《Java语言及网络编程》作业四

学号	姓名	班级
05191643	许万鹏	信息安全19-01班

1 第一题

1.1 题目

输入三个字符串，分别：

1. 必须满足密码复杂性要求（认证需求）
2. 满足身份证号码规范（15位/18位）
3. 满足电子邮件规范

1.2 答案

1.2.1 代码

```
package edu.wanpengxu.homework4.first;

import java.io.Console;
import java.util.Scanner;
import java.util.regex.Pattern;

public class Test {
    public static boolean isIdCard(String idCard) {
        String idCardCheck = "(^[1-8]\\d((0[1-9])|([1-6][0-9])|70|90)(0[1-9]|1[0-8]|2[1-9]|
[3-9][0-9])(19|20)\\d{2}((0[1-9])|(1[0-2]))(([0-2][1-9])|[1-3]0|31)\\d{3}[0-9Xx]$)|" + "
(^[1-8]\\d((0[1-9])|([1-6][0-9])|70|90)(0[1-9]|1[0-8]|2[1-9]|[3-9][0-9])\\d{2}((0[1-9])|
(1[0-2]))(([0-2][1-9])|[1-3]0|31)\\d{3}$)";
        // 检测空指针和空串
        if (idCard == null || "".equals(idCard)) return false;
        // 正则检查格式正确性
        if (!Pattern.compile(idCardCheck).matcher(idCard).matches()) return false;
        // 若为18位则对校验位进行校验
        if (idCard.length() == 18) {
            int sum = 0;
            for (int index = 0; index < 17; index++)
                sum += (Math.pow(2, 17 - index) % 11) * (idCard.charAt(index) - '0');
            char checkBit = (char) ('0' + ((12 - sum % 11) % 11));
            checkBit = checkBit == ('0' + 10) ? 'X' : checkBit;  // 最后一位若为阿拉伯数字10则
用罗马数字X表示
            return
Character.toString(checkBit).equalsIgnoreCase(String.valueOf(idCard.charAt(17)));
        }
        // 若为15位,不需校验
        else return true;
    }
```

```
    public static void main(String[] args) {
        System.out.println("欢迎来到邮箱注册程序！");
        try (Scanner scanner = new Scanner(System.in)) {
            // ^[\\w!#$%&'*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{|}~^-]+)*@(?:[a-zA-Z0-
9-]+\\.)+[a-zA-Z]{2,6}$
            String emailCheck = "^(?!.*\\.{2})[A-Za-z][0-9A-Za-z.]{4,13}[0-9A-Za-z](?!
(@\\.|@-|.*\\.-|.*\\.{2})|.*-\\.|.*\\-{2})@[0-9A-Za-z-]{1,63}(\\.[0-9A-Za-z-]{1,63})+$";
            String email;
            while (true) {
                System.out.println("请输入邮箱账号（6~18个字符，可使用字母、数字、下划线，需要以
字母开头）");
                email = scanner.nextLine();
                if (Pattern.compile(emailCheck).matcher(email).matches()) break;
                else System.out.println("请检查输入格式！");
            }

            String passwordCheck = "^(?![A-Za-z0-9]+$)(?![a-z0-9\\W]+$)(?![A-Za-z\\W]+$)(?!
[A-Z0-9\\W]+$)[a-zA-Z0-9\\W]{8,16}$";
            String password;
            while (true) {
                System.out.println("请输入密码（8~16个字符，需包含"大小写字母、数字、标点符号"中
3种或以上的组合）");
//                password = scanner.nextLine();
                Console con = System.console();
                password = new String(con.readPassword());  // 隐藏输入，在IDE中不可行，因为控
制台被重定向了
                if (Pattern.compile(passwordCheck).matcher(password).matches()) break;
                else System.out.println("请检查输入格式！");
            }

            String idCard;
            while (true) {
                System.out.println("请输入您的真实身份证号码");
                idCard = scanner.nextLine();
                if (isIdCard(idCard)) break;
                else System.out.println("请检查输入格式！");
            }
            System.out.println("注册成功！");
        }
    }
}
```

1.2.2 运行截图

```
D:\Codefield\CODE_Java\IdeaProjects\src
λ java edu.wanpengxu.homework4.first.Test
欢迎来到邮箱注册程序！
请输入邮箱账号（6~18个字符，可使用字母、数字、下划线，需要以字母开头）
xwp05191643.cn@cu-mt.edu.cn
请输入密码（8~16个字符，需包含“大小写字母、数字、标点符号”中3种或以上的组合）

请输入您的真实身份证号码
23[redacted]491
注册成功！
```

1.3 分析

构建正则表达式就和写程序一样，要由编译器运行起来才能直观地看到结果，这里可以使用正则表达式在线测试| 菜鸟工具

1.3.1 电子邮件匹配

首先，要对电子邮件进行匹配，就需要了解电子邮件规范RFC 5322

1.3.1.1 RFC 5322

电子邮件地址的域内部分可以使用以下任何ASCII字符：

- 大小写拉丁字母`A`到`Z`和`a`到`z`；
- 数字`0`到`9`；
- 除了字母与数字之外的可打印字符，`!#$%&'*+-/=?^_`{|}~`；
- 点`.`，但不能作为首尾字符，也不能连续出现，若放在引号中则不受这些限制（例如`John..Doe@example.com`是不允许的，而`"John..Doe"@example.com`是允许的）。
- 空格和特殊字符`"(),:;<>@[\]`被允许有限制地使用（域内部分字符串必须放在引号中，后面的段落将会描述，并且，反斜杠或双引号之前，必须加一个反斜杠来转义）；
- 允许将注释放在小括号内，并放在域内部分的开头或结尾，例如`john.smith(comment)@example.com`和`(comment)john.smith@example.com`都等同于`john.smith@example.com`。

电子邮件地址的域名部分必须符合严格的规则：它必须满足对主机名的要求，一个以点分隔的DNS标签序列，每个标签被限定为长度不超过63个字符，且只能由下列字符组成：

- 大小写拉丁字母`A`到`Z`和`a`到`z`；
- 数字`0`到`9`，但顶级域名不能是纯数字；
- 连字符`-`，但不能作为首尾字符。

可以看出，RFC 5322电子邮件规范包含范围广泛的特殊字符，虽然在技术上可行，但在实践中往往并不能接受所有这些字符。

所以我们需要调研知名邮箱的电子邮件规范，我在这里选择了国内最流行的163邮箱和国际最流行的gmail邮箱，他们在格式错误时都会回显原因。

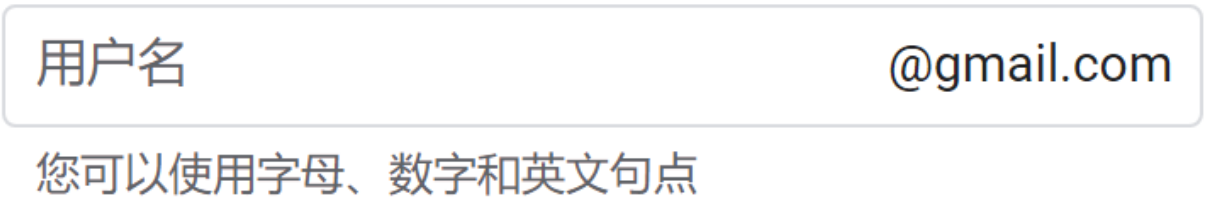
1.3.1.2 域内部分

网易 **163**邮箱

163给出的邮箱规范很具体，我们可以自行测试它的细则。

1. 邮箱地址需以字母或数字结尾

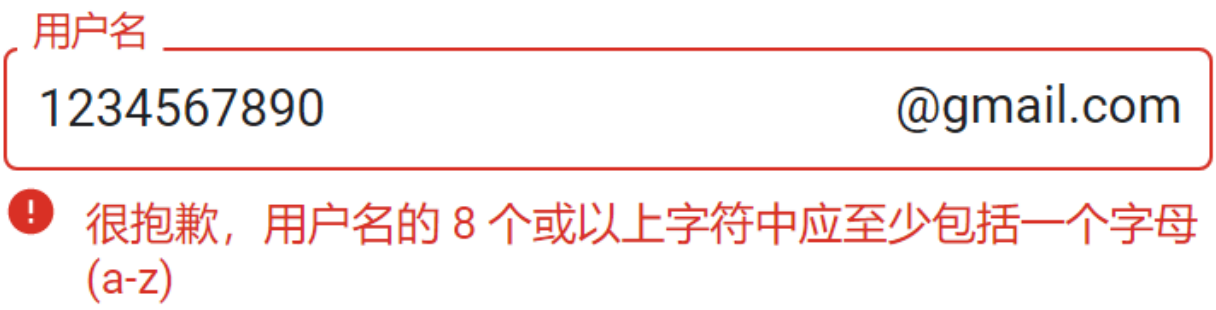
Google gmail邮箱



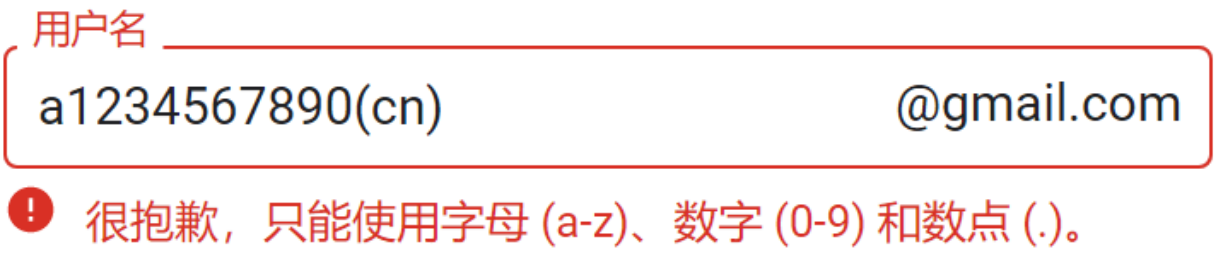
gmail给出的邮箱规范很简洁，我们也可以自行测试它的细则。

1. 用户名的字符数须介于6到30之间。

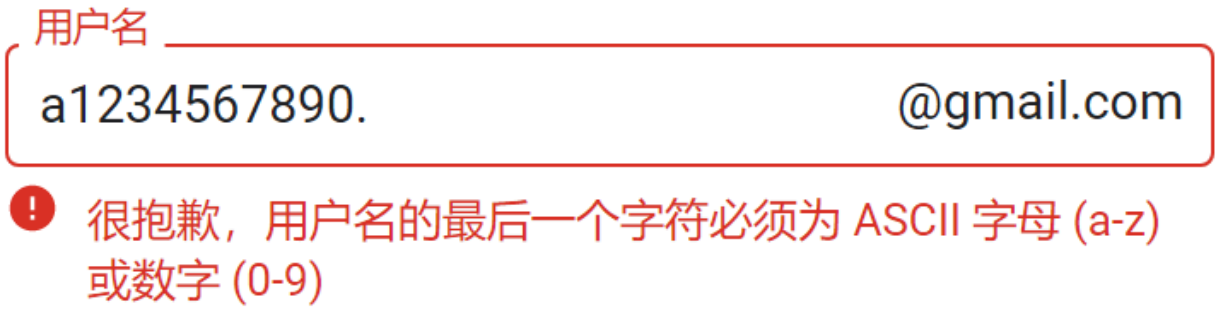
2. 用户名的8个或以上字符中应至少包括一个字母（a-z）



3. 只能使用字母（a-z）、数字（0-9）和数点（.）



3. 用户名的最后一个字符必须为ASCII字母（a-z）或数字（0-9）



4. 用户名中不能包含连续的数点（.）

我还对Microsoft outlook邮箱的电子邮件规范进行了调研，个人认为outlook邮箱的格式太过宽泛（比如单个字母可以作为用户名，且用户名最长可达65位），实现出来并不利于使用。

对以上规范进行总结，得到实践中的电子邮件规范：

1. 用户名的字符数介于一定范围
2. 可用字符为大小写字母（`a-z`和`A-Z`）、数字（`0-9`）、一些特殊符号（`-_.`）
3. 用户名至少包含一个字母（有些邮箱要求其在开头，有些邮箱在小于一定长度时不要求）
4. 用户名的结束字符为字母或数字（有些邮箱允许`_-`）

那么我们可以制定一个属于自己的、简易的邮箱规范：

1. 用户名的字符数须介于**6**到**15**之间；
2. 用户名可使用字母（`a-z`和`A-Z`）、数字（`0-9`）、数点（`.`）；
3. 用户名需要以字母（`a-z`和`A-Z`）开头；
4. 用户名中不能包含连续的数点（`.`）；
5. 用户名需要以字母或数字（`a-z`和`A-Z`和`0-9`）结尾。

开始构建表达式吧！

- `^`正则表达式开始符
- `(?!.*\.{2})`不允许两个以上的句点（环视技术，详见1.3.2）
- `[A-Za-z]`开头部分的拉丁字母

- `[0-9A-Za-z.]{4,13}`中间部分的允许字符
- `[0-9A-Za-z]`结尾部分的拉丁字母或数字
- `$`正则表达式结束符

最后可得

```
"^(?!.*\.{2})[A-Za-z][0-9A-Za-z.]{4,13}[0-9A-Za-z]$"
```

当然我也搜索学习了RFC 5322电子邮件规范的正则匹配表达式

```
"^[\w!#$%&'*+/=?`{|}~^-]+(?:\.[\\w!#$%&'*+/=?`{|}~^-]+)*$"
```

1.3.1.3 域名部分

RFC 5322制定的电子邮件规范在这里需求较少，直接构建正则表达式即可。

- `^`正则表达式开始符
- `(?!(@\.|@-|.*\.-|.*\.{2})|.*-\.|.*\-{2})` `@`后不能接`.`，`@`后不能接`-`，`.`后不能接`-`，`.`后不能接`.`，`-`后不能接`.`，`-`后不能接`-`
- `@[0-9A-Za-z-]{1,63}`第一段DNS标签（@开头），出现一次
- `(\.[0-9A-Za-z-]{1,63})+`第二段DNS标签及其之后的DNS标签（.开头），出现至少一次（用`+`匹配）
- `$`正则表达式结束符

```
"^(?!(@\.|@-|.*\.-|.*\.{2})|.*-\.|.*\-{2})@[0-9A-Za-z-]{1,63}(\.[0-9A-Za-z-]{1,63})+$"
```

综上，得到最终电子邮件匹配正则表达式

```
"^(?!.*\.{2})[A-Za-z][0-9A-Za-z.]{4,13}[0-9A-Za-z](?!(@\.|@-|.*\.-|.*\.{2})|.*-\.|.*\-{2})@[0-9A-Za-z-]{1,63}(\.[0-9A-Za-z-]{1,63})+$"
```

经测试，可以实现上述的所有功能，放一张成功匹配的截图。

测试工具

修饰符: g

```
^(?!.*\.{2})[A-Za-z][0-9A-Za-z.]{4,13}[0-9A-Za-z](?!(@\.|@-|.*\.-|.*\.{2})|.*-\.|.*\-{2})@[0-9A-Za-z-]{1,63}(\.[0-9A-Za-z-]{1,63})+$
```

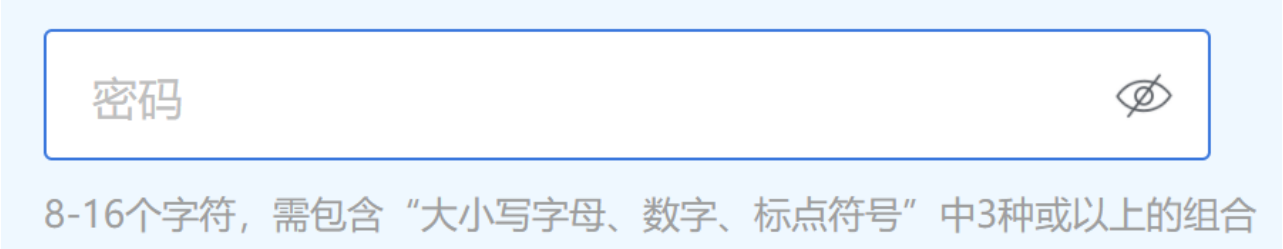
匹配文本:

```
xwp05191643.cn@cu-mt.edu.cn
```

1.3.2 密码匹配

同样地，首先我们需要制定一个密码复杂性要求。

对于密码，或者说口令，并没有严格的规范，我们先看一下163邮箱的密码复杂性要求。



我们可以在此基础上制定一个更严格的复杂性要求：

1. **8~16**个字符
2. 需包含**“**大写字母、小写字母、数字、标点符号**”**中**4**种或以上的组合

因为只对字符集进行了规定，所以很好构建正则表达式。

我们先来了解一下环视

环视（Lookaround）：匹配一个位置而不是字符，因此匹配的结果也可称为零宽字符。具体包括

1. Lookahead
2. Lookbehind

Lookahead，中文译作**“**正向预查**”**，其实字面意思就是向前看，也就是向字符串的右边看。

Lookbehind，中文译作**“**负向预查**”**，其实字面意思就是向后看，也就是向字符串的左边看。

接下来介绍一下环视所使用的符号

符号	意义
(	环视起始符
?	零宽字符，也就是位置，后续的操作就是选出合适的位置
<	Lookbehind，向零宽字符的左边看
（啥也不加，缺省）	Lookahead，向零宽字符的右边看
=	肯定形式，选出满足（有）其后pattern的零宽字符
!	否定形式，选出不满足（没有）其后pattern的零宽字符
$	字符串结尾
)	环视结束符

在这里直接对我的正则表达式进行说明即可理解环视。

```
"^(?![A-Za-z0-9]+$)(?![a-z0-9\\W]+$)(?![A-Za-z\\W]+$)(?![A-Z0-9\\W]+$)[a-zA-Z0-9\\W]{8,16}$"
```

首先我们知道四种字符，其共有$\sum_{i=1}^{4} C_4^i = 4+6+4+1 = 15$种情况，接下来要对除了$C_4^4$这种全选情况外的情况进行排除。

- `^`正则表达式开始符

- `(?![0-9A-Za-z]+$)`对整个字符串选出零宽字符后没有数字、大写字母、小写字母三者组合（$\sum_{i=1}^{3} C_3^i$）的零宽字符
- `(?![0-9a-z\\W]+$)`对整个字符串选出零宽字符后没有数字、小写字母、标点符号三者组合（$\sum_{i=1}^{3} C_3^i$）的零宽字符
- `(?![A-Za-z\\W]+$)`对整个字符串选出零宽字符后没有大写字母、小写字母、标点符号三者组合（$\sum_{i=1}^{3} C_3^i$）的零宽字符
- `(?![0-9A-Z\\W]+$)`对整个字符串选出零宽字符后没有数字、大写字母、标点符号三者组合（$\sum_{i=1}^{3} C_3^i$）的零宽字符
- `[0-9A-Za-z\\W]{8,16}`从选出的零宽字符起，选中含有8~16个数字、大写字母、小写字母、标点符号的字符串。进行到这步时，因为前面几种环视已经排除掉了$\sum_{i=1}^{3} C_4^i = 14$种情况，所以选中的字符串必是四种符号的全选情况。
- `$`正则表达式结束符

这其中`\W`代表匹配任何非单词字符，当然也可以自己指定允许的标点符号集合（注意这个转义字符包含`_`）。

下两张图可区分选中零宽字符和选中字符串。

测试工具

修饰符：g

```
(?![A-Za-z0-9]+$)(?![a-z0-9\W]+$)(?![A-Za-z\W]+$)(?![A-Z0-9\W]+$)
```

匹配文本：

```
xX1234@567890
```

测试工具

修饰符：g

```
(?![A-Za-z0-9]+$)(?![a-z0-9\W]+$)(?![A-Za-z\W]+$)(?![A-Z0-9\W]+$)[a-zA-Z0-9\W]{8,16}
```

匹配文本：

```
xX1234@567890
```

1.3.3 身份证号码匹配

1.3.3.1 格式匹配

对百位数以下的数字匹配时可以想象出一个每行10个数的表格（个位数0-9），中括号匹配时相当于用矩形选中，每次可能框最多的元素

十八位身份证正则表达式说明：

- `^` 正则表达式开头
- `[1-8]` 大区制代码，全国共分1-8八个大区
- `\\d` 省市编码
- `( (0[1-9]) | ([1-6][0-9]) | 70 | 90 )` 地级行政区：01-70一般 90直辖
- `( 0[1-9] | 1[0-8] | 2[1-9] |[3-9][0-9] )` 县级行政区：01-18、21-99
- `( 19|20 )` 生日期码，年份前两位
- `\\d{2}` 生日期码，年份后两位
- `( (0[1-9]) | (1[0-2]) )` 生日期码，月份
- `( ([0-2][1-9]) | [1-3]0 |31 )` 生日期码，日期
- `\\d{3}` 顺序码
- `[0-9Xx]` 校验码,这里只判断格式而不校验
- `$` 正则表达式结尾

```
"^[1-8]\\d((0[1-9])|([1-6][0-9])|70|90)(0[1-9]|1[0-8]|2[1-9]|[3-9][0-9])(19|20)\\d{2}((0[1-9])|(1[0-2]))(([0-2][1-9])|[1-3]0|31)\\d{3}[0-9Xx]$"
```

十五位身份证正则表达式说明：
只是在十八位身份证的基础上去掉了 生日期码，年份前两位 和 校验码。

```
"^[1-8]\\d((0[1-9])|([1-6][0-9])|70|90)(0[1-9]|1[0-8]|2[1-9]|[3-9][0-9])       \\d{2}((0[1-9])|(1[0-2]))(([0-2][1-9])|[1-3]0|31)\\d{3}       $"
```

1.3.3.2 校验码检查

首先了解一下校验码生成算法

1. 将身份证号码从左至右标记为$a_1, a_2, \cdots, a_{18}$；a_{18}即为校验码；
2. 计算权重系数$W_i = 2^{18-i} \bmod 11$；其中 mod 表示求余数。

所以：

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Wi	7	9	10	5	8	4	2	1	6	3	7	9	10	5	8	4	2	1

3. 计算$S = \sum_{i=1}^{17} a_i \cdot W_i$
4. $a_{18} = (12 - (S \bmod 11)) \bmod 11$

那么根据算法，很容易写出程序。

```
int sum = 0;
for (int index = 0; index < 17; index++)
    sum += (Math.pow(2, 17 - index) % 11) * (idCard.charAt(index) - '0');
char checkBit = (char) ('0' + ((12 - sum % 11) % 11));
checkBit = checkBit == ('0' + 10) ? 'X' : checkBit;  // 最后一位若为阿拉伯数字10则用罗马数字X表示
```

```
 D:\Codefield\CODE_Java\IdeaProjects\src
λ java edu.wanpengxu.homework4.first.Test
欢迎来到邮箱注册程序！
请输入邮箱账号（6~18个字符，可使用字母、数字、下划线，需要以字母开头）
xwp05191643.cn@cu-mt.edu.cn
请输入密码（8~16个字符，需包含“大小写字母、数字、标点符号”中3种或以上的组合）

请输入您的真实身份证号码
230882200105314 91X
注册成功！
```

2 第二题

2.1 题目

利用鼠标事件启动3个线程分别在三个窗口中同时绘制动态图形（图形自选）。

2.2 答案

2.2.1 代码

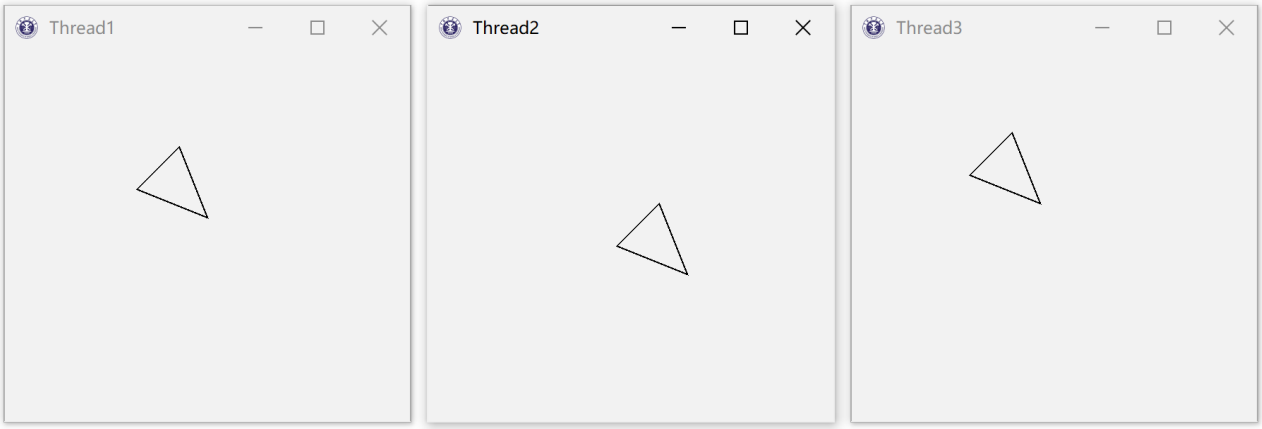
```
package edu.wanpengxu.homework4.second;


import com.formdev.flatlaf.FlatLightLaf;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.net.URL;

public class TestFrame extends JFrame implements Runnable {
    int i = 0;

    public TestFrame() {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                startThread();
            }
        });
    }

    private void startThread() {
        Thread t1 = new Thread(this);
        t1.start();
    }

    public void run() {
```

```
        for (i = 1; ; i = (i + 1) % 10) {
            try {
                repaint();
                Thread.sleep(500);
            } catch (InterruptedException e) {
                JOptionPane.showMessageDialog(null, e.getMessage(), "Exceptions",
JOptionPane.WARNING_MESSAGE);
            }
        }
    }

    public void paint(Graphics g) {
        super.paint(g);
        int[] x = {80 + i * 10, 100 + i * 10, 50 + i * 10};
        int[] y = {50 + i * 10, 100 + i * 10, 80 + i * 10};
        if (i != 0) {
            g.drawPolygon(x, y, 3);
//            g.fillOval(50 + i * 10, 50 + i * 10, 100, 100);
        }
    }

    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(new FlatLightLaf());
        } catch (Exception ex) {
            System.err.println("Failed to initialize LaF");
        }
        URL imgURL = TestFrame.class.getResource("CUMTlogo.png");
        ImageIcon imageIcon = new ImageIcon(imgURL);
        Image image = imageIcon.getImage();

        TestFrame testFrame1 = new TestFrame();
        testFrame1.setBounds(100, 200, 300, 300);
        testFrame1.setIconImage(image);
        testFrame1.setTitle("Thread1");
        testFrame1.setVisible(true);

        TestFrame testFrame2 = new TestFrame();
        testFrame2.setBounds(400, 200, 300, 300);
        testFrame2.setIconImage(image);
        testFrame2.setTitle("Thread2");
        testFrame2.setVisible(true);

        TestFrame testFrame3 = new TestFrame();
        testFrame3.setBounds(700, 200, 300, 300);
        testFrame3.setIconImage(image);
        testFrame3.setTitle("Thread3");
        testFrame3.setVisible(true);
```

```
        }
    }
```

2.2.2 运行截图

三个Frame启动三个Thread，图形绘制时刻由鼠标点击时刻决定。



2.3 分析

与课上例题没什么区别，例题创建了一个Frame，每个Frame点击后可启动Thread。那完成基本的要求只需要在主函数里再声明两个Frame即可。

为增强可读性，我将图形绘制改为了循环绘制，另外将例题中原程序的实心圆改为了指向右下方的三角形，更改了主题、自定义了图标、增加了异常弹窗等，都是上次GUI作业中使用过的方法。