀和システム	2019-10-12	99
978-4-7981-5112-0	独習Java新版	3150	翔泳社	2019-05-15	100
978-4-7981-5757-3	JavaScript逆引きレシピ	2730	翔泳社	2018-10-15	98
978-4-7981-6044-3	Androidアプリ開発の教科書	2850	翔泳社	2019-07-10	100
978-4-7981-6365-9	独習ASP.NET Webフォーム	3800	翔泳社	2020-02-17	100
978-4-8026-1226-5	SQLデータ分析・活用入門	2600	ソシム	2019-09-12	100
978-4-8156-0182-9	これからはじめるVue.js実践入門	3990	SBクリエイティブ	2019-08-22	100
978-4-8222-5391-2	iphoneアプリ超入門	2200	日経BP	2020-02-28	98
978-4-8222-8653-8	基礎からしっかり学ぶc#の教科書	2900	日経BP	2019-12-20	100

user_t テーブル

user_t テーブルにはユーザー名とパスワードを主キーとして、氏名と住所とメールアドレスと登録している送付先の件数と、送付先五件ごとに氏名と住所とメールアドレスが入っている。

フィールド

Field	Type	Null	Key	Default	Extra
userId	varchar(32)	NO	PRI	NULL	
password	varchar(32)	NO	PRI	NULL	
userName	varchar(64)	YES		NULL	
userAddress	varchar(128)	YES		NULL	
userMail	varchar(32)	YES		NULL	
num	int	YES		NULL	
name1	varchar(64)	YES		NULL	
address1	varchar(128)	YES		NULL	
mail1	varchar(32)	YES		NULL	
name2	varchar(64)	YES		NULL	
address2	varchar(128)	YES		NULL	
mail2	varchar(32)	YES		NULL	
name3	varchar(64)	YES		NULL	
address3	varchar(128)	YES		NULL	
mail3	varchar(32)	YES		NULL	
name4	varchar(64)	YES		NULL	
address4	varchar(128)	YES		NULL	
mail4	varchar(32)	YES		NULL	
name5	varchar(64)	YES		NULL	
address5	varchar(128)	YES		NULL	
mail5	varchar(32)	YES		NULL	

データ

userId	password	userName	userAddress	userMail	num
userC	9012	光安聖徳	062-0921,北海道札幌市豊平区中の島1条8-3-18	sapporo@gmail.com	1
userD	3456	聖徳太子	123-4567,奈良県飛鳥市十七町1-1-1	test2@gmail.com	3

name1	address1	mail1	name2	address2	mail2
佐藤洋一	103-8888,東京都南東京市白町3-2-1	satou@example.com	NULL	NULL	NULL
山田太郎	103-9999,東京都東京市南町1-1-1	yamada@example.com	鈴木浩二	504-1111,神奈川県松戸市金町2-4-5	suzuki@example.com

name3	address3	mail3	name4	address4	mail4	name5	address5	mail5
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
井上春子	904-2222,埼玉県浦安市本町4-7-9	inoue@example.com						

入力した送付先が小さな番号に詰められる。