



# 燕山大学

## Java Web 开发技术实验报告

### Java Web Development Technology Experiment Report

学生所在学院：软件学院

学生所在班级：18 软件 6 班

学生姓名：乔翱

学 号：201811040809

指导教师：郝晓冰 李可 刘佳新 梁顺攀

教 务 处

2020 年 3 月

# 目 录

<b>实验 1 JAVA 语言基础知识.....</b>	<b>4</b>
1.1 实验原理和要求.....	4
1.2 实验任务.....	4
1.2.1 任务一.....	4
1.2.2 任务二.....	6
1.2.3 任务三.....	7
1.2.4 任务四.....	9
<b>实验 2 JAVA 类的定义与使用.....</b>	<b>12</b>
2.1 实验原理和要求.....	12
2.2 实验任务.....	12
2.2.1 任务一.....	12
2.2.2 任务二.....	14
2.2.3 任务三.....	16
<b>实验 3 核心类的使用.....</b>	<b>21</b>
3.1 实验原理和要求.....	21
3.2 实验任务.....	21
3.2.1 任务一.....	21
3.2.2 任务二.....	24
3.2.3 任务三.....	26
3.2.4 任务四.....	28
<b>实验 4 类继承的使用.....</b>	<b>31</b>
4.1 实验原理和要求.....	31
4.2 实验任务.....	31
4.2.1 任务一.....	31
4.2.2 任务一.....	36
4.2.3 任务三.....	43
<b>实验 5 JAVA 语言之接口.....</b>	<b>54</b>
5.1 实验原理和要求.....	54
5.2 实验任务.....	54
5.2.1 任务一.....	54
5.2.2 任务二.....	56
<b>实验 6 JAVA 中的异常处理.....</b>	<b>60</b>
6.1 实验原理和要求.....	60
6.2 实验任务.....	60
6.2.1 任务一.....	60
6.2.2 任务二.....	62
6.2.3 任务三.....	63

<b>实验 7 基于 HTML、CSS、JAVASCRIPT 网页设计.....</b>	<b>66</b>
7.1 实验原理和要求.....	66
7.2 实验任务.....	66
7.2.1 任务一.....	66
<b>实验 8 JSP 技术的基础应用以及程序设计.....</b>	<b>77</b>
8.1 实验原理和要求.....	77
8.2 实验任务.....	77
8.2.1 任务一.....	77
<b>实验 9 基于 SERVLET 的程序设计.....</b>	<b>87</b>
9.1 实验原理和要求.....	87
9.2 实验任务.....	87
9.2.1 任务一.....	87
<b>实验 10 JAVABEAN 程序.....</b>	<b>95</b>
10.1 实验原理和要求.....	95
10.2 实验任务.....	95
10.2.1 任务一.....	95
<b>实验 11 标签与自定义标签.....</b>	<b>103</b>
11.1 实验原理和要求.....	103
11.2 实验任务.....	103
11.2.1 任务一.....	103
<b>实验 12 JDBC 数据库连接技术及其程序设计.....</b>	<b>111</b>
12.1 实验原理和要求.....	111
12.2 实验任务.....	111
12.2.1 任务一.....	111

# 实验 1 Java 语言基础知识

## 1.1 实验原理和要求

了解 Java 语言基础知识，了解 Java 语言的基本运行方式，以及学会简单的输入和输出。了解 Java 语言的变量、常量、数据类型、流程控制、数组等基本语言基础。

## 1.2 实验任务

### 1.2.1 任务一

#### (1) 任务内容

编写一个要求用户输入姓名以及两个整数，并输出姓名和两数之和。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

接受用户的输入，接受输入的姓名和两个整数，并且输出姓名和两数之和。

#### (3) 程序源码及说明

```
package test;

import java.util.Scanner;

public class First {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String username="";

        int a=0,b=0;

        //用户输入提示

        System.out.println("输入用户名: ");

        //获取第一个输入字符串

        if (scan.hasNext()) {
```

```

        username = scan.next();

    }

    //用户输入提示

    System.out.println("输入第一个整数: ");

    //获取输入的整数

    if (scan.hasNextInt()) {

        a = scan.nextInt();

    }

    //用户输入提示

    System.out.println("输入第二个整数: ");

    //获取输入的整数

    if (scan.hasNextInt()) {

        b = scan.nextInt();

    }

//下边编写输出用户名和两数之和的程序

    System.out.println("用户名: "+username);

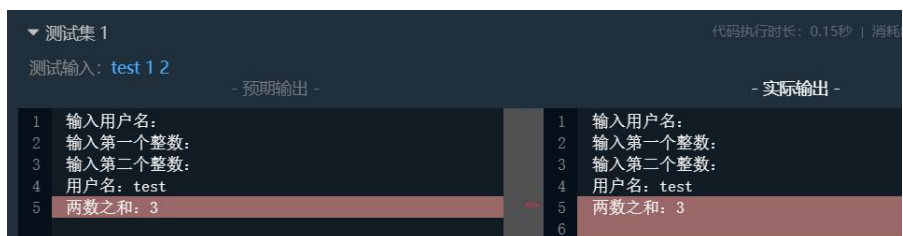
    System.out.println("两数之和: "+(a+b));

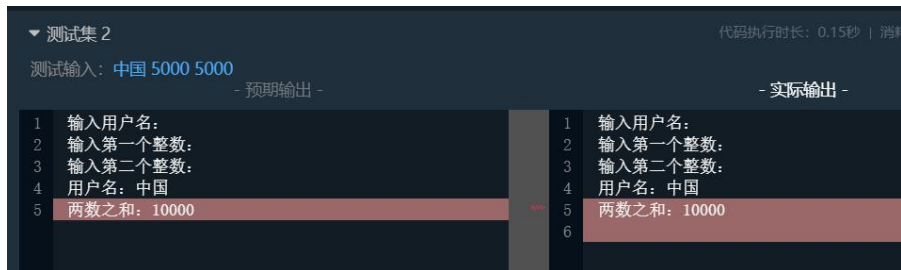
}

}

```

#### (4) 运行结果(截图)





### (5) 心得体会（问题调试及体会）

了解了 Java 语言的基本结构以及如何处理数据，学会了怎么输入和输出。

## 1.2.2 任务二

### (1) 任务内容

使用 for 循环输出用 “\*” 表达的等腰三角。

### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

寻找规律，找到每一层的空格与\*的数量与每一层的关系，利用 for 循环来输出等腰三角形。

### (3) 程序源码及说明

```

package first;

import java.util.Scanner;

public class Printsanjiao {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int i = 0;

        //获取用户输入存入整形变量 i 中

        i=scan.nextInt();

        for(int x=1;x<=i;x++){

            for(int j=1;j<=i-x;j++)

                System.out.print(" ");

```

```

        for(int k=1;k<=(2*x-1);k++)

            System.out.print("*");

            System.out.println();

        }

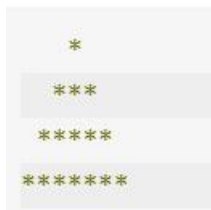
//用 for 输出等腰三角形

    }

}

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

学会并且掌握了 Java 中的循环结构。

### 1.2.3 任务三

#### (1) 任务内容

编写一个根据用户输入的整数，打印输出菱形图案。 注意：需要判断输入的整数（行数）是否能构成菱形，如果非法，则输出 error。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先先判断输入的整数是否能构成菱形，只有输入的整数是奇数并且大于 1 才能构成菱形，不能构成菱形就输出“error”。

如果输入合理，在输出菱形时，利用 for 循环结构先输出上半部分，再输出下半部分。

#### (3) 程序源码及说明

```

package sec;

import java.util.Scanner;

public class Printlingxing {

```

```
public static void main(String[] args) {  
    Scanner scan = new Scanner(System.in);  
    int i = 0;  
    //获取用户输入存入整形变量 i 中  
    i=scan.nextInt();  
    if(i%2==0||i<3){  
        System.out.println("error");  
    }  
    else  
    {  
        int size=i/2+1;  
        for(int x=1;x<=size;x++){  
            for(int y=1;y<=size-x;y++)  
                System.out.print(" ");  
            for(int j=1;j<=2*x-1;j++)  
                System.out.print("*");  
            System.out.println();  
        }  
        for(int x=1;x<=size-1;x++){  
            for(int y=1;y<=x;y++)  
                System.out.print(" ");  
            for(int j=2*size-3;j>=2*x-1;j--)  
                System.out.print("*");  
            System.out.println();  
        }  
    }  
}
```



```
        }

    }

    //判断并输出

    }

}
```

(4) 运行结果(截图)



(5) 心得体会（问题调试及体会）

学会并且掌握了 Java 中循环结构，for 循环的使用，学会处理简单的问题。

1.2.4 任务四

(1) 任务内容

编写一个能接收用户输入的整形数据并存放放到整形数组内，计算数组的和、平均值和最大值并输出的小程序。

(2) 构思过程(可用文字、流程图、UML 图等方式表达)

声明一个整形数组，接受用户的输入的整形数据，把输入的数据存入到整形数组内，遍历数组计算数据的和，并且求出平均值、最大值，注意输出结果保留一位小数。

(3) 程序源码及说明

```
package third;
```

```

import java.util.Scanner;

public class Shuzujisuan {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        //获取用户输入，注意第一个数字为接下来输入数字的数量，定义整形数组，获取用户输入存入数组内
        int size=scan.nextInt();
        int []a=new int [size];
        int sum=0;
        int max=0;
        for(int i=0;i<size;i++)
        {
            a[i]=scan.nextInt();
            if(a[i]>max)
                max=a[i];
            sum+=a[i];
        }
        //计算数组的和、平均值、最大值并输出
        double aver=(double)sum/(double)size;
        System.out.println("和:"+sum);
        System.out.println("平均值:"+String.format("%.1f",
aver));
        System.out.println("最大值:"+max);
    }
}

```

#### (4) 运行结果(截图)



▼ 测试集 2

代码执行时长: 0.14秒 | 消耗内存45.5MB

测试输入: 2 1 2

- 预期输出 -

1	和:3
2	平均值:1.5
3	最大值:2

- 实际输出 -

1	和:3
2	平均值:1.5
3	最大值:2
4	

### （5）心得体会（问题调试及体会）

了解并且掌握了 Java 中数组的使用，包括数组的声明、获取数组的长度、遍历数组等一系列对数组的操作，数组的 `length` 属性用于记录数组中有多少个元素或存储单元，即记录数组的长度是多少。而遍历数组就是把数组中的所有元素都看一遍。掌握了 Java 中保留小数的方法（`String.format("%.1f", aver)`）。

## 实验 2 Java 类的定义与使用

### 2.1 实验原理和要求

类是 Java 语言的一个重要的特点，通过本次实验了解如何在 Java 中编写类以及如何使用类。理解并且学会类的定义、对象的创建和使用、方法的使用、封装和访问控制。

### 2.2 实验任务

#### 2.2.1 任务一

##### (1) 任务内容

编写一个 Book 类，有 name 和 pages 两个属性。编写一个 BookDemo 类，实例化一个 Book 对象并显示。

##### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

声明 Book 类，声明 name 和 pages 属性，声明含有两个参数的构造方法，并且编写测试类，实例化一个 Book 对象，并且显示其中的信息。

##### (3) 程序源码及说明

```
package step1;

public class Book{

    String name;

    int pages;

    //补充构造函数

    public Book(String name,int pages){

        // 这个构造器有两个参数: name, pages

        this.name=name;

        this.pages=pages;

    }

}
```

```
//补充输出 Book 对象的名称和页数的方法

    void Bookinfo() {

        System.out.println("书名:"+this.name);

        System.out.println("页数:"+this.pages);

    }

}

package step1;

import java.util.Scanner;

public class BookDemo{

    public static void main(String[] args){

        /* 填写代码，接收用户输入的书名和页数，实例化 Book 类对象
        myBook，并显示 myBook 的信息 */

        Scanner sc=new Scanner(System.in);

        String name="";

        int pages;

        name=sc.next();

        pages=sc.nextInt();

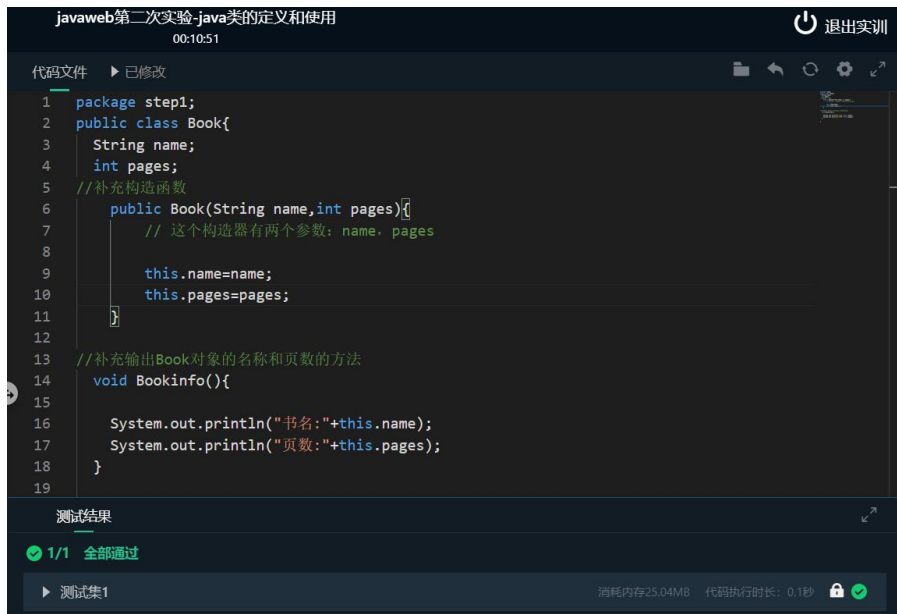
        Book b=new Book(name,pages);

        b.Bookinfo();

    }

}
```

#### (4) 运行结果(截图)



```
javaweb第二次实验-java类的定义和使用
00:10:51
退出实训

代码文件 ▶ 已修改

1 package step1;
2 public class Book{
3     String name;
4     int pages;
5     //补充构造函数
6     public Book(String name,int pages){
7         // 这个构造器有两个参数: name, pages
8
9         this.name=name;
10        this.pages=pages;
11    }
12
13    //补充输出Book对象的名称和页数的方法
14    void Bookinfo(){
15
16        System.out.println("书名:"+this.name);
17        System.out.println("页数:"+this.pages);
18    }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

测试结果
1/1 全部通过
▶ 测试集1 消耗内存25.04MB 代码执行时长: 0.1秒
```

#### (5) 心得体会（问题调试及体会）

通过本次任务学会了如何在 Java 中编写类，并且利用类解决实际问题，了解了 Java 中的封装以及面向对象的思想。

### 2.2.2 任务二

#### (1) 任务内容

编写一个 Point 类，有 x、y 两个属性。编写一个 PointDemo 类，并提供一个 distance (Point p1, Point p2) 方法用于计算两点之间的距离，实例化两个具体的 Point 对象并显示他们之间的距离。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

编写 point 类，并且声明两个成员变量，再编写 pointdemo 类来测试 point 类，声明一个 distance 方法计算两点距离

#### (3) 程序源码及说明

```
package step2;

public class Point {

    //完成类定义

    double x;
```

```
double y;

Point(double x,double y){

    this.x=x;

    this.y=y;

}

}

package step2;

import java.util.Scanner;

public class PointDemo {

    public static void main(String[] args){

        Scanner scan = new Scanner(System.in);

        //接收用户输入

        double x1,x2,y1,y2;

        x1=scan.nextDouble();

        y1=scan.nextDouble();

        x2=scan.nextDouble();

        y2=scan.nextDouble();

        //实例化 p1, p2

        Point p1=new Point(x1,y1);

        Point p2=new Point(x2,y2);

        distance(p1,p2);

        //用 distance (Point p1, Point p2) 方法输出距离, 保留一位小数

    }

}
```

```

public static void distance(Point pp1,Point pp2){

    double

dis=Math.sqrt((pp1.x-pp2.x)*(pp1.x-pp2.x)+(pp1.y-pp2.y)*(pp1.y-pp2.y));

    System.out.println(" 两点之间的距离为:"+String.format("%.1f",
dis));

}

}

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

通过本次实验熟练掌握了 Java 中类的使用方法，能够快速编写类并且利用类解决实际问题，深刻理解了 Java 中的封装思想以及面向对象的思想。

### 2.2.3 任务三

#### (1) 任务内容

编写一个用户类（Sysuser），属性包括用户名、真实姓名、年龄、出生日期、密码，类方法中包含单独修改用户年龄、判断用户名和密码、显示用户信息功能，在用户测试类中（TestSysuser），根据用户输入的数据初始化用户类，显示菜单：1. 修改用户年龄 2. 判断用户密码 3. 显示用户信息 4. 退出 根据输入的数字进入相关功能。例如输入 2 后，系统提示

1. 请输入用户名：
2. 请输入密码：

根据用户输入的用户名和密码判断是否正确，如果正确输出：登录成功，否则提示：用户名或者密码错误，修改年龄提示为：请输入年龄：。



显示用户信息模板:

1. 用户名:\*\*\*
2. 真实姓名:\*\*\*
3. 年龄:\*\*\*
4. 出生日期:\*\*\*
5. 密码:\*\*\*

每运行完一个功能，重新显示菜单，直到用户输入 4 退出。

## (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先编写用户类，并且为其声明成员变量和方法，接着编写测试类，进行对用户信息的输出。

## (3) 程序源码及说明

```
package step3;

public class Sysuser{

    String username;

    String realname;

    int age;

    String birthday;

    String password;

    Sysuser(String u,String r,int a,String b,String p){

        this.username=u;

        this.realname=r;

        this.age=a;

        this.birthday=b;

        this.password=p;

    }

    public void display() {

        System.out.println("用户名:"+this.username);
```

```
        System.out.println("真实姓名:"+this.realname);

        System.out.println("年龄:"+this.age);

        System.out.println("出生日期:"+this.birthday);

        System.out.println("密码:"+this.password);

    }

    public void setAge(int newAge) {

        this.age=newAge;

    }

}

package step3;

import java.util.Scanner;

public class TestSysuser{

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        System.out.println("按照用户名、真实姓名、年龄、出生日期、密码（全为数字）顺序输入数据：");

        String u=sc.next();

        String r=sc.next();

        int a=sc.nextInt();

        String b=sc.next();

        String p=sc.next();

        Sysuser us=new Sysuser(u,r,a, b, p);

        int input=0;

        while (input!=4) {
```

```
System.out.println("1.修改用户年龄");  
  
System.out.println("2.判断用户密码");  
  
System.out.println("3.显示用户信息");  
  
System.out.println("4.退出");  
  
input =sc.nextInt();  
  
if(input==3) {  
    us.display();  
}  
  
else if(input==1) {  
    System.out.println("请输入年龄: ");  
    int ag=sc.nextInt();  
    us.setAge(ag);  
}  
  
else if(input==2) {  
    System.out.println("请输入用户名: ");  
    String i=sc.next();  
    System.out.println("请输入密码: ");  
    String j=sc.next();  
    if(us.username.equals(i)&&us.password.equals(j)) {  
        System.out.println("登录成功");  
    }  
    else {  
        System.out.println("用户名或者密码错误");  
    }  
}
```

```
    }  
  
    }  
  
}  
  
}
```

(4) 运行结果(截图)



(5) 心得体会 (问题调试及体会)

学会了编写简单的类来解决实际的问题，可以熟练的编写简单的类解决问题，可以熟练的编写类，了解了 Java 中类的基本结构。

## 实验 3 核心类的使用

### 3.1 实验原理和要求

Java 本身提供了一些核心类供用户使用，例如：Math、Random 等，这些核心类中封装了一系列的方法，利用这些核心类极大地方便了用户解决相关的问题。

通过本次实验了解并且掌握 Java 中核心类的使用，可以熟练运用核心类解决相关问题。

### 3.2 实验任务

#### 3.2.1 任务一

##### （1）任务内容

一次从键盘接收 10 个整数，并排序输出。要求输出两行数据，第一行为升序排序结果，第二行为降序排序结果。

##### （2）构思过程(可用文字、流程图、UML 图等方式表达)

接受用户输入的数据并且存储到一个数组里面，运用冒泡排序对数组中的数据进行排序，按照要求分别进行升序排序输出和降序排序输出。

##### （3）程序源码及说明

```
package step1;

import java.util.Scanner;

public class Paixu {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        //获取用户输入

        int []a=new int[10];

        for(int i=0;i<10;i++)

        {
```

```
        a[i]=scan.nextInt();
    }
    //升序排序并输出
    int temp;
    for(int i=0;i<10;i++)
    {
        for(int j=i+1;j<10;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    for(int i= 0;i<10;i++)
    {
        if(i==9)
        {
            System.out.print(a[i]+"\\n");
        }
        else{
            System.out.print(a[i]+",");
        }
    }
}
```

```
    }  
}  
//降序排序并输出  
for(int i=0;i<10;i++)  
{  
    for(int j=i+1;j<10;j++)  
    {  
        if(a[i]<a[j])  
        {  
            temp=a[i];  
            a[i]=a[j];  
            a[j]=temp;  
        }  
    }  
}  
for(int i=0;i<10;i++)  
{  
    if(i==9)  
    {  
        System.out.print(a[i]+"\\n");  
    }  
    else{  
        System.out.print(a[i]+",");  
    }  
}
```

```

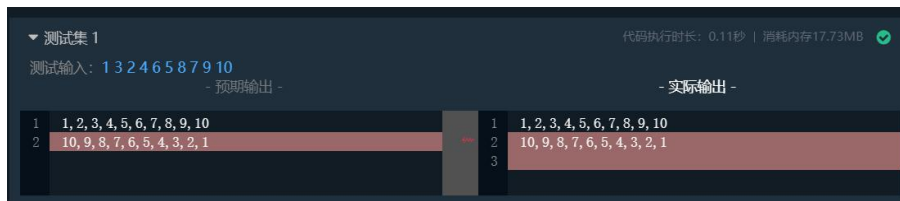
    }

    }

}

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

学会并且掌握了 Java 中数组的使用方法，遍历数组中的元素，以及数组的声明和使用，复习了相关的排序算法。

### 3.2.2 任务二

#### (1) 任务内容

编写一个小程序，实现循环从键盘接收字符串，连接成一个字符串并输出。

从键盘接收 5 个字符串，连接成一个字符串后输出。 注意如果输入中包含英文双引号，需要替换成中文双引号。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

循环接收用户输入的字符串，对字符串进行拼接，利用字符串中的 `replaceFirst()` 方法对字符串中的英文双引号进行替换。

#### (3) 程序源码及说明

```

package step2;

import java.util.Scanner;

public class Constring {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
    }
}

```



//循环获取用户输入，共 5 次，对 5 个字符串进行连接并输出。

```
String sb="";

String s;

for(int i=1;i<=5;i++)

{

    System.out.println("请输入第"+i+"个字符串:");

    s=sc.nextLine();

    sb+=s;

}

while(!sb.replaceFirst("\\\"", "\"").equals(sb)) {

    sb=sb.replaceFirst("\\\"", "\"");

    sb=sb.replaceFirst("\\'", "'");

}

System.out.println(sb);

}

}
```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

学会并且掌握了 Java 中字符串类的一系列方法，熟练掌握了对于字符串的处理，并且可以运用所学内容解决相关问题。

### 3.2.3 任务三

#### (1) 任务内容

用户输入的数字作为种子，实现随机生成 10 个不超过 100 的整数，按照从小到大的顺序排序后并输出，每个整数之间用英文逗号隔开。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先接受用户的输入作为随机数种子，利用 Random 类实例化一个对象，随机生成 10 个 100 以内的整数存入一个数组中，利用冒泡排序对数组中的数据进行顺序排序并且进行输出。

#### (3) 程序源码及说明

```
package step3;

import java.util.Random;
import java.util.Scanner;

public class RandomTest {

    public static void main(String[] args) {

        RandomTest rt = new RandomTest();

        rt.tryRandom();

    }

    public void tryRandom(){

        Scanner scan = new Scanner(System.in);

        System.out.println("请输入种子：");

        //完成代码编写，接收用户输入的种子，利用种子生成 10 个 100 以内
        //整数，并升序输出。

        long seed=scan.nextLong();

        Random r=new Random(seed);

        int []a=new int[10];

        for(int i=0;i<10;i++)
```

```
{  
    a[i]=r.nextInt(100);  
}  
  
int temp;  
for(int i=0;i<10;i++)  
{  
    for(int j=i+1;j<10;j++)  
    {  
        if(a[i]>a[j])  
        {  
            temp=a[i];  
            a[i]=a[j];  
            a[j]=temp;  
        }  
    }  
}  
  
for(int i=0;i<10;i++)  
{  
    if(i==9)  
    {  
        System.out.print(a[i]+"\\n");  
    }  
    else  
    {
```

```
        System.out.print(a[i]+",");

    }

}

}
```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

学会并且掌握了 Java 中 Random 类的使用方法，掌握了如何生成随机数，并且复习了冒泡排序方法。

### 3.2.4 任务四

#### (1) 任务内容

模仿福利彩票 23 选 5，随机生成 5 个不同的 1~23 的整数。注意输出两位数，格式为：01-08-09-02-19。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

接受用户的输入，把输入作为随机数种子，利用 Java 中的 random 类实例化一个对象，生成随机数。

注意生成随机数的范围，以及五个随机数必须是不同的随机数。

#### (3) 程序源码及说明

```
package step4;

import java.util.Random;

import java.util.Scanner;
```

```
import java.util.*;

public class CaipiaoCreate {

    public static void main(String[] args) {

        //完成彩票生成完整代码

        System.out.println("请输入种子: ");

        Scanner sc=new Scanner(System.in);

        int seed=sc.nextInt();

        Random r=new Random(seed);

        List <Integer> re=new ArrayList<Integer>();

        int i=0;

        while(true){

            int num=r.nextInt(24);

            if(re.contains(num) || num==0)

                continue;

            re.add(num);

            if(num<10)

                System.out.print("0"+num);

            else

                System.out.print(num);

            if(i<4)

                System.out.print("-");

            i++;

            if(i==5)

                break;

        }

    }

}
```

```
}  
  
}  
  
}
```

#### (4) 运行结果(截图)



#### (5) 心得体会 (问题调试及体会)

了解了 Java 中的一个核心类 Random 的一系列使用方法, 学会生成随机数的方法, 并且可以利用该类解决一些实际的问题。

## 实验 4 类继承的使用

### 4.1 实验原理和要求

面向对象是基于面向过程而言的，面向对象是将功能等通过对象来实现，将功能封装进对象之中，让对象去实现具体的细节；这种思想是将数据作为第一位，而方法或者说是算法作为其次，这是对数据一种优化，操作起来更加的方便，简化了过程。

面向对象是 Java 中一个重大的思想，通过本次实验学习类继承的使用，体会 Java 中面向对象的思想，熟练掌握 Java 中类的使用以及类的继承等内容

### 4.2 实验任务

#### 4.2.1 任务一

##### (1) 任务内容

声明一个抽象类 Pet，封装属性 name 和 sex，声明一个带有两个参数的构造函数，声明抽象方法 void talk() 和 void eat()；

声明一个 Dog 类继承自 Pet，封装属性 color，声明带有三个参数的构造函数，复写 talk() 和 eat() 方法；

声明一个 Cat 类继承自 Pet，封装属性 weight，声明带有三个参数的构造函数，复写 talk() 和 eat() 方法；

编写测试类，通过有参构造函数实例化 Dog 类对象，调用 talk() 方法和 eat() 方法；通过有参构造函数实例化 Cat 类对象，调用 talk() 方法和 eat() 方法；

##### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先声明抽象类 Pet，该类声明两个属性 name 和 sex，声明一个构造函数，此构造函数具有两个参数，并且为该抽象类声明两个抽象方法 talk() 和 eat()；

声明 Dog 类和 Cat 类，两个类都是继承 Pet 类，Dog 类中再声明一个 color 属性，声明含有三个参数的构造方法，复写抽象类中的两个方法；Cat 类声明一个 weight 属性，声明含有三个参数的构造方法，复写抽象类中的两个方法；

编写测试类，实例化 Cat 和 Dog 对象，并且调用其中的方法。

##### (3) 程序源码及说明

```
package case1;

import java.util.Scanner;

public class Task1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String dogName = sc.next();

        String dogSex = sc.next();

        String dogColor = sc.next();

        String catName = sc.next();

        String catSex = sc.next();

        System.out.println("名称: ");

        System.out.println("性别: ");

        System.out.println("颜色: ");

        System.out.println("名称: ");

        System.out.println("性别: ");

        System.out.println("体重: ");

        double catWeight = sc.nextDouble();

        // 通过有参构造函数实例化 Dog 类对象 dog

        // dog 调用 talk() 方法

        // dog 调用 eat() 方法

        /***** begin *****/

        Dog dog=new Dog(dogName,dogSex,dogColor);

        dog.talk();

        dog.eat();

    }

}
```



```

        /***** end *****/

        // 通过有参构造函数实例化 Cat 类对象 cat

        // cat 调用 talk() 方法

        // cat 调用 eat() 方法

        /***** begin *****/

        Cat cat=new Cat(catName,catSex,catWeight);

        cat.talk();

        cat.eat();

        /***** end *****/

    }

}

// 抽象类 Pet 封装属性 name 和 sex

// 构造函数初始化 name 和 sex

// 声明抽象方法 talk()

// 声明抽象方法 eat()

abstract class Pet {

    /***** begin *****/

    public String name;

    public String sex;

    public Pet(String name,String sex){

        this.name=name;

        this.sex=sex;

    }

    public abstract void talk();

```

```

        public abstract void eat();

        /***** end *****/
    }

    // Dog 类继承自 Pet 类 封装属性 color
    // 构造函数初始化 name、sex 和 color
    // 实现自己的 talk() 方法和 eat() 方法
    // talk() 输出 '名称: name, 性别: sex, 颜色: color, 汪汪叫'
    // eat() 输出 'name 吃骨头'
    class Dog extends Pet {

        /***** begin *****/

        public String color;

        public Dog(String name,String sex,String color){

            super(name,sex);

            this.color=color;

        }

        public void talk(){

            System.out.println("名称: "+this.name+", 性别: "+this.sex+", 颜色: "+this.color+", 汪汪叫");

        }

        public void eat(){

            System.out.println(this.name+"吃骨头! ");

        }

        /***** end *****/

    }

```

```
// Cat 类继承自 Pet 类 封装属性 weight
// 构造函数初始化 name、sex 和 weight
// 实现自己的 talk() 方法和 eat() 方法
// talk() 输出 '名称: name, 性别: sex, 体重: weight kg, 喵喵叫'
// eat() 输出 'name 吃鱼'
class Cat extends Pet {
    /***** begin *****/
    public double weight;
    public Cat(String name,String sex,double weight){
        super(name,sex);
        this.weight=weight;
    }
    public void talk(){
        System.out.println("名称: "+this.name+", 性别: "+this.sex+", 体重: "+this.weight+"kg, 喵喵叫");
    }
    public void eat(){
        System.out.println(this.name+"吃鱼!");
    }
    /***** end *****/
}
```

#### (4) 运行结果(截图)

```
▼ 测试集 1
测试输入: 泰迪 male brown 波斯猫 male 2.5
- 预期输出 -
1 名称:
2 性别:
3 颜色:
4 名称:
5 性别:
6 体重:
7 名称: 泰迪, 性别: male, 颜色: brown, 汪汪叫
8 泰迪吃骨头!
9 名称: 波斯猫, 性别: male, 体重: 2.5kg, 喵喵叫
10 波斯猫吃鱼!
11

- 实际输出 -
1 名称:
2 性别:
3 颜色:
4 名称:
5 性别:
6 体重:
7 名称: 泰迪, 性别: male, 颜色: brown, 汪汪叫
8 泰迪吃骨头!
9 名称: 波斯猫, 性别: male, 体重: 2.5kg, 喵喵叫
10 波斯猫吃鱼!
11

代码执行时长: 0.13秒 | 消耗内存18.83MB ✓
```

```
▼ 测试集 2
测试输入: 萨摩耶 female white 加菲猫 female 1.5
- 预期输出 -
1 名称:
2 性别:
3 颜色:
4 名称:
5 性别:
6 体重:
7 名称: 萨摩耶, 性别: female, 颜色: white, 汪汪叫
8 萨摩耶吃骨头!
9 名称: 加菲猫, 性别: female, 体重: 1.5kg, 喵喵叫
10 加菲猫吃鱼!
11

- 实际输出 -
1 名称:
2 性别:
3 颜色:
4 名称:
5 性别:
6 体重:
7 名称: 萨摩耶, 性别: female, 颜色: white, 汪汪叫
8 萨摩耶吃骨头!
9 名称: 加菲猫, 性别: female, 体重: 1.5kg, 喵喵叫
10 加菲猫吃鱼!
11

代码执行时长: 0.13秒 | 消耗内存20.04MB ✓
```

#### (5) 心得体会（问题调试及体会）

了解了 Java 中面向对象的思想，了解了封装的概念，了解了继承的概念，学会了如何声明类中的构造函数，学会了使用 `super()` 和 `this()`。

面向对象是将功能等通过对象来实现，将功能封装进对象之中，让对象去实现具体的细节；访问权限的控制常被称为是具体实现的隐藏。把数据和方法包装进类中，以及具体实现的隐藏共同被称为封装。继承使子类拥有父类所有的属性和方法，但是父类对象中的私有属性和方法，子类是无法访问到的，只是拥有，但不能使用。子类不能继承父类的构造函数，只是显式或隐式调用，可以从子类调用超类的构造函数。构造函数用来在对象实例化时初始化对象的成员变量。`super()` 关键字表示超类的意思，当前类是从超类继承而来。`this` 表示当前对象。

### 4.2.2 任务一

#### (1) 任务内容

按照要求编写一个 Java 应用程序：

定义一个抽象类 `Person`，包含抽象方法 `eat()`，封装属性 `name`、`sex`、`age`，声明包含三个参数的构造方法；

定义一个 Chinese 类，继承自 Person 类，重写父类的 eat() 方法，并定义一个自己特有的方法 shadowBoxing();

定义一个 English 类，继承自 Person 类，重写父类的 eat() 方法，并定义一个自己特有的方法 horseRiding();

编写测试类，定义一个 showEat() 方法，使用父类作为方法的形参，实现多态，分别调用 showEat() 方法，通过强制类型转换调用各自类特有的方法；

## (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先声明一个 Person 抽象类，声明抽象方法 eat()，声明 name、sex、age 属性，声明包含三个参数的构造方法。

定义 Chinese 类和 English 类，都继承自 Person 类，都重写父类的 eat() 方法，并且声明自己的方法。

编写测试类，编写一个 showEat 方法，使用父类作为方法的形参，分别调用该方法，实现多态。

## (3) 程序源码及说明

```
package case2;

import java.util.Scanner;

public class Task2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String cName = sc.next();

        String cSex = sc.next();

        int cAge = sc.nextInt();

        String eName = sc.next();

        String eSex = sc.next();

        int eAge = sc.nextInt();

        // 创建测试类对象 test

        // 创建 Person 类对象 person1, 引用指向中国人, 通过有参构造函数实例化中国人类对象
```

```

        // 通过 showEat() 方法调用 Chinese 的 eat() 方法

        // 创建 Person 类对象 person2, 引用指向英国人, 通过有参
构造函数实例化英国人类对象

        // 通过 showEat() 方法调用 English 的 eat() 方法
        /***** begin *****/

        Person test;

        System.out.println("姓名: ");

        System.out.println("性别: ");

        System.out.println("年龄: ");

        System.out.println("姓名: ");

        System.out.println("性别: ");

        System.out.println("年龄: ");

        Person person1=new Chinese(cName,cSex,cAge);

        showEat(person1);

        Person person2=new English(eName,eSex,eAge);

        showEat(person2);

        /***** end *****/

        // 强制类型转换(向下转型) 调用 Chinese 类特有的方法
shadowBoxing()

        // 强制类型转换(向下转型) 调用 English 类特有的方法
horseRiding()

        /***** begin *****/

        Chinese d=(Chinese)person1;

        d.shadowBoxing();

```

```

        English e= (English)person2;

        e.horseRiding();

        /***** end *****/

    }

    // 定义 showEat 方法，使用父类作为方法的形参，实现多态，传
    入的是哪个具体对象就调用哪个对象的 eat() 方法

    /***** begin *****/

    public static void showEat(Person p){

        p.eat();

    }

    /***** end *****/

}

// 抽象类 Person 封装属性 name、sex 和 age
// 构造函数初始化 name、sex 和 age
// 声明抽象方法 eat()
abstract class Person {

    /***** begin *****/

    public abstract void eat();

    public String name;

    public String sex;

    public int age;

    /***** end *****/

}

// Chinese 类继承自 Person 类

```

```
// 构造函数初始化 name、sex 和 age

// 重写父类方法 eat() 输出'姓名: name, 性别: sex, 年龄: age,
我是中国人, 我喜欢吃饭! '

// 定义子类特有方法 shadowBoxing(), 当父类引用指向子类对象时
无法调用该方法 输出'name 在练习太极拳! '

class Chinese extends Person {
    /******* begin *****/

    public String name;

    public String sex;

    public int age;

    public Chinese(String name,String sex,int age){

        this.name=name;

        this.sex=sex;

        this.age=age;

    }

    public void eat(){

        System.out.println(" 姓 名: "+this.name+" ,   性 别:
        "+this.sex+" ,  年 龄: "+this.age+" , 我是中国人, 我喜欢吃饭!
        ");

    }

    public void shadowBoxing(){

        System.out.println(this.name+"在练习太极拳! ");

    }

    /******* end *****/
}
```



```
}

// English 类继承自 Person 类

// 构造函数初始化 name、sex 和 age

// 重写父类方法 eat() 输出 '姓名: name, 性别: sex, 年龄: age,
我是英国人, 我喜欢吃三明治!'

// 定义子类特有方法 horseRiding(), 当父类引用指向子类对象时无
法调用该方法 输出 'name 在练习骑马!'

class English extends Person {

    /******* begin *****/

    String name;

    String sex;

    int age;

    public English(String name,String sex,int age){

        this.name=name;

        this.sex=sex;

        this.age=age;

    }

    public void eat(){

        System.out.println(" 姓 名: "+this.name+", 性 别:
        "+this.sex+", 年龄: "+this.age+", 我是英国人, 我喜欢吃三
        明 治! ");

    }

    public void horseRiding(){

        System.out.println(this.name+"在练习骑马! ");

    }

}
```

```

}

/***** end *****/

}

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

学会并且掌握了 Java 中的重写和重载，学会声明抽象类和接口，并且了解了 final 关键字和 static 关键字的使用方法，了解了 Java 中多态的思想以及如何使用多态。

方法重载（overload）必须是同一个类；方法名（也可以叫函数）一样；参数类型不一样或参数数量或顺序不一样；不能通过返回值来判断重载。

方法的重写（override）子类重写了父类的同名方法，方法名相同，参数类型相同；子类返回类型是父类返回类型的子类；子类抛出异常小于等于父类方法抛出异常；子类访问权限大于等于父类方法访问权限。

用 abstract 修饰的类表示抽象类，抽象类位于继承树的抽象层，抽象类不能被实例化。用 abstract 修饰的方法表示抽象方法，抽象方法没有方法体。抽象方法用来描述系统具有什么功能，但不提供具体的

实现，把具体实现留给继承该类的子类。

interface 中的方法默认为 public abstract (public、abstract 可以省略)，变量默认为 public static final；类中的方法全部都是抽象方法。只有声明没有实现，在不同类中有不同的方法实现。

多态是指不同类的对象对同一消息做出响应。同一消息可以根据发送对象的不同而采用多种不同的行为方式；多态存在的三个必要条件：继承、重写、父类引用指向子类对象；Java 中多态的实现方式：接口实现，继承父类进行方法重写，同一个类中进行方法重载。

### 4.2.3 任务三

#### (1) 任务内容

教练和运动员案例：

乒乓球运动员和篮球运动员；

乒乓球教练和篮球教练；

跟乒乓球相关的人员都需要学习英语；

分析，这个案例中有哪些抽象类，哪些接口，哪些具体类。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先定义人的抽象类 Person 封装 name 和 age，声明无参构造函数，声明有参构造函数初始化 name 和 age，定义具体方法 sleep() 输出‘人都是要睡觉的’，声明抽象方法 eat()。

定义运动员抽象类 Player 继承自 Person 类，声明无参构造函数，声明有参构造函数初始化 name 和 age，定义抽象方法 study()。

定义教练抽象类 Coach 继承自 Person 类，声明无参构造函数，声明有参构造函数初始化 name 和 age，定义抽象方法 teach()。

定义乒乓球运动员具体类 PingPangPlayer 继承自 Player 类并实现 SpeakEnglish 类，声明无参构造函数和有参构造函数初始化 name 和 age，实现自己的 eat() 方法、study() 方法、speak() 方法。

/定义篮球运动员具体类 BasketballPlayer 继承自 Player 类，声明 无参构造函数和有参构造函数初始化 name 和 age，实现自己的 eat() 方法和 study() 方法。

定义乒乓球教练具体类 PingPangCoach 继承自 Coach 类并实现 SpeakEnglish 类，声明无参构造函数和有参构造函数初始化 name 和 age，实现自己的 eat() 方法、teach() 方法和 speak() 方法。

定义篮球教练具体类 BasketballCoach 继承自 Coach 类声明 无参构造函数和有参构造函数初始化 name 和 age，实现自己的 eat() 方法、teach() 方法。

编写测试类，分别实例化各个对象，并且调用其中的各自的方法。

### (3) 程序源码及说明

```
package case3;

import java.util.Scanner;

public class Task3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String pppName = sc.next();

        int pppAge = sc.nextInt();

        String bpName = sc.next();

        int bpAge = sc.nextInt();

        String ppcName = sc.next();

        int ppcAge = sc.nextInt();

        String bcName = sc.next();

        int bcAge = sc.nextInt();

        // 测试运动员 (乒乓球运动员和篮球运动员)

        // 乒乓球运动员

        // 通过带参构造函数实例化 PingPangPlayer 对象 ppp

        // 输出 'name---age'

        // 分别调用 sleep()、eat()、study()、speak() 方法

        /***** begin *****/

        System.out.println("请输入乒乓球运动员姓名年龄:");

        System.out.println("请输入篮球运动员姓名年龄:");

        System.out.println("请输入乒乓球教练姓名年龄:");

        System.out.println("请输入篮球教练姓名年龄:");

    }

}
```

```

PingPangPlayer                                     ppp=new

PingPangPlayer (pppName,pppAge) ;

System.out.println (pppName+"---"+pppAge) ;

ppp.sleep() ;

ppp.eat() ;

ppp.study() ;

ppp.speak() ;

        /***** end *****/

        System.out.println("-----");

        // 篮球运动员

        // 通过带参构造函数实例化 BasketballPlayer 对象 bp

        // 输出 'name---age'

        // 分别调用 sleep()、eat()、study() 方法

        /***** begin *****/

BasketballPlayer                                     bp=new

BasketballPlayer (bpName,bpAge) ;

System.out.println (bpName+"---"+bpAge) ;

bp.sleep() ;

bp.eat() ;

bp.study() ;

        /***** end *****/

        System.out.println("-----");

        // 测试教练 (乒乓球教练和篮球教练)

        // 乒乓球教练

```

```

// 通过带参构造函数实例化 PingPangCoach 对象 ppc

// 输出 'name---age'

// 分别调用 sleep()、eat()、teach()、speak() 方法

/***** begin *****/

PingPangCoach ppc=new PingPangCoach(ppcName,ppcAge);

System.out.println(ppcName+"---"+ppcAge);

ppc.sleep();

ppc.eat();

ppc.teach();

ppc.speak();

/***** end *****/

System.out.println("-----");

// 篮球教练

// 通过带参构造函数实例化 BasketballCoach 对象 bc

// 输出 'name---age'

// 分别调用 sleep()、eat()、teach() 方法

/***** begin *****/

BasketballCoach                                     bc=new

BasketballCoach(bcName,bcAge);

System.out.println(bcName+"---"+bcAge);

bc.sleep();

bc.eat();

bc.teach();

/***** end *****/

```

```
        System.out.println("-----");
    }
}

// 说英语接口 声明抽象方法 speak()
interface SpeakEnglish {
    /******* begin *****/
    abstract void speak();
    /******* end *****/
}

// 定义人的抽象类 Person 封装 name 和 age
// 无参构造函数
// 有参构造函数初始化 name 和 age
// 定义具体方法 sleep() 输出'人都是要睡觉的'
// 抽象方法 eat() (吃的不一样)
abstract class Person {
    /******* begin *****/
    String name;
    int age;
    Person(String name,int age){
        this.name=name;
        this.age=age;
    }
    void sleep(){
        System.out.println("人都是要睡觉的");
    }
}
```

```

}

abstract void eat();

    /******* end *****/

}

// 定义运动员 Player（抽象类）继承自 Person 类
// 无参构造函数
// 有参构造函数初始化 name 和 age
// 运动员学习内容不一样，抽取为抽象 定义抽象方法 study()
abstract class Player extends Person {

    /******* begin *****/

    Player(String name,int age){

        super(name,age);

    }

    abstract void study();

    /******* end *****/

}

// 定义教练 Coach（抽象类）继承自 Person 类
// 无参构造函数
// 有参构造函数初始化 name 和 age
// 教练教的不一樣 定义抽象方法 teach()
abstract class Coach extends Person {

    /******* begin *****/

    Coach(String name,int age){

        super(name,age);

    }

}

```



```

    }

    abstract void teach();

        /***** end *****/
    }

// 定义乒乓球运动员具体类 PingPangPlayer 继承自 Player 类并
实现 SpeakEnglish 类（乒乓球运动员需要说英语）

// 无参构造函数

// 有参构造函数初始化 name 和 age

// 实现自己的 eat() 方法 输出'乒乓球运动员吃大白菜，喝小米粥'
// 实现自己的 study() 方法 输出'乒乓球运动员学习如何发球和接球'
// 实现自己的 speak() 方法 输出'乒乓球运动员说英语'

class PingPangPlayer extends Player implements
SpeakEnglish {

    /***** begin *****/

PingPangPlayer(String name,int age){

    super(name,age);

}

void eat(){

    System.out.println("乒乓球运动员吃大白菜，喝小米粥");

}

void study(){

    System.out.println("乒乓球运动员学习如何发球和接球");

}

public void speak(){

```

```

        System.out.println("乒乓球运动员说英语");
    }

    /***** end *****/
}

// 定义篮球运动员具体类 BasketballPlayer 继承自 Player 类 不需要继承接口，因为他不需要说英语

// 无参构造函数

// 有参构造函数初始化 name 和 age

// 实现自己的 eat() 方法 输出'篮球运动员吃牛肉，喝牛奶'

// 实现自己的 study() 方法 输出'篮球运动员学习如何运球和投篮'

class BasketballPlayer extends Player {

    /***** begin *****/

    BasketballPlayer(String name,int age){

        super(name,age);

    }

    void eat(){

        System.out.println("篮球运动员吃牛肉，喝牛奶");

    }

    void study(){

        System.out.println("篮球运动员学习如何运球和投篮");

    }

    /***** end *****/

}

// 定义乒乓球教练具体类 PingPangCoach 继承自 Coach 类并实现

```

SpeakEnglish 类（乒乓球教练需要说英语）

// 无参构造函数

// 有参构造函数初始化 name 和 age

// 实现自己的 eat() 方法 输出'乒乓球教练吃小白菜，喝大米粥'

// 实现自己的 teach() 方法 输出'乒乓球教练教如何发球和接球'

// 实现自己的 speak() 方法 输出'乒乓球教练说英语'

```
class PingPangCoach extends Coach implements  
SpeakEnglish {
```

```
    /***** begin *****/
```

```
PingPangCoach(String name,int age){
```

```
    super(name,age);
```

```
}
```

```
void eat(){
```

```
    System.out.println("乒乓球教练吃小白菜，喝大米粥");
```

```
}
```

```
void teach(){
```

```
    System.out.println("乒乓球教练教如何发球和接球");
```

```
}
```

```
public void speak(){
```

```
    System.out.println("乒乓球教练说英语");
```

```
}
```

```
    /***** end *****/
```

```

}

// 定义篮球教练具体类 BasketballCoach 继承自 Coach 类 不需要
// 继承接口，因为他不需要说英语

// 无参构造函数

// 有参构造函数初始化 name 和 age

// 实现自己的 eat() 方法 输出 '篮球教练吃羊肉，喝羊奶'

// 实现自己的 teach() 方法 输出 '篮球教练教如何运球和投篮'

class BasketballCoach extends Coach {

    /******* begin *****/

    BasketballCoach(String name,int age){

        super(name,age);

    }

    void eat(){

        System.out.println("篮球教练吃羊肉，喝羊奶");

    }

    void teach(){

        System.out.println("篮球教练教如何运球和投篮");

    }

    /******* end *****/

}

```

#### (4) 运行结果(截图)

```
测试输入: 张继科 30 易建联 31 刘国梁 42 杜锋 37
- 预期输出 -
1  请输入乒乓球运动员姓名年龄:
2  请输入篮球运动员姓名年龄:
3  请输入乒乓球教练姓名年龄:
4  请输入篮球教练姓名年龄:
5  张继科——30
6  人都是要睡觉的
7  乒乓球运动员吃大白菜, 喝小米粥
8  乒乓球运动员学习如何发球和接球
9  乒乓球运动员说英语
10
11 易建联——31
12 人都是要睡觉的
13 篮球运动员吃牛肉, 喝牛奶
14 篮球运动员学习如何运球和投篮
15
16 刘国梁——42
17 人都是要睡觉的
18 乒乓球教练吃小白菜, 喝大米粥
19 乒乓球教练如何发球和接球
20 乒乓球教练说英语
21
22 杜锋——37
23 人都是要睡觉的
24 篮球教练吃羊肉, 喝羊奶
25 篮球教练如何运球和投篮
26

- 实际输出 -
1  请输入乒乓球运动员姓名年龄:
2  请输入篮球运动员姓名年龄:
3  请输入乒乓球教练姓名年龄:
4  请输入篮球教练姓名年龄:
5  张继科——30
6  人都是要睡觉的
7  乒乓球运动员吃大白菜, 喝小米粥
8  乒乓球运动员学习如何发球和接球
9  乒乓球运动员说英语
10
11 易建联——31
12 人都是要睡觉的
13 篮球运动员吃牛肉, 喝牛奶
14 篮球运动员学习如何运球和投篮
15
16 刘国梁——42
17 人都是要睡觉的
18 乒乓球教练吃小白菜, 喝大米粥
19 乒乓球教练如何发球和接球
20 乒乓球教练说英语
21
22 杜锋——37
23 人都是要睡觉的
24 篮球教练吃羊肉, 喝羊奶
25 篮球教练如何运球和投篮
26

测试输入: 许昕 28 周琦 22 孔令辉 43 宫鲁鸣 61
- 预期输出 -
1  请输入乒乓球运动员姓名年龄:
2  请输入篮球运动员姓名年龄:
3  请输入乒乓球教练姓名年龄:
4  请输入篮球教练姓名年龄:
5  许昕——28
6  人都是要睡觉的
7  乒乓球运动员吃大白菜, 喝小米粥
8  乒乓球运动员学习如何发球和接球
9  乒乓球运动员说英语
10
11 周琦——22
12 人都是要睡觉的
13 篮球运动员吃牛肉, 喝牛奶
14 篮球运动员学习如何运球和投篮
15
16 孔令辉——43
17 人都是要睡觉的
18 乒乓球教练吃小白菜, 喝大米粥
19 乒乓球教练如何发球和接球
20 乒乓球教练说英语
21
22 宫鲁鸣——61
23 人都是要睡觉的
24 篮球教练吃羊肉, 喝羊奶
25 篮球教练如何运球和投篮
26

- 实际输出 -
1  请输入乒乓球运动员姓名年龄:
2  请输入篮球运动员姓名年龄:
3  请输入乒乓球教练姓名年龄:
4  请输入篮球教练姓名年龄:
5  许昕——28
6  人都是要睡觉的
7  乒乓球运动员吃大白菜, 喝小米粥
8  乒乓球运动员学习如何发球和接球
9  乒乓球运动员说英语
10
11 周琦——22
12 人都是要睡觉的
13 篮球运动员吃牛肉, 喝牛奶
14 篮球运动员学习如何运球和投篮
15
16 孔令辉——43
17 人都是要睡觉的
18 乒乓球教练吃小白菜, 喝大米粥
19 乒乓球教练如何发球和接球
20 乒乓球教练说英语
21
22 宫鲁鸣——61
23 人都是要睡觉的
24 篮球教练吃羊肉, 喝羊奶
25 篮球教练如何运球和投篮
26
```

#### (5) 心得体会 (问题调试及体会)

复习了类的所有知识, 对类中的各个知识进行了一个整合, 包括类的继承、重写和重载、Java 多态、Java 抽象类、Java 封装以及 Java 接口等内容。

类是 Java 语言的一大特色, 必须熟练掌握其中的各种知识点, 并且能够熟练应用。

## 实验 5 Java 语言之接口

### 5.1 实验原理和要求

接口是 Java 语言的一大特点，接口和类的编写方法很相似，但是接口不是类。接口包含类要实现的方法，接口无法实例化，但是可以被实现。一个实现接口的类。必须实现接口内所描述的所有方法，否则就必须声明为抽象类。

接口没有构造方法，接口中所有的方法必须是抽象方法，接口不能包含成员变量，除了 `static` 和 `final` 变量。接口不是被类继承，而是要被类实现，接口支持多继承。

接口中每一个方法是隐式抽象的，接口中的方法会被隐式的指定为 `public abstract`（只能是 `public abstract`，其他修饰符都会报错）。

接口中可以含有变量，但是接口中的变量会被隐式的指定为 `public static final` 变量（并且只能是 `public`，用 `private` 修饰会报编译错误。

通过本次实验，了解并且掌握接口的使用，包括接口的声明，接口的实现以及接口的继承等，学会怎么使用接口，并且能够熟练运用接口。

### 5.2 实验任务

#### 5.2.1 任务一

##### （1）任务内容

定义一个 `Introduce` 接口，包括一个 `introduction()` 方法，输出格式见测试样例。

定义一个 `Student` 类和一个 `Teacher` 类实现 `Introduce` 接口。

##### （2）构思过程(可用文字、流程图、UML 图等方式表达)

声明一个 `Introduce` 接口，接口的声明需要使用 `interface` 关键字，其中接口中声明一个 `introduction()` 方法，定义 `student` 类和 `teacher` 类实现接口，利用 `implements` 关键字实现接口，并且需要实现接口中声明的方法。

##### （3）程序源码及说明

```
package step3;
```

```
import java.util.Objects;
import java.util.Scanner;

public class StudentOrTeacher {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        String string = scanner.next();

        if (Objects.equals(string, "学生")) {

            Student student = new Student();

            student.introduction();

        }

        else if (Objects.equals(string, "老师")) {

            Teacher teacher = new Teacher();

            teacher.introduction();

        }

    }

}

/***** Begin *****/

interface Introduce {

    public void introduction();

}

class Student implements Introduce {

    public void introduction() {

        System.out.println("我是一名学生!");

    }

}
```

```

}

class Teacher implements Introduce {

    public void introduction() {

System.out.println("我是一名老师！");

    }

}

/***** End *****/

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

接口中包含类要实现的方法，接口无法被实例化，但是可以被实现。一个实现接口的类必须实现接口内所描述的所有方法，否则就必须声明为抽象类。

通过此任务，学会并掌握了如何声明接口，并且如何实现接口，了解了接口的相关特点，并且能够熟练使用接口。

### 5.2.2 任务二

#### (1) 任务内容

完成 MyFavouriteTeam 类以实现其需要实现的接口，具体要求如下：  
 通过 setSports() 设置喜欢的体育赛事类型，如篮球、足球；  
 通过 setTeam() 设置支持的队伍，如洛杉矶湖人、皇家马德里等；  
 通过 introduction() 大声说出你喜欢的运动，和喜欢的球队，格式如下：“我最喜欢的体育运动是 xxx，我最喜欢的球队是 xxx”。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

接口可以多继承，通过多继承，子接口继承父接口的方法，



ChooseTeam 接口继承 Sports 接口和 Team 接口实现接口的多继承 (ChooseTeam extends Sports, Team)，声明 MyFavouriteTeam 类实现 ChooseTeam 接口 (class MyFavouriteTeam implements ChooseTeam)，并且实现每个接口中声明的方法。

### (3) 程序源码及说明

```
package step4;

import java.util.Scanner;

public class SportsEvents {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        String sports = scanner.next();

        String team = scanner.next();

        MyFavouriteTeam    myFavouriteTeam    =    new
MyFavouriteTeam();

        myFavouriteTeam.setSports(sports);

        myFavouriteTeam.setTeam(team);

        myFavouriteTeam.introduction();

    }

}

interface Sports {

    void setSports(String sports);

}

interface Team {

    void setTeam(String team);

}

interface ChooseTeam extends Sports, Team {
```

```

        void introduction();
    }

    /***** Begin *****/

    class MyFavouriteTeam implements ChooseTeam {

    public String sports;

        public String team;

        public void setSports(String sports){

            this.sports=sports;

        }

        public void setTeam(String team){

            this.team=team;

        }

        public void introduction(){

            System.out.println(" 我最喜欢的体育运动是
            "+this.sports+", 我最喜欢的球队是"+this.team);

        }

    }

    /***** End *****/

```

#### (4) 运行结果(截图)



```
▼ 测试集 2
测试输入: 足球 皇家马德里
- 预期输出 -
1 我最喜欢的体育运动是足球, 我最喜欢的球队是皇家马德里
2
- 实际输出 -
1 我最喜欢的体育运动是足球, 我最喜欢的球队是皇家马德里
2
```

### （5）心得体会（问题调试及体会）

在 Java 中，类的多继承是不合法，但接口允许多继承。

在接口的多继承中 extends 关键字只需要使用一次，在其后跟着继承接口。

一个接口能继承另一个接口，和类之间的继承方式比较相似。接口的继承也是使用 extends 关键字，子接口继承父接口的方法。

学会并且掌握了 Java 中接口的继承，接口可以实现多继承，弥补了 Java 中类不可以多继承的缺陷，子接口继承父接口中声明的所有方法。

## 实验 6 Java 中的异常处理

### 6.1 实验原理和要求

异常是指程序在运行过程中产生的不正常情况。程序在运行的时候，可能会发生一些没有预料到的事件，导致程序没有按照我们事先编写好的代码运行，这就是异常。

Java 中的异常可以分为检查性异常和运行时异常，运行时异常程序员可以不用处理，当异常出现时，虚拟机会处理。常见的运行时异常有空指针异常、数据存储异常、数组越界等。而检查性异常需要我们去处理，如果我们不处理的话，程序是不能被编译的。

Java 中所有的异常类都是从 `java.lang.Exception` 类继承的子类，异常类有两个主要的子类：`IOException` 类和 `RuntimeException` 类。

通过本次实验了解并且学会 Java 中对于异常的处理，学会使用 `try`，`catch` 捕获异常，使用 `throw` 抛出异常，并且学会使用自定义异常类。

### 6.2 实验任务

#### 6.2.1 任务一

##### （1）任务内容

编辑器中的代码运行时可能会有异常，利用本关知识处理该异常。捕获程序的异常，输出异常处理的结果。

##### （2）构思过程(可用文字、流程图、UML 图等方式表达)

对于程序可能产生的异常使用 `try`，`catch` 进行捕捉异常并且处理异常，对于可能产生异常的代码放到 `try` 里面，并且使用 `catch` 捕获异常并且进行处理。

##### （3）程序源码及说明

```
package step2;

import java.util.Scanner;

public class Task {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
```

```

        int num1 = sc.nextInt();

        int num2 = sc.nextInt();

        /***** Begin *****/

        try{

            int c=num1/num2;

        }

        catch(Exception e){

            System.out.println("除数不能为 0");

        }

        System.out.println(num1/num2);

        /***** End *****/

    }

}

```

#### （4）运行结果(截图)



#### （5）心得体会（问题调试及体会）

对于 Java 中的异常处理机制有了一个清楚的认识，学会并且掌握了 try, catch 异常处理机制。当程序存在检测型异常的时候，程序是

不能编译通过的，就需要我们进行异常处理。Java 中的异常处理是通过 try, catch 来实现的，将可能产生异常的代码放到 try 中，利用 catch 捕捉异常并且进一步对异常进行处理。

## 6.2.2 任务二

### (1) 任务内容

异常的抛出和处理。对于产生的异常进行抛出异常并且处理异常。

### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

使用 throw 关键字主动抛出异常，对于程序中可能产生的异常使用 throw 关键字抛出异常，注意使用 throw 关键字主动抛出检测性异常的时候，在方法名上必须使用 throws 表明调用这个方法可能存在要抛出的异常。

### (3) 程序源码及说明

```
package step3;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class Task {

    /******* Begin *****/

    //请在合适的部位添加代码

    public static void main(String[] args) throws

FileNotFoundException {

        test();

    }

    public static void test() throws

FileNotFoundException {
```

```

        File file = new File("abc");

        if(!file.exists()){           //判断文件是否存在

            //文件不存在，则 抛出 文件不存在异常

            throw new FileNotFoundException("该文件不存在

");

        }else{

            FileInputStream          fs          =          new

FileInputStream(file);

        }

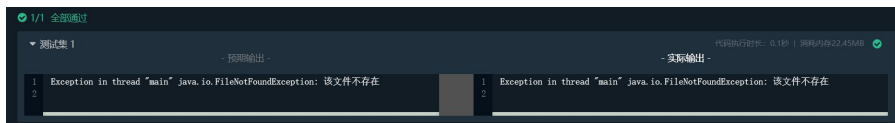
    }

    /***** End *****/

}

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

学会并且掌握了利用 throw 语句主动抛出异常，当程序出现某种逻辑错误时由程序员主动抛出某种特定类型的异常。使用 throw 关键字主动抛出检测性异常的时候，在方法名上必须使用 throws 表明调用这个方法可能存在要抛出的异常。FileNotFoundException 是属于检测性异常，是在编译之前就需要处理的，所以第二段程序要加上 throws 才能通过编译。

### 6.2.3 任务三

#### (1) 任务内容

定义一个自定义异常，判断用户名是否小于三位，如果用户名小于

三位，就抛出一个自定义异常。

### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

通过自定义异常来解决此任务，首先接受用户的输入，利用 String 的 length 方法判断用户输入的用户名的长度，假如用户输入的用户名的长度小于 3 就抛出自定义的异常。

定义一个自定义异常类，继承 Exception 类，当用户的输入不合法时，抛出自定义异常。

### (3) 程序源码及说明

```
package step4;

import java.util.Scanner;

public class Task {

    /******* Begin *****/

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

        Scanner sc = new Scanner(System.in);

        String username = sc.next();

        System.out.print("请输入用户名: ");

        //判断用户名

        if(username.length()<3)

        {

            throw new MyException("用户名小于三位 Exception");

        }

        else

        {

            System.out.print("用户名格式正确");

        }

    }

}
```



```

    }

}

class MyException extends Exception {

    public MyException(){}

    public MyException(String msg){

        super(msg);

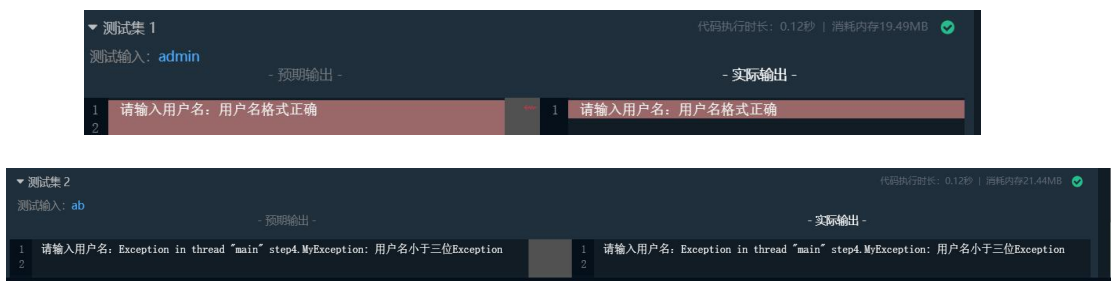
    }

}

/***** End *****/

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

在开发一个复杂的项目的时候会经常遇到系统自带的异常不能满足我们的需要的时候，这个时候就需要我们来自定义异常，自定义的异常一般需要继承 `Exception` 类，自定义异常继承 `Exception` 类，再将信息传递给父类就可以了。

学会并且掌握了如何使用自定义异常类来解决问题，学会了抛出自定义异常。

# 实验 7 基于 HTML、CSS、JAVASCRIPT 网页设计

## 7.1 实验原理和要求

HTML 是超文本标记语言，用来编写静态的页面，所谓的静态页面指的是页面上的数据不会随着用户操作进行改变。CSS 是层叠样式表，用来对页面中的样式进行修饰，对页面进行布局。JavaScript 是一种脚本语言，可以实现和用户的交互效果。

通过本次实验，学会 HTML 静态页面的编写，熟练掌握 HTML 技术，并且学会利用 CSS 进行页面的美化，熟练掌握 CSS 中的盒子模型等技术，掌握利用 JavaScript 在前端对数据进行验证，例如验证用户名长度超过 4 位等。

## 7.2 实验任务

### 7.2.1 任务一

#### (1) 任务内容

制作用户注册和登录界面，要求有自己的风格，完成 html+css，实现 JavaScript 相关的验证功能，如用户名非空、用户名长度要超过 4 位、邮件名称是否规范等等）。

#### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先进行对页面的大致 UI 设计，设计各个模块该有什么功能，该展示的大概样子。接着进行 HTML 代码的编写以及 CSS 代码的编写，利用 CSS 对样式进行修饰，采用 CSS 中的盒子模型进行布局，美化界面。

最后利用 JavaScript 对用户名等进行验证，例如验证输入是否为空，输入是否符合要求。

#### (3) 程序源码及说明

```
Login.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">
```

```

<title>用户登录</title>

<link          rel="stylesheet"          type="text/css"
href="css/style.css">

</head>

<body class= "bg" >

<div class="container_login">

    <div style="float: left;margin: 15px;" >

        <p style="color: #20B2AA;font-size: 20px;">
记账本小程序用户登录</p>

        <p style=" color:  #808080; font-size:
20px;">USER LOGIN</p>

    </div>

    <div style=" float:left;width: 450px;">

        <div>

            <form>

                <table>

                    <tr style="padding-top: 100px">

                        <td class="td_1" ><label
for="username">用户名</label></td>

                        <td class="td_2"><input
type="text"          name="username"          id="username"
placeholder="UserName"></td>

                    </tr>

```

```

                <tr>

                    <td                class="td_1"><label
for="password">密码</label></td>

                    <td                class="td_2"><input
type="password"        name="password"        id="password"
placeholder="Password"></td>

                </tr>

                <tr>

                    <td                colspan="2"
align="center"><input  type="submit"  value=" 登 录  "
id="submit"></td>

                </tr>

            </table>

        </form>

    </div>

</div>

    <div style=" float: right; margin: 15px;">

        <p style="font-size: 20px;">没有账号? <a
href="register.html" style=" color: lightblue;">注册新
账号</a></p>

    </div>

```

```
</div>

</body>

</html>

Register.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>用户注册</title>

<link          rel="stylesheet"          type="text/css"
href="css/style.css">

</head>

<body class= "bg" >

<div class="container">

    <div style="float: left;margin: 15px;" >

        <p style="color: #20B2AA;font-size: 20px;">
记账本小程序新用户注册</p>

        <p style="color: #808080; font-size:
20px;">USER REGISTER</p>

    </div>

    <div style=" float:left;width: 450px;">

        <div>

            <form>
```

```
<table>

    <tr>

        <td            class="td_1"><label
for="username">用户名</label></td>

        <td            class="td_2"><input
type="text"            name="username"            id="username"
placeholder="UserName"></td>

    </tr>

    <tr>

        <td            class="td_1"><label
for="password">密码</label></td>

        <td            class="td_2"><input
type="password"        name="password"        id="password"
placeholder="Password"></td>

    </tr>

    <tr>

        <td            class="td_1"><label
for="email">Email</label></td>

        <td            class="td_2"><input
type="email"           name="email"           id="email"
placeholder="E-mail"></td>

    </tr>
```



```
name="gender" value="female"> 女
    </td>
</tr>

<tr>
    <td class="td_1"><label
for="birthday">出生日期</label></td>
    <td class="td_2"><input
type="date" name="birthday" id="birthday"
placeholder="Birthday"></td>
</tr>
<tr>
    <td colspan="2"
align="center"><input type="submit" value=" 注册 "
id="submit"></td>
</tr>
</table>
</form>
</div>
</div>

<div style=" float: right; margin: 15px;">
    <p style="font-size: 20px;">已有帐号? <a
href="login.html" style=" color: lightblue;">立即登录
```



```
</a></p>
```

```
</div>
```

```
</div>
```

```
</body>
```

```
</html>
```

Style.css

```
*{
```

```
    margin: 0px;
```

```
    padding: 0px;
```

```
    box-sizing: border-box;
```

```
}
```

```
.bg {
```

```
    background:          url('../images/background.jpg')
```

```
no-repeat;
```

```
    background-size: cover;
```

```
    position: absolute;
```

```
}
```

```
.container{
```

```
    width: 620px;
```

```
    height: 670px;
```

```
    background-color: #FFFFFF0 ;
```

```
        border: 10px solid #EEEEEE;  
        margin-top: 80px;  
        margin-left: 460px;  
    }
```

```
.container_login{  
    width: 620px;  
    height: 400px;  
    background-color: #FFFFFF0 ;  
    border: 10px solid #EEEEEE;  
    margin-top: 150px;  
    margin-left: 460px;  
}
```

```
.td_1{  
    width: 100px;  
    text-align: right;  
    height: 45px;  
    font-size: 20px;  
    padding-left: 20px;  
    padding-left: 20px;  
}
```

```
.td_2{
```

```
        padding-top: 10px;

        padding-left: 50px;
    }

#username,#password,#email,#name,#tel,#birthday{

    width: 280px;

    height: 50px;

    border: 2px solid #FFE4E1;

    border-radius: 5px;

    padding-left: 20px;

    font-size: 15px;

}

#submit{

    width: 150px;

    height: 70px;

    border-radius: 15px;

    background-color: #FFDEAD;

    border:4px solid #9ACD32;

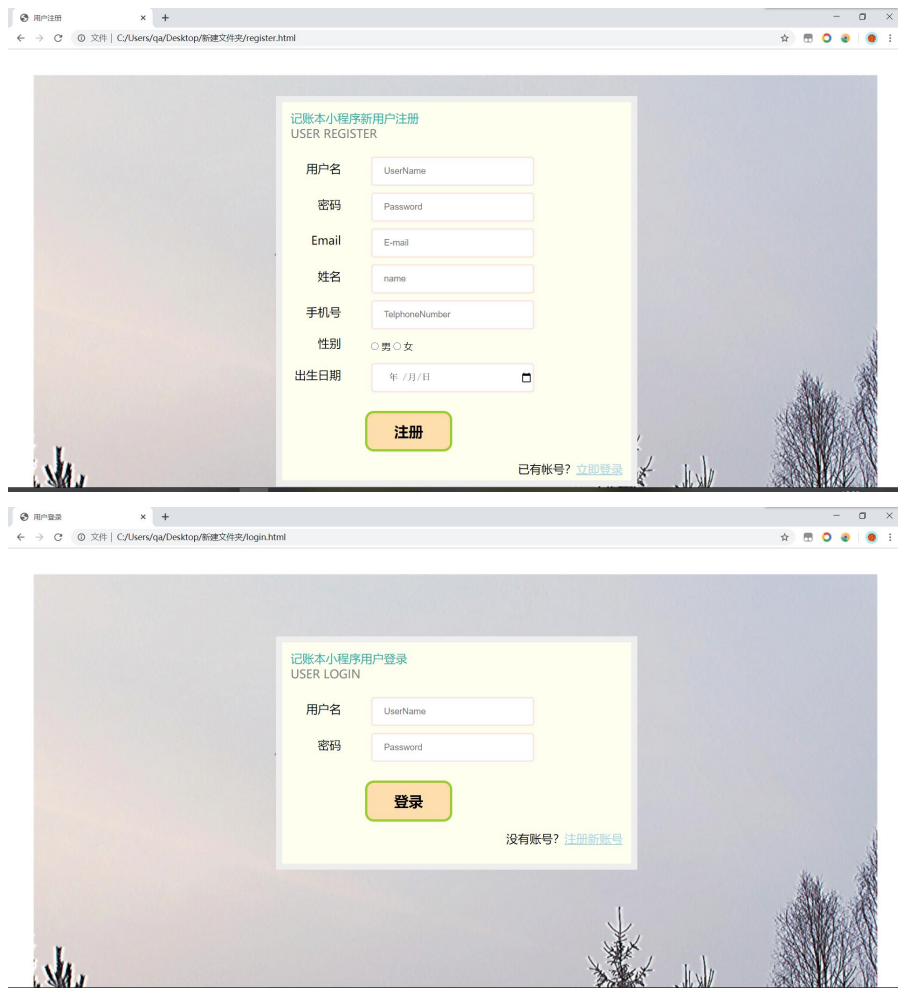
    margin-top: 30px;

    font-size: 25px;

    font-weight: bold;

}
```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

通过本次实验学会并且掌握了 HTML 页面的编写,可以熟练编写 HTML 页面,并且熟练运用 CSS 技术进行界面的美化。

由于接触 HTML、CSS、JavaScript 的时间较短,对于前端页面的编写还不是很熟练,编写出的页面不够美观,UI 设计效果较差,这是需要在后期多加练习来提高的,并且希望在后期学习前端框架,例如:bootstrap、vue 等进行前端更深入的学习。

## 实验 8 JSP 技术的基础应用以及程序设计

### 8.1 实验原理和要求

JSP 是一个动态网页，之前编写的 HTML 页面属于静态网页，如果想要修改静态网页的效果和数据只能修改源代码，而动态网页可以在运行时根据一些条件来修改网页的效果和数据，动态网页和用户是有交互的。

JSP 可以认为是能嵌入 JAVA 代码的网页，JSP 程序中的绝大部分标签是以<%开始，以%>结束的，被标签包围的部分称为 JSP 元素的内容。开始标签、结束标签和元素内容组成 JSP 元素。主要的 JSP 元素有脚本元素、指令元素和动作元素。

通过本次实验了解并且掌握 JSP 的运行原理，学会如何利用 JSP 来编写动态网页，熟练掌握 JSP 页面的编写。

### 8.2 实验任务

#### 8.2.1 任务一

##### (1) 任务内容

在之前编写的记账本项目的基础上，利用 jsp 完成用户的注册和登录功能。

##### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先进行 JSP 页面的编写，对于 JSP 页面的编写，要注意 JSP 语法格式，注意在 JSP 中嵌入 Java 代码的写法，编写注册、登录的两个 JSP 页面。

编写完 JSP 页面的时候，要注意在 JSP 页面中加上对输入框的验证，例如输入框不能为空，输入不符合要求等，这些都需要在 JSP 页面中的 JavaScript 中进行验证，只有符合要求才能提交到后端进行在数据库中的进一步操作。

编写用户的数据库操作层代码，业务逻辑层代码以及 servlet 层，编写完整的 MVC 模式代码，采取面向接口编程的方法。

在对于用户注册的时候对于输入的用户名进行异步处理，利用 ajax 技术验证用户名是否已经存在。

### (3) 程序源码及说明

```
//Login.jsp
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>用户登录</title>
<link rel="stylesheet" type="text/css"
href="css/style1.css">
<script type="text/javascript">
function changeR(node) {
    //用于点击时长生不同的验证码
    node.src = "randomcode.jpg?time="+new Date().getTime();
}

function checkusername()
{
    var myform=document.getElementById("form1");
    var username = myform.username.value;
    if(username.length==0){
        alert("用户名不能为空")
        return false;
    }
    else{
        return true;
    }
}

function checkpassword(){
    var myform=document.getElementById("form1");
    var password=myform.password.value;
    if(password.length==0){
        alert("密码不能为空");
        return false;
    }
}
```

```

        else
            return true;
    }

    function check()
    {
        if(!checkusername())
            return false;
        else if(!checkpassword()) {
            return false;
        }
        else
            return true;
    }
}

</script>
</head>
<body class= "bg" >
<div class="container_login">
    <div style="float: left;margin: 15px;" >
        <p style="color: #20B2AA;font-size: 20px;">记账
        本小程序用户登录</p>
        <p style=" color: #808080; font-size: 20px;">USER
        LOGIN</p>
    </div>
    <div style=" float:left;width: 450px;">
        <div>
            <form id="form1" action="check.jsp"
            method="post" >
                <table>
                    <tr style="padding-top: 100px">
                        <td class="td_1" ><label
                        for="username">用户名</label></td>
                        <td class="td_2"><input
                        type="text" name="username" id="username"
                        placeholder="UserName"></td>

```

```

        </tr>
        <tr>
            <td class="td_1"><label
for="password">密码</label></td>
            <td class="td_2"><input
type="password" name="password" id="password"
placeholder="Password"></td>
        </tr>
        <tr>
            <td class="td_1">验证码</td>
            <td ><a class="td_2">
                <input style="height: 50px;width:
135px;" type="text" name="r" id="r">
            </a>
        </td>
        </tr>
        <tr>
            <td colspan="2"
align="center"><input type="submit" value="登录"
id="submit"></td>
        </tr>
    </table>
</form>
</div>
</div>
<div style=" float: right; margin: 15px;">
    <p style="font-size: 20px;">没有账号? <a
href="register.jsp" style=" color: Lightblue;">注册新账号
</a></p>
</div>
</div>
</body>
</html>
//Register.jsp
<%@ page language="java" contentType="text/html;

```



```
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">

<title>用户注册</title>
<link rel="stylesheet" type="text/css"
href="css/style1.css">
<script type="text/javascript"
src="js/registervalidate.js"></script>
<script type="text/javascript">
    // 定义一个全局的 XMLHttpRequest 对象
    var xhr = false;
    // 创建 XMLHttpRequest 对象
    function createXHR() {
        try {
            // 适用于 IE7+, Firefox, Chrome, Opera, Safari
            xhr = new XMLHttpRequest();
        } catch (e) {
            try {
                // 适用于 IE6, IE5
                xhr = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e1) {
                xhr = false;
            }
        }
        if (!xhr)
            alert("初始化 XMLHttpRequest 对象失败! ");
    }
    // 进行 Ajax 请求和响应结果处理
    function CheckUserName(obj) {
        // 创建 XMLHttpRequest 对象
        createXHR();
        // 获取请求数据
        var username = obj.value;
```

```

        if(username=="||username==null)

document.getElementById("span01").innerHTML="<font
color='red'>用户名不能为空! </font>";

        else{
            // 设定请求地址
            var url = "CheckUserNameServlet?name=" +username;
            // 建立对服务器的调用
            xhr.open("GET", url, true);
            // 指定响应事件处理函数
            xhr.onreadystatechange = function() {
                // 当 readyState 等于 4 且状态为 200 时,表示响应已
就绪
                if (xhr.readyState == 4 && xhr.status == 200) {
                    // 对响应结果进行处理
                    var responseData = xhr.responseText;
                    // 将响应数据更新到页面控件中显示
                    if(responseData==1){

document.getElementById("span01").innerHTML="<font
color='red'>用户名已存在! </font>";

                        }else{

document.getElementById("span01").innerHTML="<font
color='green'>√</font>";

                        }
                    }
                };
            // 向服务器发出请求
            xhr.send(null);
        }
    }
</script>
</head>
<body class= "bg" >
<div class="container">
    <div style="float: left;margin: 15px;" >

```

```

        <p style="color: #20B2AA;font-size: 20px;">记账
    本小程序新用户注册</p>
        <p style=" color: #808080; font-size: 20px;">USER
    REGISTER</p>
    </div>

    <div style=" float:left;width: 450px;">
        <div>
            <form id="form1" action="AddUserServlet"
    method="post" onsubmit="return check()">
                <table>
                    <tr>
                        <td class="td_1"><label
    for="username">用户名</label></td>
                        <td class="td_2"><input
    type="text" name="username" id="username"
    placeholder="UserName" onblur="CheckUserName(this)"><span
    id="span01">(用户名长度必须为 4 到 16 位（字母，数字，下划线，减
    号）)</span></td>

                    </tr>

                    <tr>
                        <td class="td_1"><label
    for="password">密码</label></td>
                        <td class="td_2"><input
    type="password" name="password" id="password"
    placeholder="Password">(最少 6 位，包括至少 1 个大写字母，1 个小
    写字母，1 个数字，1 个特殊字符)</td>

                    </tr>

                    <tr>
                        <td class="td_1"><label
    for="email">Email</label></td>
                        <td class="td_2"><input
    type="email" name="email" id="email"
    placeholder="E-mail"></td>
                    </tr>
                </table>
            </form>
        </div>
    </div>

```

```
        </tr>

        <tr>
            <td class="td_1"><label
for="name">姓名</label></td>
            <td class="td_2"><input
type="text" name="name" id="name" placeholder="name"></td>
        </tr>

        <tr>
            <td class="td_1"><label
for="tel">手机号</label></td>
            <td class="td_2"><input
type="text" name="tel" id="tel"
placeholder="TelephoneNumber"></td>
        </tr>

        <tr>
            <td class="td_1"><label>性别
</label></td>
            <td class="td_2">
                <input type="radio"
name="gender" value="male"> 男
                <input type="radio"
name="gender" value="female"> 女
            </td>
        </tr>

        <tr>
            <td class="td_1"><label
for="birthday">出生日期</label></td>
            <td class="td_2"><input
type="date" name="birthday" id="birthday"
placeholder="Birthday"></td>
        </tr>

        <tr>
            <td colspan="2">
```

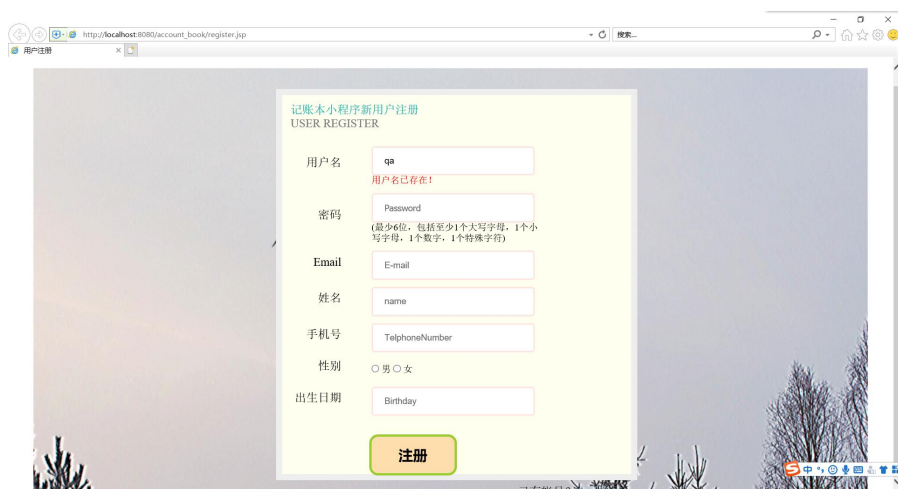
```

align="center"><input type="submit" value="注册"
id="submit"></td>

    </tr>
  </table>
</form>
</div>
</div>
<div style=" float: right; margin: 15px;">
  <p style="font-size: 20px;">已有帐号? <a
href="login.jsp" style=" color: lightblue;">立即登录</a></p>
</div>
</div>
</body>
</html>

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

通过本次实验，了解并且掌握了 JSP 页面编写的基础模式，熟练掌握了编写 JSP 页面，并且进行一套完整的用户注册登录的设计，熟练掌握了 MVC 三层模式实现对数据库增删改查的编写。

在编写 JSP 页面时，对于 Java 代码的嵌入开始的时候使用不顺畅，因为在 HTML 中嵌入 Java 代码会造成一定的混乱，在多次编码练习后熟练掌握了 JSP 页面的编写。

在编写 MVC 模式的三层架构对用户进行增删改查操作的时候，开始

的时候编写比较困难，后来逐渐熟悉了这套流程后，发现其实没有很难，差不多就是一个固定的套路。

在完成注册登录功能后，发现假如用户注册的用户名在数据库中已经有了用户会注册失败，重新注册，但是用户就需要重新输入一次所有的注册信息，这个地方对用户非常不友好，于是查阅相关资料，最终采用 ajax 技术异步处理，对用户名是否存在进行验证，只要用户输入用户名就能立即判断是否存在此用户名，加强了程序的用户友好性。

## 实验 9 基于 Servlet 的程序设计

### 9.1 实验原理和要求

使用 JavaScript 代码和一些脚本语言也是可以登录一些网站，一个网站如果没有验证码，只需要编写一段脚本就可以无限次数的登陆某个网站，这样无数次的尝试就可以暴力破解用户的密码，如果是注册行为那就会给网站制造很多垃圾信息，这个就会对该网站造成极大的资源浪费，严重的可能会让这个网站崩溃，所以就有了验证码。验证码就是用来判断是人在操作还是机器在操作。

一个完整的验证码流程是：用户打开网页显示服务端生成的验证码，点击“看不清楚”标签可以重新生成，这个时候会重新请求服务端数据，服务端用 Session 来保存验证码信息。当用户点击确认按钮的时候，需要对用户通过表单提交的验证码进行校验，服务端获取 Session 保存的验证码信息和用户提交的验证码数据进行校验如果两者一致则校验通过。

验证码的流程总结为：前端表单登陆 => 后端获取到验证码校验 => 前端收到后端的响应。

对于验证码的编写可以自己编写进行画图展示验证码，也可以利用 kaptcha 组件进行编写，一般开发中都采用 kaptcha 进行验证码的编写。通过配置 kaptcha 的配置文件可以设置图片边框，边框颜色，中文验证码等。

通过本次实验学会使用 servlet 进行程序的编写，学会如何使用 kaptcha 进行验证码的编写，并且进行验证码的校验，掌握 servlet 的运行过程，可以熟练编写 servlet 完成所要求。

### 9.2 实验任务

#### 9.2.1 任务一

##### （1）任务内容

在之前的项目的基础上，在登录界面开发验证码功能，在前端显示验证码，并且可以点击图片更新验证码，在后端进行验证码的校验功能。

##### （2）构思过程(可用文字、流程图、UML 图等方式表达)

对于验证码的编写有两种方法，一种是自己编写画图的代码，自己画出验证码并且增加干扰线进行验证码的显示，另一种是直接利用

kaptcha 进行验证码的编写，一般在实际开发中，利用 kaptcha 编写更加的方便，所以本次实验中，我采用 kaptcha 进行验证码的编写。

首先需要在 web.xml 文件进行配置，其中不仅需要配置好对应的 servlet 以及产生的图片，还可以配置一系列的参数，例如验证码的长度和宽度、验证码上显示的字符、干扰线的颜色等等。

接着需要进行 servlet，此步骤需要运用到图像类进行验证码的绘制，当然由于在配置文件中已经配置好了验证码图片的属性，这步相对于自己手动编写验证码较为方便。

最后在前端显示出验证码，前端编写 JavaScript 脚本实现点击图片更新验证码的功能。对验证码进行验证，在验证验证码的时候，获取前端文本框的数据和已经在 session 中存好的验证码正确值进行对比，正确则验证用户名密码是否正确，不正确弹出警告框。

### （3）程序源码及说明

```
//在 web.xml 中配置 kaptcha
<servlet>
    <servlet-name>kaptcha</servlet-name>
    <servlet-class>com.google.code.kaptcha.servlet.KaptchaServlet
</servlet-class>

    <init-param>
        <description>图片边框，合法值: yes, no</description>
        <param-name>kaptcha.border</param-name>
        <param-value>yes</param-value>
    </init-param>

    <!-- 图片宽度 -->
    <init-param>
        <param-name>kaptcha.image.width</param-name>
        <param-value>135</param-value>
    </init-param>

    <!-- 图片高度 -->
    <init-param>
        <param-name>kaptcha.image.height</param-name>
        <param-value>50</param-value>
    </init-param>

    <!-- 使用哪些字符生成验证码 -->
```



```

        <init-param>
            <param-name>kaptcha.textproducer.char.string</param-name>

<param-value>ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345679
            </param-value>
        </init-param>
        <!-- 字符个数 -->
        <init-param>
            <param-name>kaptcha.textproducer.char.length</param-name>
            <param-value>4</param-value>
        </init-param>
        <!-- 字体大小 -->
        <init-param>
            <param-name>kaptcha.textproducer.font.size</param-name>
            <param-value>43</param-value>
        </init-param>
        <!-- 干扰线的颜色 -->
        <init-param>
            <param-name>kaptcha.noise.color</param-name>
            <param-value>yellow</param-value>
        </init-param>
        <!-- 使用哪些字体 -->
        <init-param>
            <param-name>kaptcha.textproducer.font.names</param-name>
            <param-value>Arial</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>kaptcha</servlet-name>
        <url-pattern>/randomcode.jpg</url-pattern>
    </servlet-mapping>

//编写 KaptchaServlet

package com.user.servlet;

```

```
import com.google.code.kaptcha.Producer;

import com.google.code.kaptcha.util.Config;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.util.Enumeration;

import java.util.Properties;

import javax.imageio.ImageIO;

import javax.servlet.Servlet;

import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/KaptchaServlet")

public class KaptchaServlet extends HttpServlet implements Servlet {

    private static final long serialVersionUID = 5021890762875428174L;

    private Properties props;

    private Producer kaptchaProducer;

    private String sessionKeyValue;

    public KaptchaServlet() {

        this.props = new Properties();

        this.kaptchaProducer = null;

    }

}
```

```
        this.sessionKeyValue = null;

    }

    public void init(ServletConfig conf) throws ServletException {
super.init(conf);

        ImageIO.setUseCache(false);

        Enumeration initParams = conf.getInitParameterNames();

        while (initParams.hasMoreElements()) {

            String key = (String) initParams.nextElement();

            String value = conf.getInitParameter(key);

            this.props.put(key, value);

        }

        Config config = new Config(this.props);

        this.kaptchaProducer = config.getProducerImpl();

        this.sessionKeyValue = config.getSessionKey();

    }

    public void doGet(HttpServletRequest req, HttpServletResponse resp)

        throws ServletException, IOException {

        resp.setDateHeader("Expires", 0L);

        resp.setHeader("Cache-Control", "no-store, no-cache, must-revalidate");

        resp.addHeader("Cache-Control", "post-check=0, pre-check=0");

        resp.setHeader("Pragma", "no-cache");

        resp.setContentType("image/jpeg");

    }

}
```

```

        String capText = this.kaptchaProducer.createText();

        String s1 = capText.substring(0, 1);

        String s2 = capText.substring(1, 2);

        int r = Integer.valueOf(s1).intValue() + Integer.valueOf(s2).intValue();

        req.getSession().setAttribute(this.sessionKeyValue, String.valueOf(r));

        BufferedImage bi = this.kaptchaProducer.createImage(s1+" "+s2+"=?");

        ServletOutputStream out = resp.getOutputStream();

        ImageIO.write(bi, "jpg", out);

        try {

            out.flush();

        } finally {

            out.close();

        }

    }

}

//检查是否是正确的验证码
//校验验证码  check.jsp
String k =
(String)session.getAttribute(com.google.code.kaptcha.Constants.KAPTCHA_SESSION_KEY);
System.out.print(k);

String str = request.getParameter("r");

String username=request.getParameter("username");

String password=request.getParameter("password");

if(username.length()==0){

    response.getWriter().write("<script language=javascript>alert('用户名不能为
空');window.location='login.jsp'</script>");

```

```

    }

    else{
        if(password.length()==0){
            response.getWriter().write("<script language=javascript>alert('密码不能为空');window.location='login.jsp'</script>");
        }
        else{
            if(!k.equals(str)){
                response.getWriter().write("<script language=javascript>alert('验证码错误');window.location='login.jsp'</script>");
            }
            else{
                session.setAttribute("username",username);
                session.setAttribute("password", password);
                response.sendRedirect("ValidateUserServlet");
            }
        }
    }
}

```

//前端显示验证码以及刷新验证码

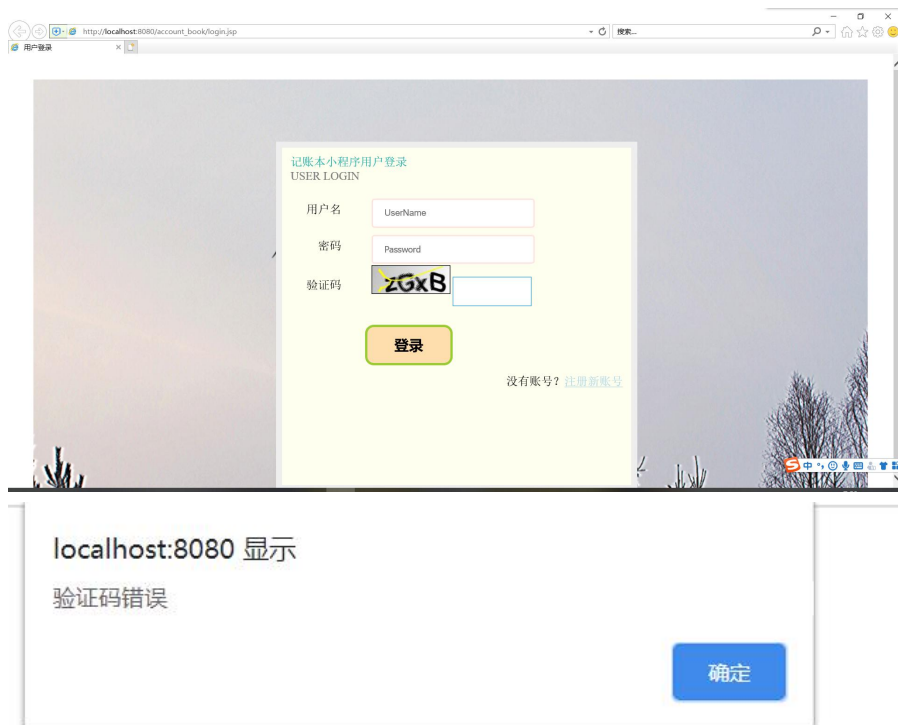
```

function changeR(node) {
    //用于点击时长生不同的验证码
    node.src = "randomcode.jpg?time="+new Date().getTime();
}

<tr>
    <td class="td_1">验证码</td>
    <td ><a class="td_2">
    <input style="height: 50px;width: 135px;" type="text" name="r" id="r">
    </a>
    </td>
</tr>

```

#### (4) 运行结果(截图)



#### (5) 心得体会（问题调试及体会）

通过本次实验，了解并且掌握了 servlet 进行程序设计的一个基本的操作，学会了使用 kaptcha 组件进行验证码的编写，并且学会在后端进行验证码的验证，返回给前端响应。了解了 servlet 运行的一个机制，接受前端的请求，返回给前端响应。

在利用 kaptcha 进行验证码的编写时，开始的时候配置文件中的映射写错了导致不能显示图片，这个 bug 的寻找花费了大量的时间，还有就是对于其中的一些配置参数理解不深，导致出错。

在对于后端的验证时需要获取 session 中正确的验证码的值与前端获取的用户输入进行对比，这一步要注意获取 session 中的属性时属性名的正确拼写。

## 实验 10 JavaBean 程序

### 10.1 实验原理和要求

JavaBean 是特殊的 Java 类,使用 Java 语言书写,并且遵守 JavaBean API 规范, JavaBean 与其他 Java 类相比而言具有以下特征: 提供一个默认的空参构造函数, 需要被序列化并且实现了 `Serializable` 接口, 可能有一系列可读写属性, 可能有一系列的 `getter` 或 `setter` 方法。

一个 JavaBean 对象的属性应该是可访问的, 这个属性可以是任意合法的 Java 数据类型, 包括自定义 Java 类。JavaBean 对象的属性可以通过 JavaBean 实现类中提供的两个方法来访问 `getPropertyName()` 和 `setPropertyName()`。

要求通过本次实验, 了解 JavaBean 组件在 JSP 中的应用, 掌握利用 JDBC 访问数据库, 在 JavaBean 中用 jdbc 方式进行数据库数据的处理。

### 10.2 实验任务

#### 10.2.1 任务一

##### (1) 任务内容

在之前的记账本项目的基础上, 开发 JavaBean, 使用这个 JavaBean 实现记账本记账显示列表的分页显示, 每页的数量可以设定。增加记账备注栏, 要求支持 200 字以内的备注信息 (超过字数要有警告提示), 并且实现记账的详情显示页。

##### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

首先思考如何来编写这个 JavaBean, 这个 JavaBean 的作用是什么以及其中有哪些属性。

对于分页功能, 首先来编写 dao 层的代码, dao 层主要负责对于数据库的操作, 也就是利用 jdbc 进行对数据库的增删改查, 对于不同的数据库来说, 分页查询的 SQL 语句有一定的区别, 本实验使用的是 mysql 数据库, 只需要利用 `limit` 就可以完成分页查询, 但是要注意 `limit` 后面的两个参数, 一个是当前页码数, 一个是每一页显示的数据数, 并且要注意页码数是从 0 开始的。

在编写完 dao 层的代码后, 编写业务逻辑层 service 层的代码接着编写 servlet 进行与前端的数据交互。在 servlet 中获取查询到的当前

页的 pagebean，并且存到 session 域中，方便返回前端进行数据显示。

在编写前端 jsp 的代码时，当用户改变当前页面显示数据数量的时候，要进行一个页面的刷新，对于这一点，采用 jQuery 实现，当改变选择框的内容时就进行刷新，并且利用 jQuery 实时更新 select 框中默认选择项。而对于前端显示数据则可以采用 JSTL 和 EL 进行对 session 域中的 pagebean 进行访问，展示数据。

对于账单详情页的编写，只需要更改原有的 HomeCost，在里面新加一个 note 属性作为每一条账单的备注，并且注意修改数据库中的表结构，以及相关代码。

### (3) 程序源码及说明

```
//pagebean
package com.cost.bean;
import java.util.List;
public class pagebean {
    private int currentPage;
    private int pageSize;
    private int totalCount;
    @Override
    public String toString() {
        return "pagebean [currentPage=" + currentPage + ", pageSize="
+ pageSize + ", totalCount=" + totalCount
        + ", totalPage=" + totalPage + ", homeCosts=" + homeCosts
+ "]);
    }
    public pagebean(int currentPage, int pageSize, int totalCount, int
totalPage, List<HomeCost> homeCosts) {
        super();
        this.currentPage = currentPage;
        this.pageSize = pageSize;
        this.totalCount = totalCount;
        this.totalPage = totalPage;
        this.homeCosts = homeCosts;
    }
    public pagebean() {
        super();
    }
}
```



```

    public int getCurrentPage() {
        return currentPage;
    }
    public void setCurrentPage(int currentPage) {
        this.currentPage = currentPage;
    }
    public int getPageSize() {
        return pageSize;
    }
    public void setPageSize(int pageSize) {
        this.pageSize = pageSize;
        this.totalPage=this.totalCount%this.pageSize==0?this.totalCount
/this.pageSize:this.totalCount/this.pageSize+1;
    }
    public int getTotalCount() {
        return totalCount;
    }
    public void setTotalCount(int totalCount) {
        this.totalCount = totalCount;
    }
    public int getTotalPage() {
        return totalPage;
    }
    public void setTotalPage(int totalPage) {
        this.totalPage = totalPage;
    }
    public List<HomeCost> getHomeCosts() {
        return homeCosts;
    }
    public void setHomeCosts(List<HomeCost> homeCosts) {
        this.homeCosts = homeCosts;
    }
    private int totalPage;
    private List<HomeCost> homeCosts;
}

```

//对数据库中的数据进行分页查询

```

public List<HomeCost> queryCostsByPage(int currentPage,int

```

```

pageSize,String username) {
    int start=pageSize*(currentPage-1);
    String sql = "select * from home "+"where username = 
    '"+username+"' limit "+start+","+pageSize;
    List<HomeCost> list = new ArrayList<>();
    Connection conn = DBUtils.getConn();
    Statement state = null;
    ResultSet rs = null;
    try {
        state = conn.createStatement();
        rs = state.executeQuery(sql);
        while (rs.next()) {
            int id = rs.getInt("id");//获取查询结果中的 id
            BigDecimal sum = queryMoneySum(id,username);
            String name = rs.getString("name");//获取查询结果中的
name
            BigDecimal money = rs.getBigDecimal("money");//获取查
询结果中的 money
            String date = rs.getString("date");//获取查询结果中的
date
            String note=rs.getString("note");
            HomeCost homeCost = new
HomeCost(id,name,money,date,sum,note);//调用构造方法赋值
            list.add(homeCost);//添加到 list 集合中
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        DBUtils.close(rs, state, conn);
    }
    return list;
}

```

**//QueryByPageServlet**

```

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HomeCostService homeCostService = new HomeCostService();

```

```

        request.getSession().setAttribute("username", "qa");
        String
username=(String)request.getSession().getAttribute("username");
        System.out.println(username);
        int count=homeCostService.getTotalCount(username);

        pagebean pagebean=new pagebean();
        String cpage=request.getParameter("currentPage");
        if(cpage==null)
            cpage="1";
        int currentPage=Integer.parseInt(cpage);
        pagebean.setCurrentPage(currentPage);
        pagebean.setTotalCount(count);
        String size=(String)request.getParameter("pagesize");
        if(size==null||size=="")
            size="5";
        int pageSize=Integer.parseInt(size);
        pagebean.setPageSize(pageSize);
        List<HomeCost>
costs=homeCostService.queryCostsByPage(currentPage,
pageSize,username);
        pagebean.setHomeCosts(costs);
        request.setAttribute("page", pagebean);
        request.getRequestDispatcher("manager.jsp").forward(request,
response);
    }

```

//前端 jsp 页面核心代码

```

        <td><a
href="QueryCostsByPageServlet?currentPage=1&pagesize=${pageSize}">
首页</a></td>

        <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getCurrentPag
e()-1%>&pagesize=${pageSize}">上一页</a></td>

        <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getCurrentPag
e()+1%>&pagesize=${pageSize}">下一页</a></td>

```

```

<td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getTotalPage(
)%>&pagesize=${pageSize}">尾页</a></td>
<select name="pageSize" id="pageSizeSelect">
    <option value="5">5 条/页</option>
    <option value="10">10 条/页</option>
    <option value="15">15 条/页</option>
</select>
<tr>
    <td colspan="4">共有
    ${requestScope.page.getTotalCount()}笔消费记录</td>
    <td>当前页数: <%=pagebean.getCurrentPage() %> </td>
    <td>总共有<%=pagebean.getTotalPage() %> 页</td>
</tr>

<script>
$( "#pageSizeSelect" ).change( function () {
    var pageSize = this.value;
    location.href =
    "QueryCostsByPageServlet?currentPage=1&pagesize="+pageSize;
});
$(function () {
    $( "#pageSizeSelect" ).val( ${requestScope.page.getPageSize()} );
});
</script>

```

#### (4) 运行结果(截图)



家庭记账本

新增消费记录  清除

消费名称	消费金额	累计消费	登记日期	操作		
粥	15.00	15.00	2020-06-13 16:22:39	修改	详情	删除
娱乐	150.00	165.00	2020-06-13 16:22:49	修改	详情	删除
书费	500.00	665.00	2020-06-13 16:22:57	修改	详情	删除
晚饭	100.00	765.00	2020-06-14 21:13:02	修改	详情	删除
午饭	500.00	1265.00	2020-06-14 22:17:33	修改	详情	删除
新增						
首页	上一页	下一页	尾页	5 条/页		
共有7笔消费记录				当前页数: 1 总共有2 页		



请输入关键字

共有7笔消费记录

当前页数: 1 总共有1 页



请输入关键字  查询

消费名称	消费金额	登记日期
午餐	500.00	2020-06-14 22:17:33
备注	有羔羊、羔鹌鹑、羔鸡呢儿、烧花肉、烧鸡、烧子鸡、卤猪、卤鸡、酱肉、腊肉、松花小肚儿、奶汤、葱丝、什锦豆腐、麻肉豆儿儿、卤菜八宝猪、玉米糁子粥、罐儿虾肉、罐儿猪肉、合什饼、凉菜、山药、兔腿、菜饼、银鱼、清蒸鲑子。	



请输入关键字  查询

消费名称	消费金额	登记日期	操作
早餐	20	日期系统自动录入	

俗话说：“早餐要吃好，午餐要吃饱，晚餐要吃少。”为什么要这么安排呢？每当我们吃完饭后，大约经过4个小时，食物便在体内的消化道，将全部吸收。因此，为了补给人体补充能量，必须4~6小时安排一次用餐。早晨，当我们经过6个小时的睡眠后，会感到特别的精神，自然上午的工作、学习效率都是比较高的。但是，许多人为为了赶时间，就把早餐“省略”了，其实，这是一个很不

### （5）心得体会（问题调试及体会）

通过本次实验，了解并且掌握了 Javabean 的实验，能够熟练编写 Javabean 并且运用 Javabean 解决实际的问题，学会了使用 Javabean 来对数据库进行操作，并且能够熟练使用 Javabean 里自带的两个方法操作 Javabean。

通过本次实验还掌握了如何实现分页功能，首先复习了相关 SQL 语句，知道如何写出查询出当前页的数据的 SQL 语句，对于 mysql 来说直接使用 limit 查询就可以，而对于 oracle 数据库或者 SqlServer 数据库则比较复杂需要建立索引列来实现查询，还注意到对于 MySQL 来说，页码的编号是从 0 开始的，这点在开始并没有注意到，导致查询出来的数据是错误的。

对于分页需要的数据进行一个封装，封装成一个 Javabean，在编写完 pagebean 的 Javabean 后，在编写完 dao 层的代码后，接着编写 service 层的代码，最后编写 servlet，在 servlet 中与前端数据进行交互。对于 MVC 的设计模式有了更加深刻的认识。

在本次任务进行的过程中，出现了很多的 bug，通过本次实验，逐渐掌握了通过查看控制台中的错误语句修复 bug，虽然整个过程比较艰辛，但是通过这个过程学到了很多知识，自己的代码水平得到了很大的提升。

## 实验 11 标签与自定义标签

### 11.1 实验原理和要求

JSTL 的中文全称是 JSP 标准标识库。JSTL 标签是基于 JSP 页面的，这些标签可以插入到 JSP 代码中本质上 JSTL 也是提前定义好的一组标签，这些标签封装了不同的功能，在页面上调用标签时，就等于调用了封装起来的功能。JSTL 的目标是简化 JSP 页面的设计。对于页面设计人员来说，使用脚本语言操作动态数据是比较困难的，而采用标签和表达式语言则相对容易，JSTL 的使用为页面设计人员和程序开发人员的分工协作提供了便利。

JSTL 标识库的作用是减少 JSP 文件的 Java 代码，使 Java 代码与 HTML 代码分离，所以 JSTL 标识库符合 MVC 设计理念。MVC 设计理念的优势是将动作控制、数据处理、结果显示三者分离。

通过本次实验掌握核心标签库等标签库的使用，熟练掌握 JSTL 的使用，熟练运用 JSTL，使用 JSTL 对之前完成的项目进行改写。

### 11.2 实验任务

#### 11.2.1 任务一

##### (1) 任务内容

综合运用核心标签、格式化标签、自定义标签，完善前边所实现的记账本系统中的记账信息详情显示页

##### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

对于 JSTL 的使用，首先需要先引入 jar 包，把从官网下载的 jar 包放到 lib 文件夹下才可以使用 JSTL 标签。

而使用的时候必须在 JSP 的头部文件中引入，这样才可以在程序中使用 JSTL 标签。

利用 JSTL 中的核心标签库修改记账信息显示的代码，利用 JSTL 中的标签对账单信息进行一个遍历。

##### (3) 程序源码及说明

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:forEach items="${requestScope.page.getHomeCosts()}" var="item">
    <tr>
        <td>${item.name}</td>
```

```

        <td>${item.money}</td>
        <td>${item.sum}</td>
        <td>${item.date}</td>
        <td><a
href="${pageContext.request.contextPath }/manager/homeCostServlet?action=getHomeCostById&id=${item.id}">修改</a></td>
        <td><a
href="${pageContext.request.contextPath }/manager/homeCostServlet?action=getNoteById&id=${item.id}">详情</a></td>
        <td><a class="deleteClass"
href="${pageContext.request.contextPath }/manager/homeCostServlet?action=delete&id=${item.id}">删除</a></td>
    </tr>
</c:forEach>

<%@page import="org.apache.taglibs.standard.lang.jstl.test.Bean1"%>
<%@page import="com.cost.bean.pagebean"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>消费记录管理</title>
    <!-- 采用绝对路径导入 css 文件 -->
    <link rel="stylesheet" type="text/css"
href="${pageContext.request.contextPath }/css/style.css" />
    <!-- 采用绝对路径导入 jquery 文件 -->
    <script type="text/javascript"
src="${pageContext.request.contextPath }/js/jquery-1.7.2.js"></script>
    <script type="text/javascript">

        $(function () {
            //提示用户添加失败，删除失败，修改失败
            if(!${empty requestScope.msg}){
                alert("${requestScope.msg}");
            }
        });
    </script>

```



```

    }
    //验证非空，并提交查询请求
    $("#submit").click(function () {
        //验证输入框是否为空
        var keyword = $("#keyword").val();
        if(keyword == ""){
            alert("请输入关键字");
            return false;
        }else {
            //javascript 语言提供了一个 location 地址栏对象
            //它有一个属性 href, 可以获取浏览器中地址栏地址

            location.href="${pageContext.request.contextPath }/manager/homeCostServ
let?action=query&keyword="+keyword;
        }

    });

    //删除提示
    $("#a.deleteClass").click(function () {

        return confirm("你确定要删除【"+
$(this).parent().parent().find("td:first").text()+"】?");
    });

});

</script>
</head>
<body>
    <div id="header">
        <span class="wel_word">家庭记账本</span>
        <div>
            <a href="${pageContext.request.contextPath }/cost_edit.jsp">新
增消费记录</a>
            <input style="margin-left:20px" id="keyword" name="keyword"
type="text" placeholder="请输入关键字" value="" />

```

```

        <input id="submit" type="submit" value="查询"/>
    </div>
</div>
<div id="main">
    <table id="form1" style="margin-top:30px">
        <tr>
            <td class="costname" style="width:200px">消费名称</td>
            <td>消费金额</td>
            <td>累计消费</td>
            <td style="width:200px">登记日期</td>
            <td colspan="3">操作</td>
        </tr>
        <!-- 使用 el 表达式注意在 jsp 页面(如本页面第一行)导入相应的包 -->
        <c:forEach items="${requestScope.page.getHomeCosts()}"
var="item">
            <tr>
                <td>${item.name}</td>
                <td>${item.money}</td>
                <td>${item.sum}</td>
                <td>${item.date}</td>
                <td><a
href="${pageContext.request.contextPath }/manager/homeCostServlet?action=getHomeCostById&id=${item.id}">修改</a></td>
                <td><a
href="${pageContext.request.contextPath }/manager/homeCostServlet?action=getNoteById&id=${item.id}">详情</a></td>
                <td><a class="deleteClass"
href="${pageContext.request.contextPath }/manager/homeCostServlet?action=delete&id=${item.id}">删除</a></td>
            </tr>
        </c:forEach>
        <tr>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td colspan="3"><a

```

```

href="${pageContext.request.contextPath }/cost_edit.jsp">新增</a></td>
</tr>
<%
pagebean pagebean=(pagebean)request.getAttribute("page");
%>
<tr>
<%
if(pagebean.getTotalPage()==1){
    %>
    <td><a href="#">首页</a></td>
    <td><a href="#">上一页</a></td>
    <td><a href="#">下一页</a></td>
    <td><a href="#">尾页</a></td>
    <%
}
else if(pagebean.getCurrentPage()==pagebean.getTotalPage()){
    %>
    <td><a
href="QueryCostsByPageServlet?currentPage=1&pagesize=${pageSize}">首页
</a></td>
    <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getCurrentPage()-1%>&
pagesize=${pageSize}">上一页</a></td>
    <td><a href="#">下一页</a></td>
    <td><a href="#">尾页</a></td>
    <%
}
else if(pagebean.getCurrentPage()==1)
{
    %>
    <td><a href="#">首页</a></td>
    <td><a href="#">上一页</a></td>
    <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getCurrentPage()+1%>&
pagesize=${pageSize}">下一页</a></td>
    <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getTotalPage()%>&page

```

```

size=${pageSize}">尾页</a></td>

        <%
        }
        else{
        %>

                <td><a
href="QueryCostsByPageServlet?currentPage=1&pageSize=${pageSize}">首页
</a></td>

                <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getCurrentPage()-1%>&
pageSize=${pageSize}">上一页</a></td>

                <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getCurrentPage()+1%>&
pageSize=${pageSize}">下一页</a></td>

                <td><a
href="QueryCostsByPageServlet?currentPage=<%=pagebean.getTotalPage()%>&page
size=${pageSize}">尾页</a></td>

        <%
        }
        %>

        <td colspan="3">
                <select name="pageSize" id="pageSizeSelect">
                        <option value="5">5 条/页</option>
                        <option value="10">10 条/页</option>
                        <option value="15">15 条/页</option>
                </select>
        </td>
</tr>

<tr>
        <td colspan="4">共有${requestScope.page.getTotalCount()}笔
消费记录</td>

        <td>当前页数: <%=pagebean.getCurrentPage() %> </td>
        <td>总共有<%=pagebean.getTotalPage() %> 页</td>

</tr>
</table>
</div>

```

```

</body>
<script>

$("#pageSizeSelect").change(function () {
    var pageSize = this.value;
    location.href =
    "QueryCostsByPageServlet?currentPage=1&pagesize="+pageSize;
});

$(function () {
    $("#pageSizeSelect").val(${{requestScope.page.pageSize}});
});
</script>
</html>

```

#### (4) 运行结果(截图)



消费名称	消费金额	累计消费	登记日期	操作
粥	15.00	15.00	2020-06-13 16:22:39	修改 详情 删除
娱乐	150.00	165.00	2020-06-13 16:22:49	修改 详情 删除
书费	500.00	665.00	2020-06-13 16:22:57	修改 详情 删除
晚饭	100.00	765.00	2020-06-14 21:13:02	修改 详情 删除
午饭	500.00	1265.00	2020-06-14 22:17:33	修改 详情 删除
新增				
首页	上一页	下一页	尾页	5条/页
共有7笔消费记录 当前页数: 1 总共有2 页				

#### 账单详情qa



消费名称	消费金额	登记日期
夜宵	500.00	2020-06-15 17:22:04

备注

城市当中，夜宵跟正餐的选择几乎泾渭分明，夜宵吃得多了一些不啻如吃泡的小吃，例如烧烤、糖水（甜品店），以及小龙虾等，夜宵消费吃泡、海鲜料理等，都是成年人把酒谈天，放松心情或观赛夜场球赛的另一结合。

### （5）心得体会（问题调试及体会）

通过本次实验掌握了 JSTL 的使用，如何利用 JSTL 进行开发，利用 JSTL 可以极大的简化 JSP 页面的编码，极大的方便了程序员的开发。

JSTL 的开发首先得要导入包，一定要记得导包，否则无法使用 JSTL，接着就是要记得在 JSP 页面最上面导入标签，才可以使用 JSTL 进行开发。对于 JSTL 的使用，核心标签库是最常用的，利用核心标签库可以极大的方便程序员开发，使得代码的结构清晰。而对于其他标签库的使用，也可以带来很大的方便，例如对于 sql 标签库的使用可以方便对数据库进行操作。

## 实验 12 JDBC 数据库连接技术及其程序设计

### 12.1 实验原理和要求

JDBC 是 Java 中用户和数据库进行交互的一种技术，要使用 JDBC 首先到导入 JDBC 所需要的包才可以操作，而对于 JDBC 使用的一个大致流程如下：

- (1) 加载驱动，
- (2) 获取连接，
- (3) 获取预编译对象，
- (4) 执行 sql 语句，
- (5) 获取结果，
- (6) 关闭资源，

通过本次实验了解并且掌握 JDBC 的使用，学会使用 JDBC 技术完成和数据库的交互，并且能够完整编写出用户信息的增删改查功能。

### 12.2 实验任务

#### 12.2.1 任务一

##### (1) 任务内容

在记账本系统中制作用户管理功能，实现用户的增删改查，每个用户实现单独的记账功能、修改密码功能，如张三登录后只能看到自己的记账信息，也只能管理自己的记账信息。完善相关功能，如用户列表的分页、密码加密、界面美化等。

##### (2) 构思过程(可用文字、流程图、UML 图等方式表达)

对于用户后台信息管理的前端页面采用 bootstrap 开发，而对于用户信息的管理，实质上是对用户表的增删改查，利用 MVC 三层架构进行编写，大致写法可以仿照前面的记账本的增删改查，内容大同小异。

对于数据库的操作可以对数据库操作的一些方法进行封装，实现代码的重用，方便开发。

对于数据库层以及业务逻辑层而是采用面向接口的编程方式进行编码。

对于其他功能例如分页查询等逐步就行完善。

### (3) 程序源码及说明

```
package com.user.util;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DBUtil {
    private          static          final          String
    URL="jdbc:mysql://localhost:3306/jz?serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8";
    private static final   String Name="com.mysql.cj.jdbc.Driver";
    private static final String username="root";
    private static final String password="12297804qa";
    public static PreparedStatement pstmt = null ;
    public static Connection connection = null ;
    public static ResultSet rs = null ;
    public static boolean executeUpdate(String sql,Object[] params) {
        try {
            pstmt = createPreParedStatement(sql,params);
            int count = pstmt.executeUpdate() ;
            if(count>0)
                return true ;
            else
                return false ;

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return false ;
        } catch (SQLException e) {
            e.printStackTrace();
            return false ;
        } catch (Exception e) {
```



```

        e.printStackTrace();
        return false ;
    }
    finally {
        closeAll(null,pstmt,connection);
    }
}

public static void closeAll(ResultSet rs,Statement stmt,Connection
connection)
{
    try {
        if(rs!=null)rs.close();
        if(pstmt!=null)pstmt.close();
        if(connection!=null)connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

}

public static Connection getConnection() throws ClassNotFoundException,
SQLException {
    Class.forName(Name) ;
    return DriverManager.getConnection( URL,username,password ) ;
}

public static PreparedStatement createPreParedStatement(String sql,Object[]
params) throws ClassNotFoundException, SQLException {
    pstmt = getConnection() .prepareStatement( sql) ;
    if(params!=null ) {
        for(int i=0;i<params.length;i++) {
            pstmt.setObject(i+1, params[i]);
        }
    }
    return pstmt;
}

public static ResultSet executeQuery( String sql ,Object[] params) {
    try {

```

```

        pstmt = createPreParedStatement(sql,params);
        rs = pstmt.executeQuery() ;
        return rs ;
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
        return null ;
    } catch (SQLException e) {
        e.printStackTrace();
        return null ;
    } catch (Exception e) {
        e.printStackTrace();
        return null ;
    }
}
}

```

```

package com.cost.dao;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import com.cost.bean.HomeCost;
import com.cost.util.DBUtils;
public class HomeCostDao {

    /**
     * updatesql()用来执行 insert/update/delete 语句
     * @param sql 具体的 sql 语句
     * @return 返回-1，说明执行失败；否则为影响数据条数
     */
    public int updatesql(String sql) {
        Connection conn = DBUtils.getConn();//获取连接对象
        Statement state = null;
        try {

```

```

        state = conn.createStatement();
        return state.executeUpdate(sql);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        DBUtils.close(state, conn);
    }
    return -1;
}

//添加
public int add(HomeCost homecost,String username) {
    //insert 语句，形如 insert into 表名(字段 1， 字段 2) values('值 1','
    值 2');
    String sql = "insert into home(name,money,note,username)
values('"+ homecost.getName()
        + "','" + homecost.getMoney() +
    "','"+homecost.getNote()+"','"+username+"')";
    return updatesql(sql);
}

//删除
public int delete (int id,String username) {
    //delete 语句，形如 delete from 表名 where id='值';
    String sql = "delete from home where id='" + id + "' and username =
    '"+username+"'";
    return updatesql(sql);
}

//修改
public int update(HomeCost homecost,String username) {
    //update 语句，形如 update 表名 set 字段 1 = '值 1',字段 2 = '值
    2'where id = '值 3';
    String sql = "update home set name='" + homecost.getName() + "',
    money="
        + homecost.getMoney()+
    "','"+homecost.getNote()+"' where id='" + homecost.getId() + "' and
    username = '"+username+"'";

```

```

        return updatesql(sql);

    }

    //查询
    public List<HomeCost> query(String keyword,String username) {
        String sql = "select * from home WHERE name LIKE
        '%" + keyword + "%' OR money LIKE '%" + keyword
            + "%' OR date LIKE '%" + keyword + "%' and username =
        '" + username + "'";
        List<HomeCost> list = new ArrayList<>();
        Connection conn = DBUtils.getConn();
        Statement state = null;
        ResultSet rs = null;
        try {
            state = conn.createStatement();
            rs = state.executeQuery(sql);
            while (rs.next()) {
                int id = rs.getInt("id");//获取查询结果中的 id
                String name = rs.getString("name");//获取查询结果中的
name
                BigDecimal money = rs.getBigDecimal("money");//获取
查询结果中的 money
                String date = rs.getString("date");//获取查询结果中的
date
                HomeCost homeCost = new
HomeCost(id,name,money,date);//调用构造方法赋值
                list.add(homeCost);//添加到 list 集合中
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DBUtils.close(rs, state, conn);
        }
        return list;
    }
}

```

```

//通过 id 找到某条信息
public HomeCost getHomeCostById(int id,String username) {
    String sql = "select * from home where id =" + id + " and username
= ""+username+""";
    Connection conn = DBUtils.getConn();
    Statement state = null;
    ResultSet rs = null;
    HomeCost homeCost = null;
    try {
        state = conn.createStatement();
        rs = state.executeQuery(sql);
        while (rs.next()) {
            String name = rs.getString("name");
            BigDecimal money = rs.getBigDecimal("money");
            String date = rs.getString("date");
            homeCost = new HomeCost(id,name,money,date);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        DBUtils.close(rs, state, conn);
    }
    return homeCost;
}

public HomeCost getNoteById(int id,String username) {
    String sql = "select * from home where id =" + id + " and username
= ""+username+""";
    Connection conn = DBUtils.getConn();
    Statement state = null;
    ResultSet rs = null;
    HomeCost homeCost = null;
    try {
        state = conn.createStatement();
        rs = state.executeQuery(sql);
        while (rs.next()) {
            String name = rs.getString("name");
            BigDecimal money = rs.getBigDecimal("money");

```

```

        String date = rs.getString("date");
        String note=rs.getString("note");

        homeCost = new HomeCost(id,name,money,date,note);

    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    DBUtils.close(rs, state, conn);
}
return homeCost;
}

/**
 * 通过 id 计算该条消费记录累计消费金额
 * @return BigDecimal 类型 money
 */
public BigDecimal queryMoneySum(int id,String username) {
    String sql = "select money from home where id <="+id+" and
username = '"+username+"'";
    BigDecimal sum = new BigDecimal("0.00");
    Connection conn = DBUtils.getConn();
    Statement state = null;
    ResultSet rs = null;
    try {
        state = conn.createStatement();
        rs = state.executeQuery(sql);
        while (rs.next()) {
            BigDecimal money = rs.getBigDecimal("money");
            //sum 是 money 累加值
            sum = sum.add(money);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        DBUtils.close(rs, state, conn);
    }
}

```

```

    }
    return sum;
}

//获取全部数据
public List<HomeCost> list(String username) {
    String sql = "select * from home where username ='" + username + "'";
    List<HomeCost> list = new ArrayList<>();
    Connection conn = DBUtils.getConn();
    Statement state = null;
    ResultSet rs = null;
    try {
        state = conn.createStatement();
        rs = state.executeQuery(sql);
        HomeCost homeCost = null;
        while (rs.next()) {
            int id = rs.getInt("id");
            BigDecimal sum = queryMoneySum(id, username);
            String name = rs.getString("name");
            BigDecimal money = rs.getBigDecimal("money");
            String date = rs.getString("date");
            homeCost = new HomeCost(id, name, money, date, sum);
            list.add(homeCost);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        DBUtils.close(rs, state, conn);
    }
    return list;
}

```

//查询总数据量

```

public int getTotalCount(String username) {
    String sql = "select count(1) from home "+" where username = '" + username + "'";
    return DBUtils.getTotalCount(sql);
}

```

```

    }

    public List<HomeCost> queryCostsByPage(int currentPage,int
    pageSize,String username) {
        int start=pageSize*(currentPage-1);
        String sql = "select * from home "+"where username =
        ""+username+" limit "+start+", "+pageSize;
        List<HomeCost> list = new ArrayList<>();
        Connection conn = DBUtils.getConn();
        Statement state = null;
        ResultSet rs = null;
        try {
            state = conn.createStatement();
            rs = state.executeQuery(sql);
            while (rs.next()) {
                int id = rs.getInt("id");//获取查询结果中的 id
                BigDecimal sum = queryMoneySum(id,username);
                String name = rs.getString("name");//获取查询结果中的
                name
                BigDecimal money = rs.getBigDecimal("money");//获取
                查询结果中的 money
                String date = rs.getString("date");//获取查询结果中的
                date
                String note=rs.getString("note");
                HomeCost homeCost = new
                HomeCost(id,name,money,date,sum,note);//调用构造方法赋值
                list.add(homeCost);//添加到 list 集合中
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DBUtils.close(rs, state, conn);
        }
        return list;
    }
}

```



## (4) 运行结果(截图)

### 用户信息后台管理

用户名	姓名	密码	电话	性别	生日	邮箱	操作
hqy520_	小气鬼	123456Qa@	13546051452	female	2020-06-06	1378446627@qq.com	修改 删除
qa	小乔	12297804qa	18234236305	male	2020-04-04	1223954783@qq.com	修改 删除
共有3笔消费记录							

新增

### 用户信息后台管理

用户名	姓名	密码	电话	性别	生日	邮箱	操作
hqy520_	小气鬼	123456Qa@	13546051452	female	2020-06-06	1378446627@qq.com	修改 删除
qa	小乔	12297804qa	18234236305	male	2020-04-04	1223954783@qq.com	修改 删除
qa325	小明	123456	13546051134	female	2020-06-11	1223954783@qq.com	修改 删除
共有3笔消费记录							

新增

### 修改学生信息

用户名:

姓名:

密码:

电话:

性别:

生日:

电子邮箱:

### 添加用户

用户名:

姓名:

密码:

电话:

性别:

生日:

电子邮箱:

家庭记账本

新增消费记录

消费名称	消费金额	累计消费	登记日期	操作
粥	15.00	15.00	2020-06-13 16:22:39	修改 详情 删除
晚饭	100.00	115.00	2020-06-14 21:13:02	修改 详情 删除
午饭	500.00	615.00	2020-06-14 22:17:33	修改 详情 删除
西游记	50.00	665.00	2020-06-15 17:14:33	修改 详情 删除
夜宵	500.00	1165.00	2020-06-15 17:22:04	修改 详情 删除
新增				
首页	上一页	下一页	尾页	<input type="text" value="10 条/页"/>
共有5笔消费记录 当前页数: 1 总共有1 页				

### （5）心得体会（问题调试及体会）

通过完成本次记账本的项目，掌握了很多技术，掌握了很多知识，学到了很多東西。

首先对于前端页面的编写，开始的时候学会了HTML+CSS+JavaScript技术，并且能够开发出简单的界面，当然可能界面不够美观，后期我自学了bootstrap框架，采取框架制作了用户信息的管理的界面以及增加和修改的界面，发现采用框架来开发前端界面是非常方便的，而且开发出来的效果也不错，对于前端界面的设计还是很重要的，好的前端设计可以给用户带来良好的体验性，用户友好性高。

对于后端的编写，我发现其实都是一个套路，就是先编写数据库层进行与数据库的数据交互，接着编写业务逻辑层进行业务处理，最后编写servlet作为控制器与前端进行数据交互。对于数据库的操作可以封装成一个类进行代码的复用，这样很大程度上减少了代码的冗余，而对于控制器servlet的编写也很重要，这块调用业务逻辑层实现数据库的操作，以及返回给前端的数据。

对于项目的功能，我实现了以下功能：

1、用户的登录注册以及前端对用户的登录注册的信息的验证，例如：输入不能为空，用户名不少于四位。

2、验证码以及验证码的校验，采用kaptcha组件

3、采用ajax技术异步处理，在用户注册填入用户名后立即反馈给用户。

此用户是否存在，增强了用户友好性。

4、分页查询的功能，对于用户和记账都有分页查询的功能，并且当用户。

改变当前页显示的数据的数量时刷新显示，此处采用jquery进行编写。

5、对于后端用户信息的增删改查，此处前端页面采用bootstrap进行编写。

6、记账详情页的添加。

总的来说，通过这个小实验我掌握了很多知识，学会编写简单的网站，深刻理解了MVC模式，学会了很多技术，例如：bootstrap, jsp, servlet, jdbc, ajax, JSTL。当然我希望接下来对这个项目进行进一步完善，例如对于界面的UI设计还有待改善，对于功能的完善还有待增加，后期希望采用框架技术对本项目做一个改良版本。

封面设计： 贾丽

地 址： 中国河北省秦皇岛市河北大街 438 号

邮 编： 066004

电 话： 0335-8057068

传 真： 0335-8057068

网 址： <http://jwc.ysu.edu.cn>