武汉纺织大学

Java应用开发课程设计

**EMS物流下单程序**

**学 院： 数学与计算机学院**

**班 级： 软件11802班**

**姓 名： 欧阳行**

**学 号： 1804240624**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年6月16日**

目录

[1.需求分析 1](#_Toc43228172)

[1.1主界面 1](#_Toc43228173)

[1.2读取excel信息 1](#_Toc43228174)

[1.3显示快递信息 1](#_Toc43228175)

[1.4生成快递图片以及PDF 1](#_Toc43228176)

[2.系统设计 2](#_Toc43228177)

[2.1用户用例图： 2](#_Toc43228178)

[2.2UML类图 2](#_Toc43228179)

[2.3UML活动图 3](#_Toc43228180)

[3.系统结构 4](#_Toc43228181)

[3.1项目架构 4](#_Toc43228182)

[3.2文件夹 4](#_Toc43228183)

[3.3VO类 express.java 5](#_Toc43228184)

[3.4SERVICE类 ExcelService.java 8](#_Toc43228185)

[3.5SERVICE类PictureService.java 10](#_Toc43228186)

[3.6SERVICE类PDFservic.java 14](#_Toc43228187)

[3.7用户界面Driver.java 16](#_Toc43228188)

[4.系统测试 19](#_Toc43228189)

[4.1读取信息： 19](#_Toc43228190)

[4.2显示快递信息： 19](#_Toc43228191)

[4.3查看生成的图片： 20](#_Toc43228192)

[5.系统总结 27](#_Toc43228193)

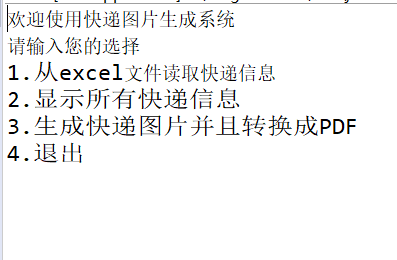
[6.致谢 27](#_Toc43228194)

# 1.需求分析

如今网络速度飞速发展，各种各样的网购平台层出不穷，如今网上购物已经完全进入到每个人的生活中网上购物已经成为每个人生活中不可或缺的一部分，对于网上购物最重要的一部分就是快递的运输，而快递的运输最重要的是快递图片的生成，因此如何批量高效的解决这个问题是我们需要结解决的，因此我做了这个ems快递物流单生成程序，其主要实现对于excel中的物流信息（快递单号随机生成格式为规则：yyyyMMddhhmmssSSS+三位随机号），批量生成快递图片，并且可支持转换成PDF。

## 1.1主界面

程序运行，显示操作选项如下图所示：选择自己需要操作的选项



## 1.2读取excel信息

用户选择选项1系统从指定excel文件中读取物流信息并将物流信息放入类集中供后续操作提供

## 1.3显示快递信息

用户选择选项3时将从excel读取到的信息输出到显示面板上

## 1.4生成快递图片以及PDF

用户选择选项3系统批量将之前读取的快递信息批量生成快递图片以及PDF文件，并且存储到指定路径。

# 2.系统设计

## 2.1用户用例图：

该系统只有管理员角色，可以读取，显示快递信息，并且生成图片和PDF

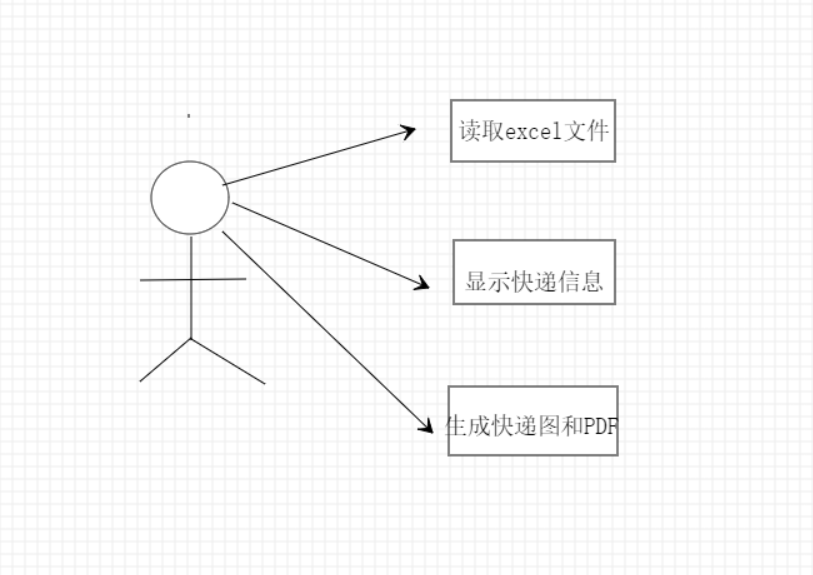
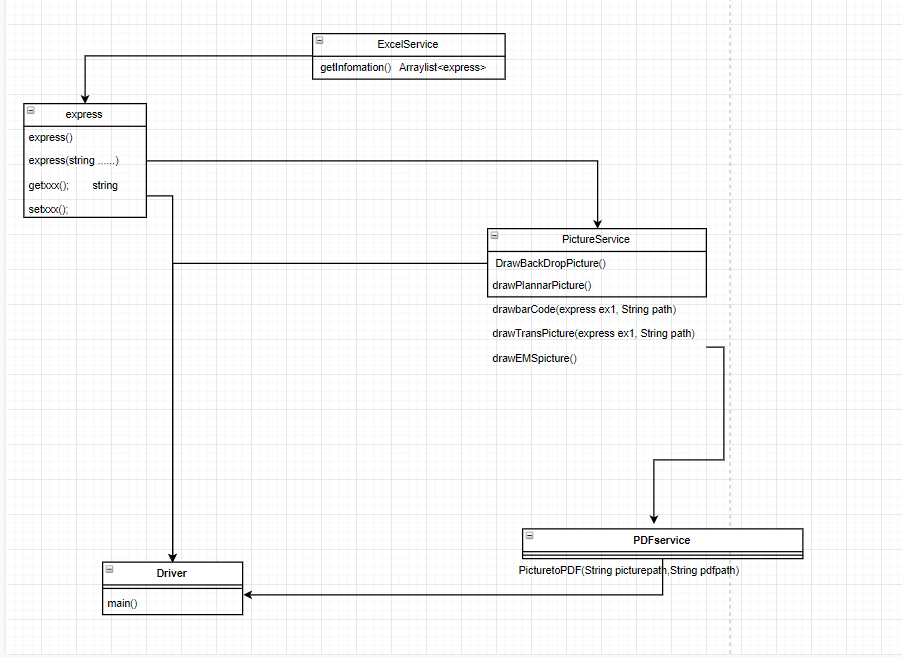


图2-1 用户用例图

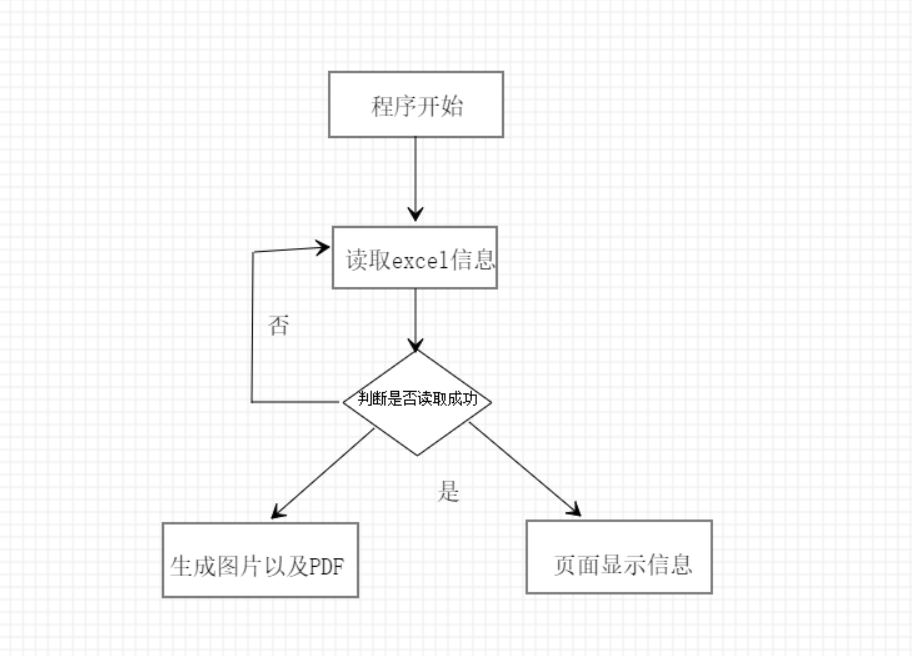
## 2.2UML类图

* Vo实体类express 与excel中表格结构进行映射的类，主要由属性，get，set tostring方法组成，express类的属性与excel表中的字段对应
* 业务逻辑实现类ExcelService PDFservice PictureService 分别实现对生成图片以及生成PDF需要的各种方法经行整合
* Drive类，主函数，整个系统逻辑顺序在这里决定。

各类的结构及类之间的关系如图

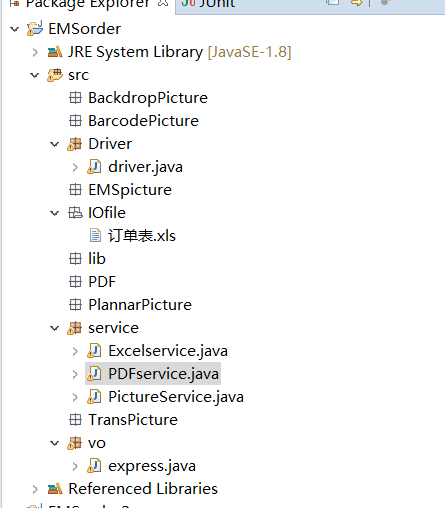


## 2.3UML活动图

****

# 3.系统结构

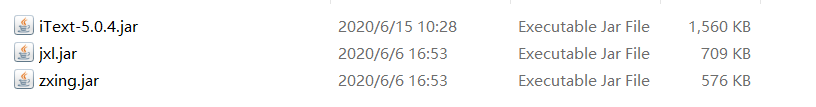
## 3.1项目架构

****

## 3.2文件夹

**Lib文件夹：**

存放次系统需要的除jdk自带包以外别的包如下图所示：



iText包：与生成PDF文件相关包

jxl包：读取excel文件相关包

zxing包：生成条形码以及二维码相关包

**Iofile文件夹：**

存放需要读取的EXCEL文件

**BackdropPicture文件夹：**

存放需要生成的背景框架图

**BarcodePicture文件夹：**

存放根据订单号生成的条形码图片

**TransPicture文件夹：**

存放根据物流单号生成的条形码图片

**PlannarPicture文件夹：**

存放生成的二维码

**EMSpicture文件夹**：

存放最终生成的EMS快递图片

**PDF文件夹：**

## 3.3VO类 express.java

该类用来实例化快递这一实体，里面包含的属性为快递图片上需要的信息

代码如下：

**package** vo;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Date;

**public** **class** express {

**private** String num;//快递编号

**private** String order;//订单号

**private** String order2;//物流单号

**private** String weight;//重量

**private** String count;//数量

**private** String product;//包含产品

**private** String ReceName;//收件人姓名

**private** String ReceZipCode;//收件人邮编

**private** String ReceAdress;//收件人地址

**private** String ReceTele;//收件人电话

**private** String SenName;//发件人姓名

**private** String SenZipCode;//发件人邮编

**private** String SenAdress;//发件人地址

**private** String SenTele;//发件人电话

**public** express() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** express(String num,String order2,String weight, String count, String product, String receName,

String receZipCode, String receAdress, String receTele, String senName, String senZipCode, String senAdress,

String senTele) {

**super**();

//订单号为时间加上随机数生成

SimpleDateFormat dF2 = **new** SimpleDateFormat("yyyyMMddhhmmss");//获得当前系统时间按照规定格式

Date today = **new** Date();

String date = dF2.format(today);

**int** i= **new** java.util.Random().nextInt(900)+100;

String s=String.*valueOf*(i);//订单号：时间加三位随机数

**this**.order=date+s;

**this**.order2=order2;

**this**.num = num;

**this**.weight = weight;

**this**.count = count;

**this**.product=product;

ReceName = receName;

ReceZipCode = receZipCode;

ReceAdress = receAdress;

ReceTele = receTele;

SenName = senName;

SenZipCode = senZipCode;

SenAdress = senAdress;

SenTele = senTele;

}

**public** String getOrder2() {

**return** order2;

}

**public** **void** setOrder2(String order2) {

**this**.order2 = order2;

}

**public** String getNum() {

**return** num;

}

**public** **void** setNum(String num) {

**this**.num = num;

}

**public** String getOrder() {

**return** order;

}

**public** **void** setOrder(String order) {

**this**.order = order;

}

**public** String getWeight() {

**return** weight;

}

**public** **void** setWeight(String weight) {

**this**.weight = weight;

}

**public** String getCount() {

**return** count;

}

**public** **void** setCount(String count) {

**this**.count = count;

}

**public** String getProduct() {

**return** product;

}

**public** **void** setProduct(String product) {

**this**.product = product;

}

**public** String getReceName() {

**return** ReceName;

}

**public** **void** setReceName(String receName) {

ReceName = receName;

}

**public** String getReceZipCode() {

**return** ReceZipCode;

}

**public** **void** setReceZipCode(String receZipCode) {

ReceZipCode = receZipCode;

}

**public** String getReceAdress() {

**return** ReceAdress;

}

**public** **void** setReceAdress(String receAdress) {

ReceAdress = receAdress;

}

**public** String getReceTele() {

**return** ReceTele;

}

**public** **void** setReceTele(String receTele) {

ReceTele = receTele;

}

**public** String getSenName() {

**return** SenName;

}

**public** **void** setSenName(String senName) {

SenName = senName;

}

**public** String getSenZipCode() {

**return** SenZipCode;

}

**public** **void** setSenZipCode(String senZipCode) {

SenZipCode = senZipCode;

}

**public** String getSenAdress() {

**return** SenAdress;

}

**public** **void** setSenAdress(String senAdress) {

SenAdress = senAdress;

}

**public** String getSenTele() {

**return** SenTele;

}

**public** **void** setSenTele(String senTele) {

SenTele = senTele;

}

@Override

**public** String toString() {

**return** "express [num=" + num + ", order=" + order + ", order2=" + order2 + ", weight=" + weight + ", count=" + count

+ ", product=" + product + ", ReceName=" + ReceName + ", ReceZipCode=" + ReceZipCode + ", ReceAdress="

+ ReceAdress + ", ReceTele=" + ReceTele + ", SenName=" + SenName + ", SenZipCode=" + SenZipCode

+ ", SenAdress=" + SenAdress + ", SenTele=" + SenTele + "]";

}

}

## 3.4SERVICE类 ExcelService.java

主要实现读取excel中的信息，并且将其放入类集中

代码如下：

**package** service;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** jxl.Cell;

**import** jxl.Sheet;

**import** jxl.Workbook;

**import** jxl.read.biff.BiffException;

**import** vo.express;

**public** **class** Excelservice {

**public** **static** ArrayList<express> getinformatio() **throws** Exception//从excel中读取文件得到订单数据

{

File f = **new** File("src/IOfile/订单表.xls");

Workbook wok = Workbook.*getWorkbook*(f);

Sheet sh = wok.getSheet(0);

ArrayList<express> expList=**new** ArrayList<express>();

**int** flag=-1;//标记此时操作的是运单数据集合中的第几个

**for**(**int** i=1;i<sh.getRows();i++)

{

express ex1=**null**;

ArrayList<String> list=**new** ArrayList<String>();//获得数据集合

Cell c1=sh.getCell(0,i);

**if**(c1.getContents().length()!=0)//判断第一列是否位空，不为空的话则生产新的运单数据

{

**for**(**int** j=0;j<sh.getColumns();j++)

{

Cell c2=sh.getCell(j,i);

list.add(c2.getContents());//将该行所有数据添加进数据集合中

}

ex1=**new** express(list.get(0), list.get(1), list.get(3),list.get(4), list.get(5), list.get(6),list.get(7), list.get(8), list.get(9), list.get(10), list.get(11), list.get(12), list.get(13));

expList.add(ex1);

flag++;

}

**else** {//第一行为空的话将数据中商品名称添加进属于他的数据集合中

Cell c3=sh.getCell(5,i);

express ex2=expList.get(flag);//得到这一行应该添加进的运单对象

String product=ex2.getProduct();

product=product+c3.getContents();//将新的商品名称加入进去

ex2.setProduct(product);//将添加新的数据后的运单对象替换原来的对象

expList.set(flag, ex2);

}

}

**return** expList;

}

}

## 3.5SERVICE类PictureService.java

主要功能首先是生成背景框架图，其次根据express中的订单号以及物流单号生成条形码图片，最后将所有信息画到背景图片上得到最终EMS图片

代码如下：

**package** service;

**import** java.awt.BasicStroke;

**import** java.awt.Color;

**import** java.awt.Font;

**import** java.awt.Graphics;

**import** java.awt.Graphics2D;

**import** java.awt.Image;

**import** java.awt.geom.Line2D;

**import** java.awt.image.BufferedImage;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.Hashtable;

**import** javax.imageio.ImageIO;

**import** com.google.zxing.BarcodeFormat;

**import** com.google.zxing.EncodeHintType;

**import** com.google.zxing.MultiFormatWriter;

**import** com.google.zxing.WriterException;

**import** com.google.zxing.client.j2se.MatrixToImageWriter;

**import** com.google.zxing.common.BitMatrix;

**import** com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;

**import** vo.express;

**public** **class** PictureService {

**public** **static** **void** DrawBackDropPicture() **throws** Exception// 画该运单图片的背景框架

{

// 设置背景图长宽

**int** imageWidth = 1000;

**int** imageHeight = 1200;

BufferedImage image = **new** BufferedImage(imageWidth, imageHeight, BufferedImage.***TYPE\_INT\_RGB***);

Graphics graphics = image.getGraphics();//获得图片画笔工具

Graphics2D graphics2d = (Graphics2D) image.getGraphics();

graphics.setColor(Color.***WHITE***);

graphics.fillRect(0, 0, imageWidth, imageHeight);

graphics.setColor(**new** Color(220, 240, 240));

graphics.setColor(Color.***black***);

graphics.drawRect(0, 0, 900, 1200);//画整体矩形框架

graphics2d.setColor(Color.***black***);

BasicStroke dashed = **new** BasicStroke(2.5f, BasicStroke.***CAP\_BUTT***, BasicStroke.***JOIN\_ROUND***, 3.5f,

**new** **float**[] { 10, 5 }, 0f);

graphics2d.setStroke(dashed);//获得虚线画笔工具

graphics.drawLine(0, 1190, 900, 1190);

**int** start = 0;

**for** (**int** i = 0; i < 6; i++) {//动态画矩形中分界线

start = start + 150;

graphics2d.draw(**new** Line2D.Double(0, start, 900, start));// (x,y,z,w) x:开始时x坐标；y：开始时y坐标，z：线段长度，：结束时y坐标

}

graphics2d.draw(**new** Line2D.Double(450, 450, 450, 600));

graphics2d.draw(**new** Line2D.Double(0, 1050, 600, 1050));

graphics2d.draw(**new** Line2D.Double(600, 900, 600, 1200));

ImageIO.*write*(image, "jpg", **new** File("src/BackdropPicture/back.jpg"));

}

**public** **static** **void** drawPlannarPicture() {//生产二维码图片，扫描结果是ems官网

**int** width = 250;

**int** height = 250;

String contents = "www.ems.com";

String imgPath = "src/PlannarPicture/plannar.jpg";//图片生成位置

Hashtable<Object, Object> hints = **new** Hashtable<Object, Object>();

// 指定纠错等级

hints.put(EncodeHintType.***ERROR\_CORRECTION***, ErrorCorrectionLevel.***L***);

// 指定编码格式

hints.put(EncodeHintType.***CHARACTER\_SET***, "GBK");

**try** {

BitMatrix bitMatrix = **new** MultiFormatWriter().encode(contents, BarcodeFormat.***QR\_CODE***, width, height);

MatrixToImageWriter.~~writeToFile~~(bitMatrix, "jpg", **new** File(imgPath));

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **static** **void** drawbarCode(express ex1, String path) **throws** Exception {//画订单号条形码

**int** codeWidth = 3 + // start guard

(7 \* 6) + // left bars

5 + // middle guard

(7 \* 6) + // right bars

3; // end guard

**int** width = 100, height = 50;

String contents = ex1.getOrder();

codeWidth = Math.*max*(codeWidth, width);

//条形码编码为CODE\_128格式

BitMatrix bitMatrix = **new** MultiFormatWriter().encode(contents, BarcodeFormat.***CODE\_128***, codeWidth, height, **null**);

MatrixToImageWriter.~~writeToFile~~(bitMatrix, "jpg", **new** File(path));

}

**public** **static** **void** drawTransPicture(express ex1, String path) **throws** Exception {//画物流单号二维码

**int** codeWidth = 3 + // start guard

(7 \* 6) + // left bars

5 + // middle guard

(7 \* 6) + // right bars

3; // end guard

**int** width = 100, height = 50;

String contents = ex1.getOrder2();

codeWidth = Math.*max*(codeWidth, width);

BitMatrix bitMatrix = **new** MultiFormatWriter().encode(contents, BarcodeFormat.***CODE\_128***, codeWidth, height, **null**);

MatrixToImageWriter.~~writeToFile~~(bitMatrix, "jpg", **new** File(path));

}

**public** **static** **void** drawEMSpicture(express ex1, String BackPath, String plannarPath, String BarPath,

String TransPath, String EMSPath) **throws** Exception {//画最后生成的快递图片

BufferedImage img = **new** BufferedImage(1000, 1200, BufferedImage.***TYPE\_INT\_RGB***);//生成一张新的图片

BufferedImage BarcodePicture = ImageIO.*read*(**new** File(BarPath));//获得订单号条形码图

BufferedImage BackPicture = ImageIO.*read*(**new** File(BackPath));//获得背景框架图

BufferedImage PlannarPicture = ImageIO.*read*(**new** File(plannarPath));//获得二维码图片

BufferedImage TransPicture = ImageIO.*read*(**new** File(TransPath));//获得物流单号图片

Graphics g = img.getGraphics();

g.drawImage(BackPicture.getScaledInstance(1000, 1200, Image.***SCALE\_DEFAULT***), 0, 0, **null**);//将背景框架画进图片中

g.drawImage(TransPicture.getScaledInstance(450, 90, Image.***SCALE\_DEFAULT***), 250, 20, **null**);//将物流单号条形码画入图片

g.drawImage(PlannarPicture.getScaledInstance(250, 250, Image.***SCALE\_DEFAULT***), 630, 930, **null**);//将二维码画入图片

g.drawImage(BarcodePicture.getScaledInstance(550, 85, Image.***SCALE\_DEFAULT***), 180, 620, **null**);//将订单号划入图片

g.drawImage(TransPicture.getScaledInstance(450, 90, Image.***SCALE\_DEFAULT***), 200, 770, **null**);

// 画ems

g.setColor(Color.***black***);

g.setFont(**new** Font("宋体", Font.***BOLD***, 75));

g.drawString("EMS", 70, 80);

g.setFont(**new** Font("宋体", Font.***BOLD***, 30));

g.drawString("e标准", 90, 120);

g.setFont(**new** Font("宋体", Font.***BOLD***, 30));

g.drawString(ex1.getOrder2(), 400, 140);// 画运单号

g.drawString(ex1.getOrder2(), 350, 890);

g.drawString(ex1.getOrder(), 300, 740);// 画订单号

// 画收件人信息

g.drawString("收件人:" + ex1.getReceName() + " 电话：" + ex1.getReceTele(), 20, 340);

g.drawString("地址：" + ex1.getReceAdress(), 20, 390);

g.drawString("邮编：" + ex1.getReceZipCode(), 20, 440);

// 画内件详情

g.drawString("内件详情：", 20, 190);

String product[] = ex1.getProduct().split("克");//因可能产品有多个故动态画，将名称字符串划分，避免画出图片过界

**for** (**int** i = 0; i < product.length; i++) {

g.drawString(product[i], 160, 190 + i \* 40);

}

// 画内件重量数量

g.drawString("内件数量：" + ex1.getCount(), 20, 490);

g.drawString("总重量：" + ex1.getWeight(), 20, 540);

// 画签收界面

g.setFont(**new** Font("宋体", Font.***BOLD***, 25));

g.drawString("收件人/代收件人：", 470, 490);

g.drawString("签收时间： 年 月 日", 470, 570);

// 画发件人信息

g.setFont(**new** Font("宋体", Font.***BOLD***, 28));

g.drawString("发件人:" + ex1.getSenName() + " 电话：" + ex1.getSenTele(), 15, 940);

g.drawString("地址：" + ex1.getSenAdress(), 15, 990);

g.drawString("邮编：" + ex1.getSenZipCode(), 15, 1040);

// 画地下栏收件人信息

g.setFont(**new** Font("宋体", Font.***BOLD***, 22));

g.drawString("收件人:" + ex1.getReceName() + " 电话：" + ex1.getReceTele(), 12, 1090);

g.drawString("地址：" + ex1.getReceAdress(), 12, 1140);

g.drawString("邮编：" + ex1.getReceZipCode(), 12, 1180);

// 画二维码下方网址即联系电话：

g.setFont(**new** Font("宋体", Font.***BOLD***, 26));

g.drawString("www.ems.com.cn", 650, 1160);

g.drawString("2017 电话：11183", 650, 1185);

/// \*\*\*\*\*\*\*

g.setFont(**new** Font("宋体", Font.***BOLD***, 26));

g.drawString("订单号", 50, 635);

// 画陆运信息

g.setFont(**new** Font("宋体", Font.***BOLD***, 80));

g.drawString("B-C", 700, 850);

g.setFont(**new** Font("宋体", Font.***BOLD***, 50));

g.drawString("全程", 300, 510);

g.drawString("陆运", 300, 570);

ImageIO.*write*(img, "jpg", **new** File(EMSPath));

}

}

## 3.6SERVICE类PDFservic.java

主要功能将生成的EMS图片转化成PDF格式

代码如下：

**package** service;

**import** java.awt.Image;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** com.itextpdf.text.Document;

**import** com.itextpdf.text.DocumentException;

**import** com.itextpdf.text.PageSize;

**import** com.itextpdf.text.Paragraph;

**import** com.itextpdf.text.pdf.PdfWriter;

**public** **class** PDFservice {

**public** **static** **void** PicturetoPDF(String picturepath, String pdfpath) **throws** Exception, DocumentException {

String tAGString = "PdfManager";

Document doc = **new** Document(PageSize.***A4***, 20, 20, 20, 20);// 得到pdf绘制工具

PdfWriter.*getInstance*(doc, **new** FileOutputStream(pdfpath));

doc.open();

doc.newPage();

com.itextpdf.text.Image png1 = com.itextpdf.text.Image.*getInstance*(picturepath);// 得到快递运单图片

**float** heigth = png1.getHeight();// 获得图片高度

**float** width = png1.getWidth();// 获得图片宽度

png1.setAlignment(com.itextpdf.text.Image.***MIDDLE***);

**int** percent = *getPercent2*(heigth, width);

png1.scalePercent(percent + 3);// 表示是原来图像的比例;

doc.add(png1); // 将图片写入pdf中

doc.close();

File f = **new** File(pdfpath);

f.createNewFile();

}

**public** **static** **int** getPercent(**float** h, **float** w) {

**int** p = 0;

**float** p2 = 0.0f;

**if** (h > w) {

p2 = 297 / h \* 100;

} **else** {

p2 = 210 / w \* 100;

}

p = Math.*round*(p2);

**return** p;

}

**public** **static** **int** getPercent2(**float** h, **float** w) {

**int** p = 0;

**float** p2 = 0.0f;

p2 = 530 / w \* 100;

p = Math.*round*(p2);

**return** p;

}

}

## 3.7用户界面Driver.java

主要功能，主函数，实现系统功能，以及系统逻辑功能

代码如下：

**package** Driver;

**import** java.awt.image.BufferedImage;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**import** javax.imageio.ImageIO;

**import** javax.print.attribute.standard.Sides;

**import** com.itextpdf.text.xml.simpleparser.IanaEncodings;

**import** jxl.read.biff.File;

**import** service.Excelservice;

**import** service.PDFservice;

**import** service.PictureService;

**import** vo.express;

**public** **class** driver {

**public** **static** **void** main(String[] args) **throws** Exception {

*meanu*();

Scanner input = **new** Scanner(System.***in***);

**int** choice;

ArrayList<express> explist = **new** ArrayList<express>();

**while** ((choice = input.nextInt()) != 0) {

**switch** (choice) {

**case** 1:

explist = Excelservice.*getinformatio*();// 获得所有运单信息

System.***out***.println("共获取到" + explist.size() + "条信息");

**break**;

**case** 2:

**if** (explist.size() == 0)// 判断是否读取了信息

{

System.***out***.println("您还没读取信息请重新输入");

**break**;

} **else** {

**for** (**int** i = 0; i < explist.size(); i++) {

System.***out***.println(explist.get(i).toString());

}

**break**;

}

**case** 3:

**if** (explist.size() == 0)// 判断是否读取了信息

{

System.***out***.println("您还没读取信息请重新输入");

**break**;

} **else** {

PictureService.*drawPlannarPicture*();// 画二维码

PictureService.*DrawBackDropPicture*();// 画背景图

// 动态生成订单号图片

String barPath = "src/BarcodePicture/";

**int** flag = 1;// 标记文件位置，动态生成不覆盖

**for** (**int** i = 0; i < explist.size(); i++) {

express ex1 = explist.get(i);

String path = barPath + String.*valueOf*(flag) + ".jpg";

PictureService.*drawbarCode*(ex1, path);

flag++;

}

// 动态生成物流单号条形码图片

**int** flag2 = 1;// 标记文件位置，动态生成不覆盖

String transPath = "src/TransPicture/";

**for** (**int** i = 0; i < explist.size(); i++) {

express ex1 = explist.get(i);

String path = transPath + String.*valueOf*(flag2) + ".jpg";

PictureService.*drawTransPicture*(ex1, path);

flag2++;

}

// 将所有快递图片上的要素画进去，生成快递图片

**int** flag3 = 1;// 标记文件位置，动态生成不覆盖

String emspath = "src/EMSpicture/";

**for** (**int** i = 0; i < explist.size(); i++) {

String BackPath = "src/BackdropPicture/back.jpg";

String plannarPath = "src/PlannarPicture/plannar.jpg";

String BarPath = barPath + String.*valueOf*(flag3) + ".jpg";

String TransPath = transPath + String.*valueOf*(flag3) + ".jpg";

String EMSPath = emspath + String.*valueOf*(flag3) + ".jpg";

express ex1 = explist.get(i);

PictureService.*drawEMSpicture*(ex1, BackPath, plannarPath, BarPath, TransPath, EMSPath);

flag3++;

}

// 动态将快递图片转换成pdf

String pdfpath = "src/PDF/";

**int** flag4 = 1;// 标记文件位置，动态生成不覆盖

**for** (**int** i = 0; i < explist.size(); i++) {

String pdfpath2 = pdfpath + String.*valueOf*(flag4) + ".pdf";

String picturepath = emspath + String.*valueOf*(flag4) + ".jpg";

PDFservice.*PicturetoPDF*(picturepath, pdfpath2);

flag4++;

}

System.***out***.println("一共生成" + explist.size() + "张图片，请前往SRC目录下EMSpicture目录查看");

System.***out***.println("一共生成" + explist.size() + "张PDF，请前往SRC目录下PDF目录查看");

**break**;

}

**default**:

System.***out***.println("您的输入有误请重新输入");

**break**;

}

*meanu*();

}

}

**public** **static** **void** meanu()// 菜单

{

System.***out***.println("欢迎使用快递图片生成系统");

System.***out***.println("请输入您的选择");

System.***out***.println("1.从excel文件读取快递信息");

System.***out***.println("2.显示所有快递信息");

System.***out***.println("3.生成快递图片并且转换成PDF");

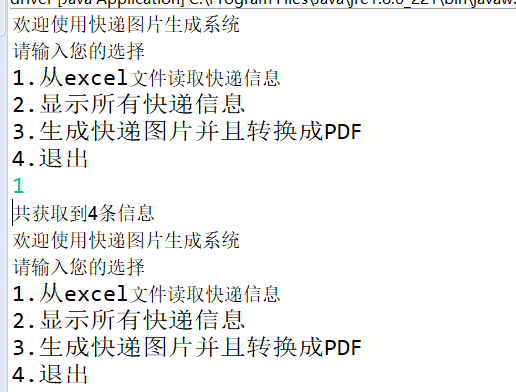
System.***out***.println("4.退出");

}

}

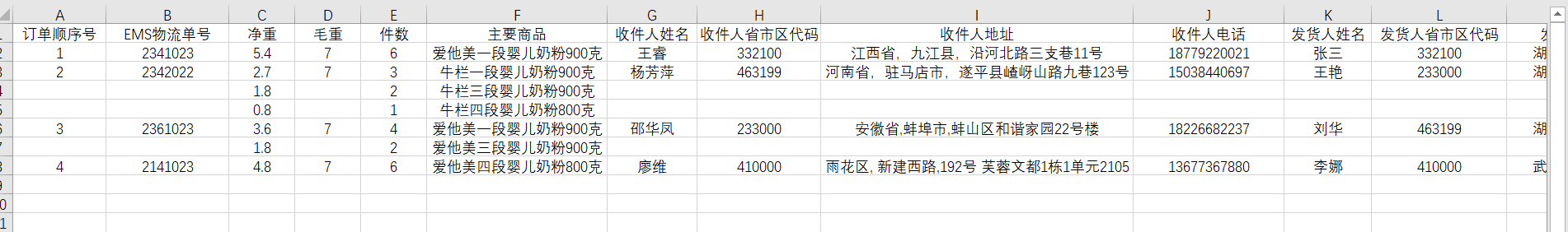
# 4.系统测试

## 4.1读取信息：

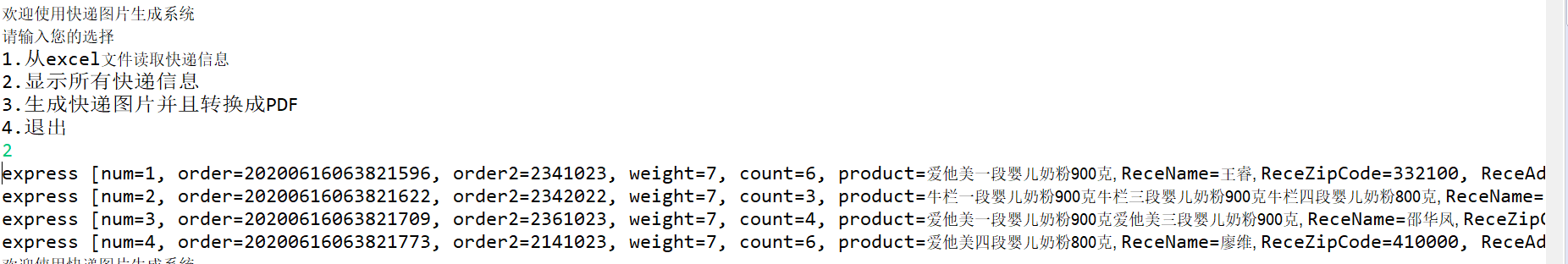


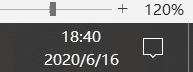
## 4.2显示快递信息：

Excel信息：

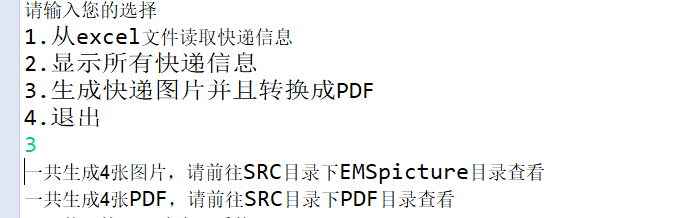


输出到控制台：

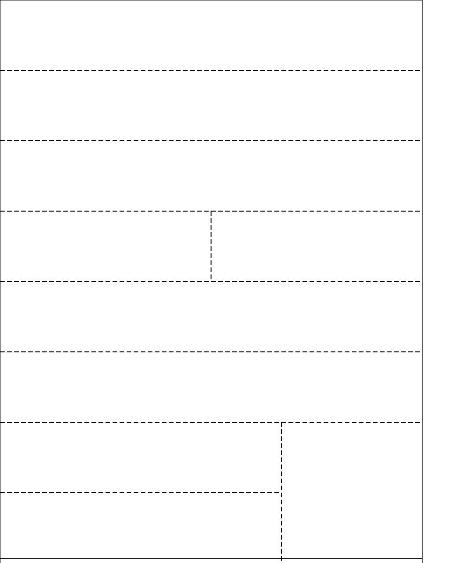


 可以看到订单号生成规则与系统时间吻合

## 4.3查看生成的图片：



背景框架图：

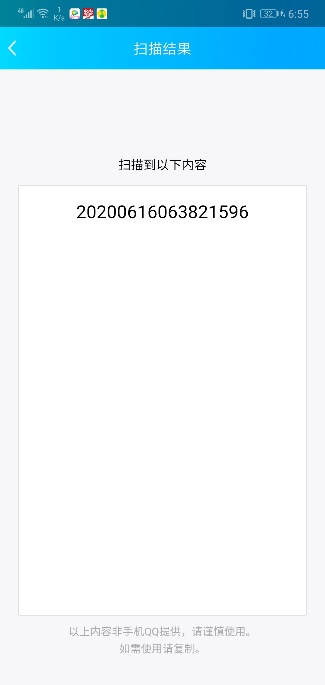


条形码图：

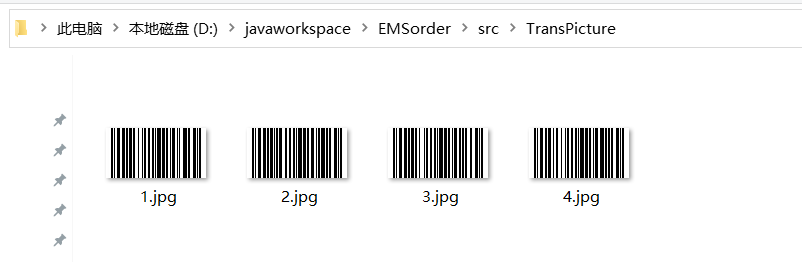


测试扫描结果发现吻合测试成功



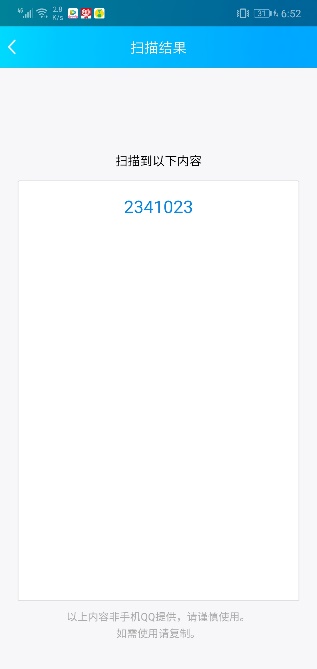


物流单号条形码：



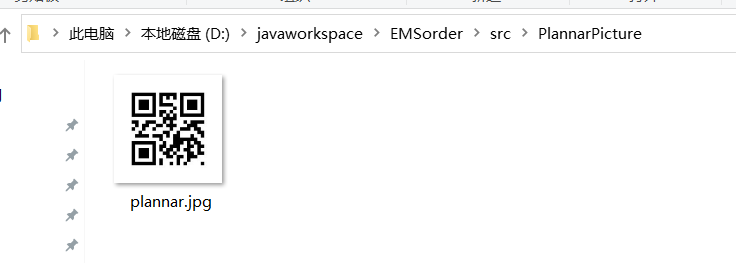
测试扫描结果：看到与物流单号吻合测试成功



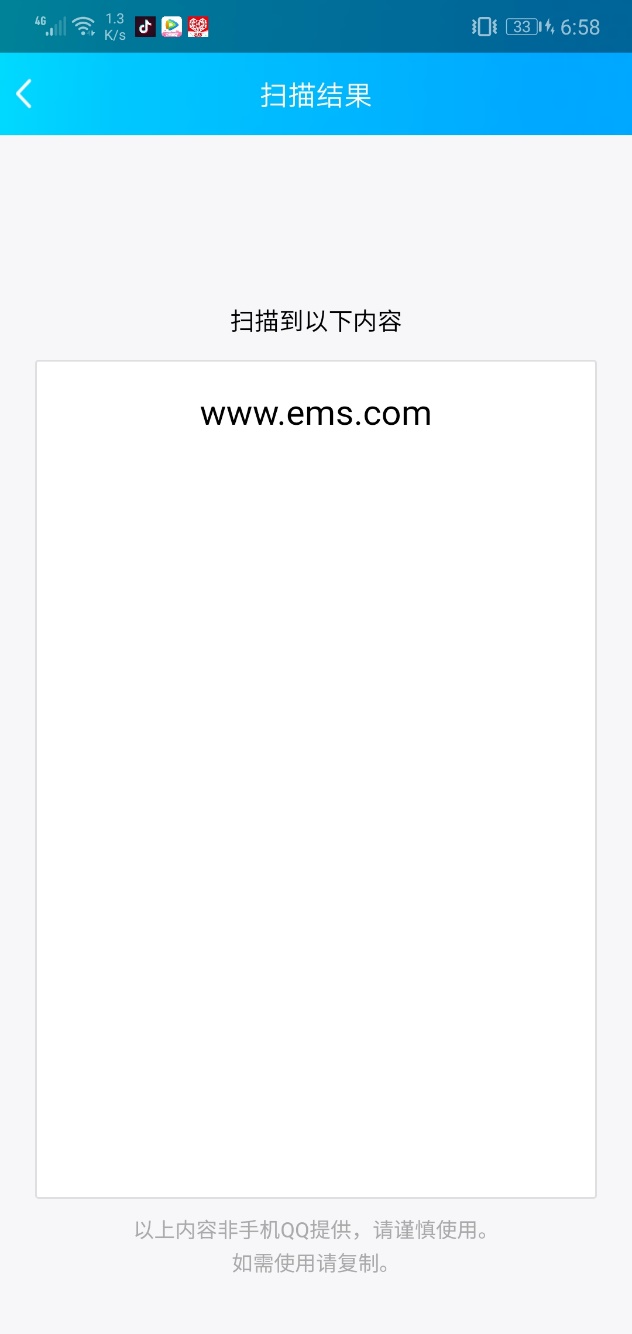
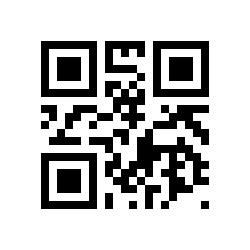




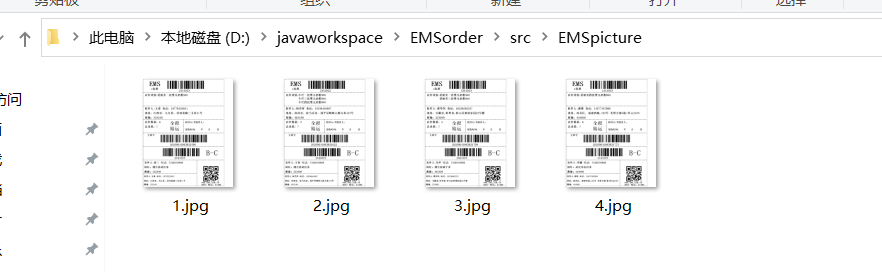
二维码图：



测试扫描结果：测试成功



生成的EMS快递图片：











生成的PDF文件：





经过上面测试发现系统功能完成成功

# 5.系统总结

此次实验内容是老师课上没讲过的问题，对于我来说最大的问题就是在有限的时间新学习新的内容，在这过程中也遇到了不少问题，比如图片生成时如何控制好大小以及图片上每一部分内容位置的规划字体设置等问题，设计之初遇到的问题也很多，如何生成条形码，二维码，以及如何转成pdf后来经过查阅相关api文档这些问题才慢慢解决。

系统中存在的不足，有一点就是我尝试过通过类似于java swing流布局布置，后来发现操作起来不熟悉，没弄好，导致可能存在如果内容过长不能自动转行排版，虽然我后来通过切割字符串避免了这个问题，但是感觉效果也不是很好。其次对于excel读取的方式因为这个excel文档格式不规律，我设计的时候读取的算法过于复杂，希望通过以后的学习能更加优化。

对于这学期的收获来说可以是巨大的，java语言我之前并未接触过，通过了一学期的学习我感觉写起来由以前的蹑手蹑脚，到现在得心应手。老师每次布置的练习我都有完成，老师与其他老师不同，布置的联系相较于别的老师虽然复杂，但更有针对性，和代表性。虽然我每次都花了很多时间去琢磨。但是通过这些联系我感觉自己的能力有了很大提升，也感觉充实。其次也学到了很多别的老师不讲的东西，比如DAO模式，以及Mybatis框架等等，因此收获很多

对于课程的建议就是希望以后能多多讲解以下平常的习题。

自己以后的发展方向时目前我想走的是java web方面发展，java web应用开发，也希望下学期能选到老师的这门课程。

# 6.致谢

**感谢老师这学期的幸苦教导**