**《Java程序设计》实验报告**

**学生姓名： 耿立博 班级： 19计算机3班 学号： 2019207320123**

**实验日期： 2021.5.25 指导教师： 胡继礼 、谷宗运、李芳芳**

**实验环境：win7+Jdk1.8**

**一、实验目的**

**掌握** 1.掌握String类的使用；

2.掌握正则表达式的使用；

3.StringBuilder、StringBuffer类的使用。

**熟悉**：1.字符串及不可变性；

3.常用字符串处理方法。

**二、实验内容**

1.新建一个java类TestString，要求如下

（1）接收用户从键盘输入的字符串s1和s2，字符串内容为“hello，world！”；

（2）分别使用==和equals方法判断字符串是否相等，并打印输出结果；

（3）判断字符串s1是否以’h’开头，并打印输出结果；

（4）截取字符串，获取”world”单词，并打印输出结果；

（5）打印输出逗号,在字符串中的位置；

（6）对字符串中所有字母o替换为a；并打印输出结果.

2. 新建一个java类TestString2

从键盘输入一行字符串“Dog loves”，利用String类的contains函数测试是否包含“Dog”字符串，如果包含在字符串结尾添加“ bones”，并打印输出 （使用StringBuilder拼接），否则输出“字符串不包含指定字符Dog”。

3．新建一个java类LetterCount，要求如下

（1）随机生成100个小写字母（a-z）存入字符串数组；

（2）统计输出100个字母中，各个字母出现的次数

实现思路供参考：

1. 使用Math.random()\*26+’a’ 随机生成字母，存入字符串数组s；
2. 对s进行遍历，存入集合map，如果key存在value+1，否则value=1；

4．新建一个java类TestPhoneRegex，要求如下

（1）接收用户从键盘输入的手机号；

（2）使用正则表达式判断用户输入的手机号是否正确，手机号码要求，见下图；

（3）打印输出判断结果。

5．新建一个java类TestEmailRegex，要求如下

（1）接收用户从键盘输入的邮箱；

（2）使用正则表达式判断用户输入的邮箱是否正确；

（3）打印输出判断结果。



**三、实验过程（按上述实验内容要求写出源代码及运行结果）**

**1.实验-1**

**【源码】**

**package** cn.edu.ahtcm.bean;

/\*

\* 1.新建一个java类TestString，要求如下

（1）接收用户从键盘输入的字符串s1和s2，字符串内容为“hello，world！”；

（2）分别使用==和equals方法判断字符串是否相等，并打印输出结果；

（3）判断字符串s1是否以’h’开头，并打印输出结果；

（4）截取字符串，获取”world”单词，并打印输出结果；

（5）打印输出逗号,在字符串中的位置；

（6）对字符串中所有字母o替换为a；并打印输出结果.

\*/

**import** java.util.Scanner;

**public** **class** TestString {

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

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

String s1 = sc.nextLine();

String s2 = sc.nextLine();

**if** (s1==s2){

System.***out***.println("s1==s2");

}**else**{

System.***out***.println("s1 is not == s2");

}

**if** (s1.equals(s2)){

System.***out***.println("s1 equals s2");

}**else**{

System.***out***.println("s1 is not equals s2");

}

/\*substring(int beginIndex, int endIndex)

\* 返回从起始位置（beginIndex）到目标位置（endIndex）之间的字符串，

\* 但不包含目标位置（endIndex）的字符\*/

// if (s1.substring(0,1)=="h"){

// System.out.println("s1 首字母为 h");

// }else{

// System.out.println("s1 首字母不为h");

// }这个方法达不到效果省略不用！！！

/\*java.lang.String类中的一个方法

\* public char charAt(int index)返回指定索引处的 char 值。

\* 索引范围为从 0 到 length() - 1。序列的第一个 char 值在索引 0 处，

\* 第二个在索引 1 处，依此类推，这类似于数组索引。\*/

// if (s1.charAt(0)=='h'){

// System.out.println("s1 首字母为 h");

// }else{

// System.out.println("s1 首字母不为h");

// }

**if** (s1.startsWith("h")){

System.***out***.println("s1 首字母为 h");

}**else**{

System.***out***.println("s1 首字母不为h");

}

// startsWith() 方法用于检测字符串是否以指定的前缀开始。如果字符串以指定的前缀开始，则返回 true；否则返回 false。

System.***out***.println(s1.substring(6,11));

/\* indexOf(String str): 返回指定字符在字符串中第一次出现处的索引，

\* 如果此字符串中没有这样的字符，则返回 -1。\*/

System.***out***.println(s1.indexOf(","));

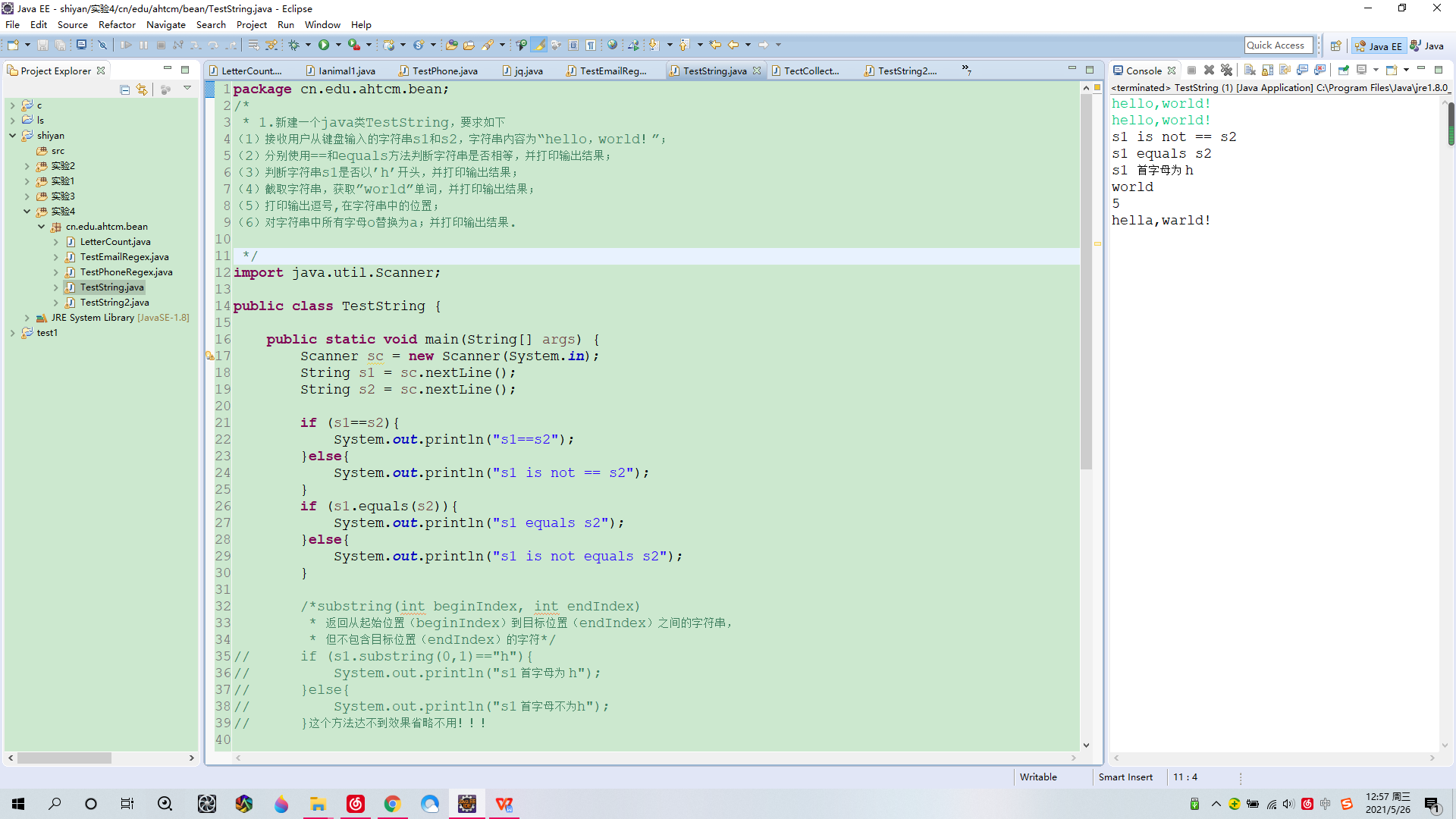
String s3=s1.replace("o","a");

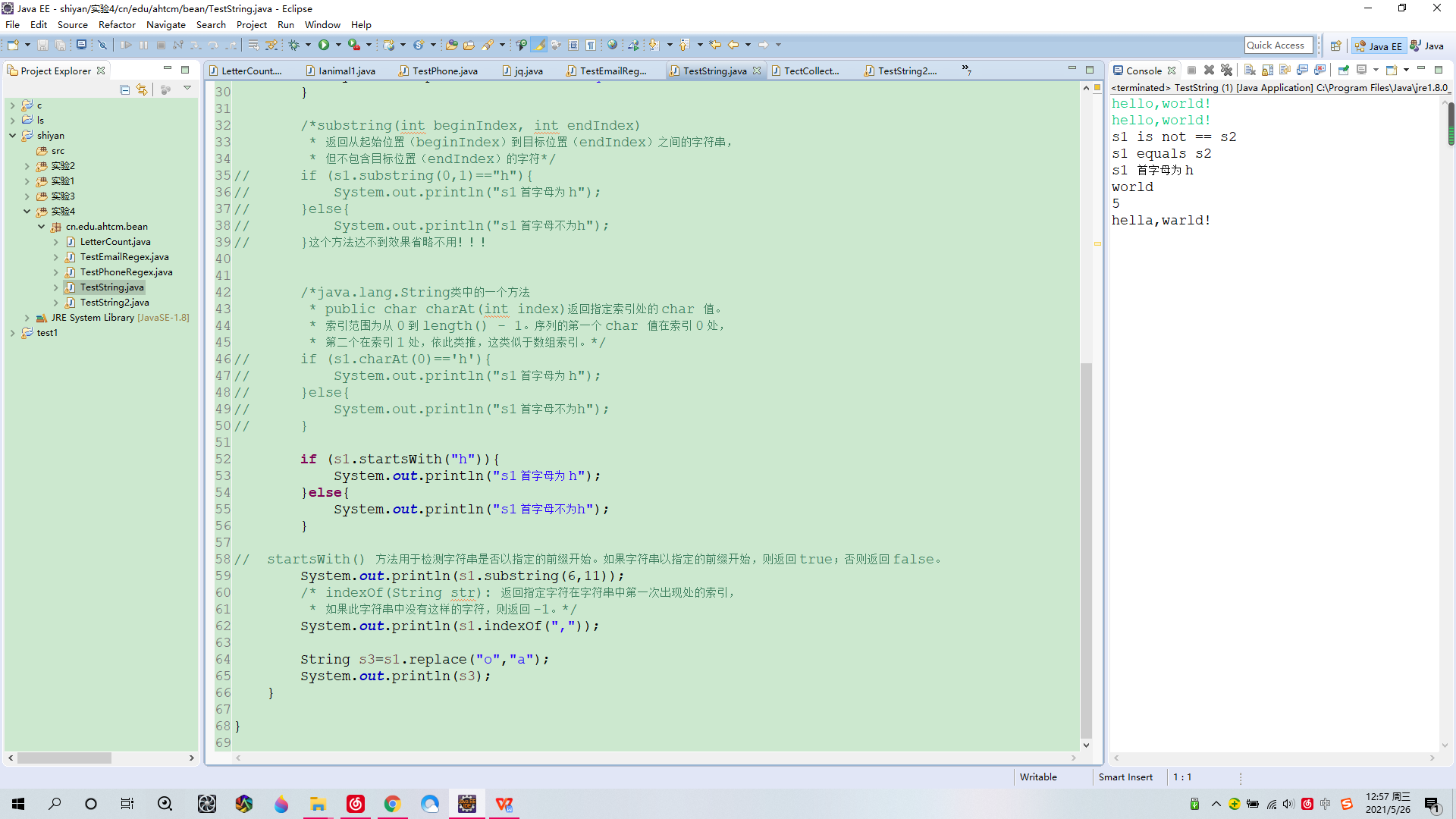
System.***out***.println(s3);

}

}

**【运行结果】**





**2.实验-2：**

**【源码】**

**package** cn.edu.ahtcm.bean;

/\* 2. 新建一个java类TestString2

从键盘输入一行字符串“Dog loves”，利用String类的contains函数测试是否包含“Dog”字符串，

如果包含在字符串结尾添加“ bones”，并打印输出 （使用StringBuilder拼接），

否则输出“字符串不包含指定字符Dog”。

\*/

**import** java.util.Scanner;

**public** **class** TestString2 {

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

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

String s1 = sc.nextLine();

**if**(s1.contains("Dog")){

StringBuilder sb = **new** StringBuilder(s1);

sb.append(" bones");

System.***out***.println(sb);

}**else**{

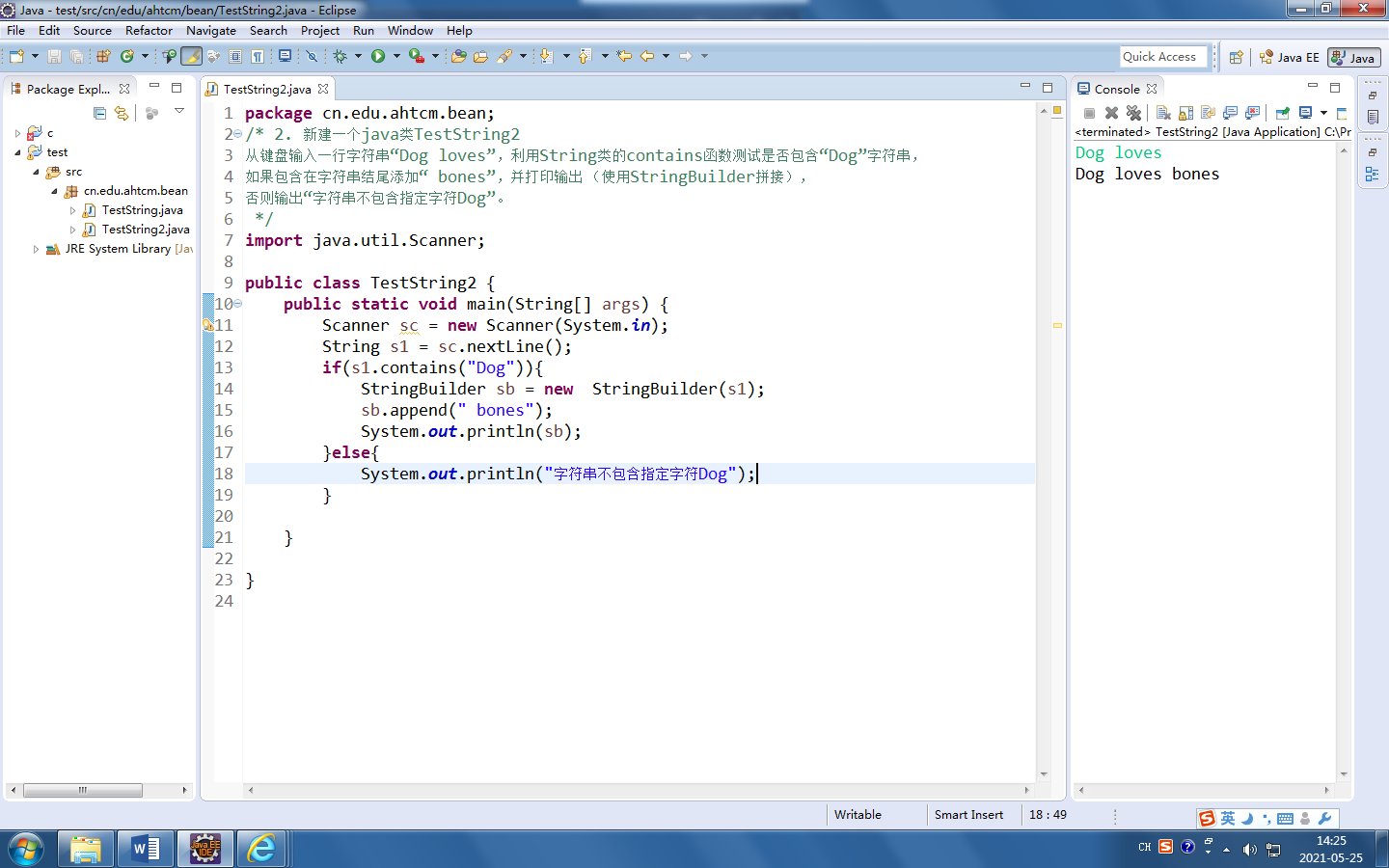
System.***out***.println("字符串不包含指定字符Dog");

}

}

}

**【运行结果】**



**3.实验-3：**

**【源码】**

**package** cn.edu.ahtcm.bean;

**import** java.util.HashMap;

**import** java.util.Map;

/\*3．新建一个java类LetterCount，要求如下

（1）随机生成100个小写字母（a-z）存入字符串数组；

（2）统计输出100个字母中，各个字母出现的次数

实现思路供参考：

（1） 使用Math.random()\*26+’a’ 随机生成字母，存入字符串数组s；

（2） 对s进行遍历，存入集合map，如果key存在value+1，否则value=1； \*/

**public** **class** LetterCount {

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

StringBuilder sb = **new** StringBuilder("");

**for**(**int** i=0;i<100;i++){

sb.append((**char**)(Math.*random*()\*26+'a'));

}

System.***out***.println(sb);

/\*substring(int beginIndex, int endIndex)

\* 返回从起始位置（beginIndex）到目标位置（endIndex）之间的字符串，

\* 但不包含目标位置（endIndex）的字符

\*用这个方法可以截取字符串然后将其存入集合中作为键\*/

//如果键存在，将值加一，首先这个方法很陌生似乎并不知道这种还有这种操作，

//我们可以用get方法将此键对应值取出放在一个临时变量t里，然后将临时变量t加一再用put方法放进去

//可以用containsKey方法map集合中是否存在指定的key，将其添加进去，并将值置为1；

Map <String,Integer> map = **new** HashMap<String,Integer>();

**for**(**int** i=0;i<100;i++){

**if**(map.containsKey(sb.substring(i,i+1))){

**int** t = map.get(sb.substring(i,i+1))+1;

map.put(sb.substring(i,i+1),t);

}**else**{

map.put(sb.substring(i,i+1),1);

}

}

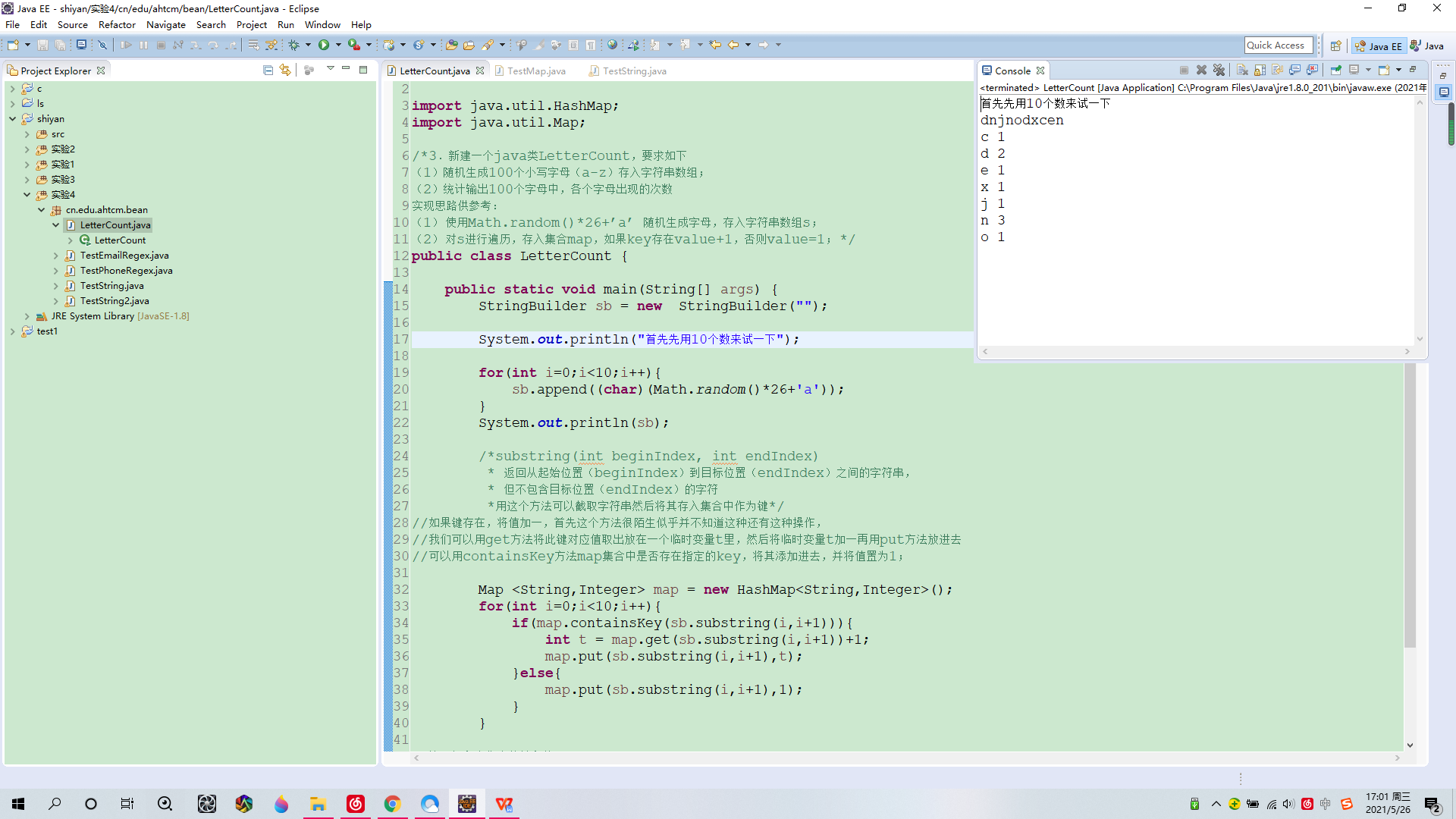
//遍历打印出集合的键和值

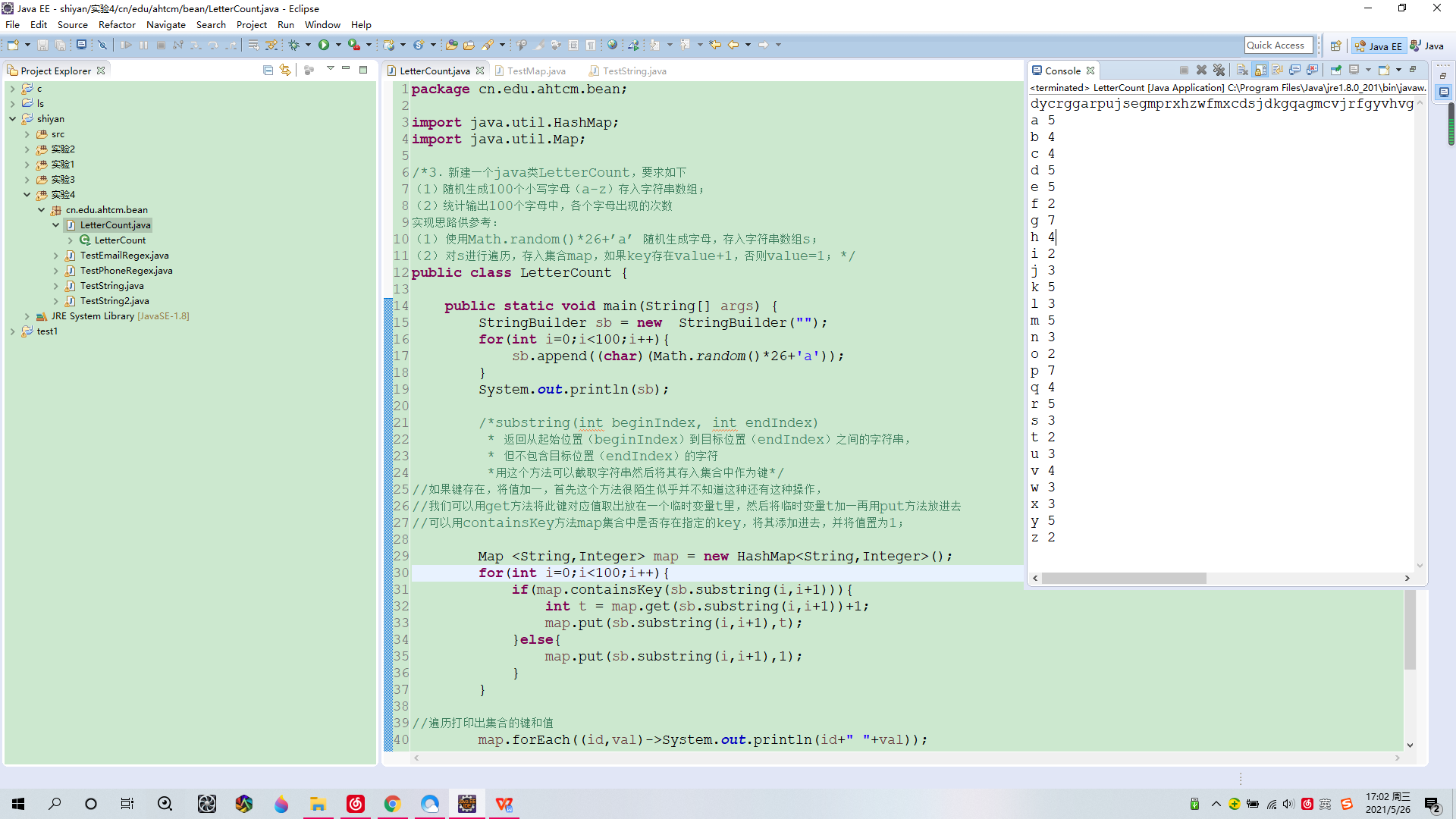
map.forEach((id,val)->System.***out***.println(id+" "+val));

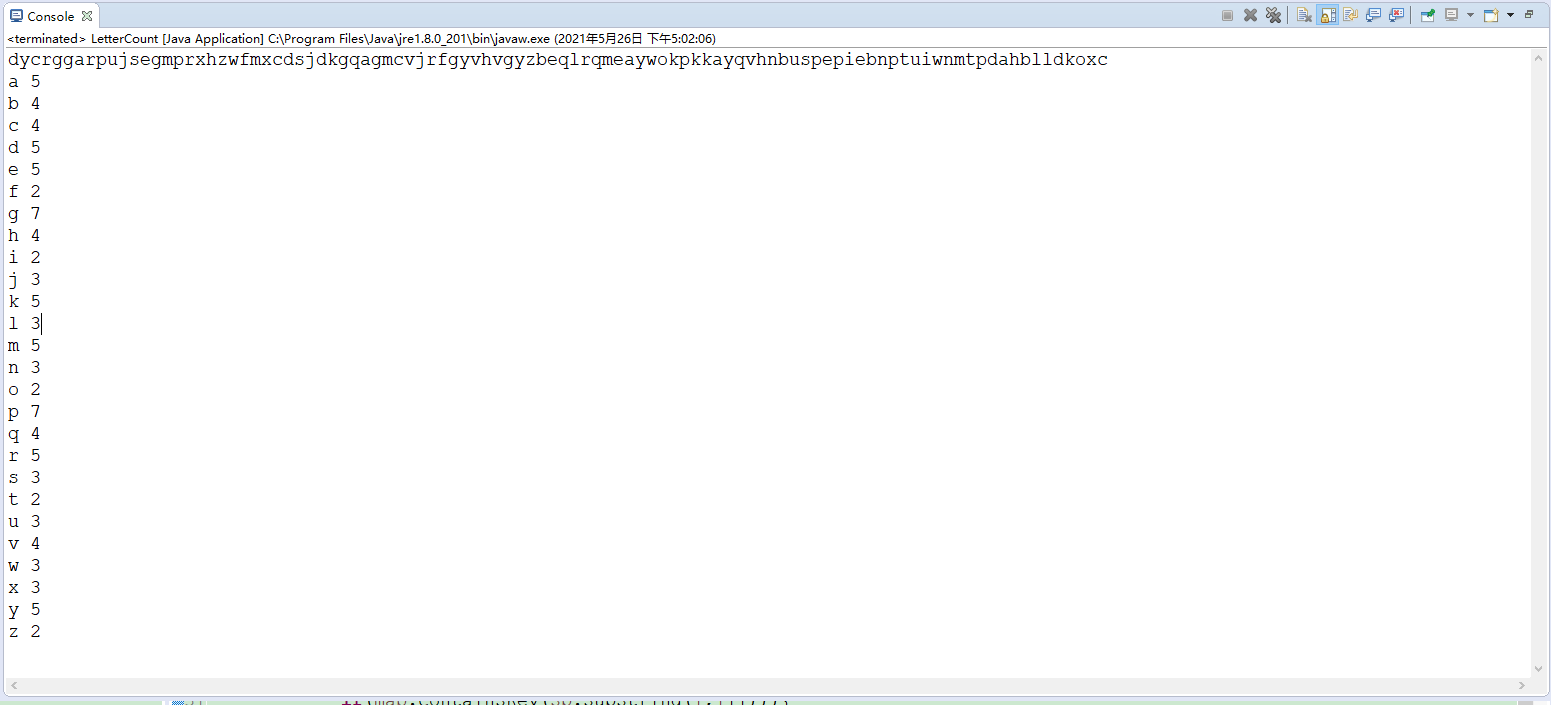
}

}

**【运行结果】**







**4.实验-4：**

**【源码】**

**package** cn.edu.ahtcm.bean;

/\*4．新建一个java类TestPhoneRegex，要求如下

（1）接收用户从键盘输入的手机号；

（2）使用正则表达式判断用户输入的手机号是否正确，手机号码要求，见下图；

（3）打印输出判断结果。\*/

**import** java.util.Scanner;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

//String phoneRegex = "^((13[0-9])|(14[5,7])|(15[0-3,5-9])|(17[6-8])|(18[0-9]))\\d{8}$";

**public** **class** TestPhoneRegex {

**private** **static** **final** Pattern ***CHINA\_PATTERN*** = Pattern

.*compile*("^((13[0-9])|(14[5,7])|(15[0-3,5-9])|(17[6-8])|(18[0-9]))\\d{8}$");

**public** **static** **boolean** isChinaPhone(String str){

Matcher m = ***CHINA\_PATTERN***.matcher(str);

**return** m.matches();

}

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

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

String phonenum = sc.nextLine();

**if**(*isChinaPhone*(phonenum)){

System.***out***.println("手机号码正确");

}**else**{

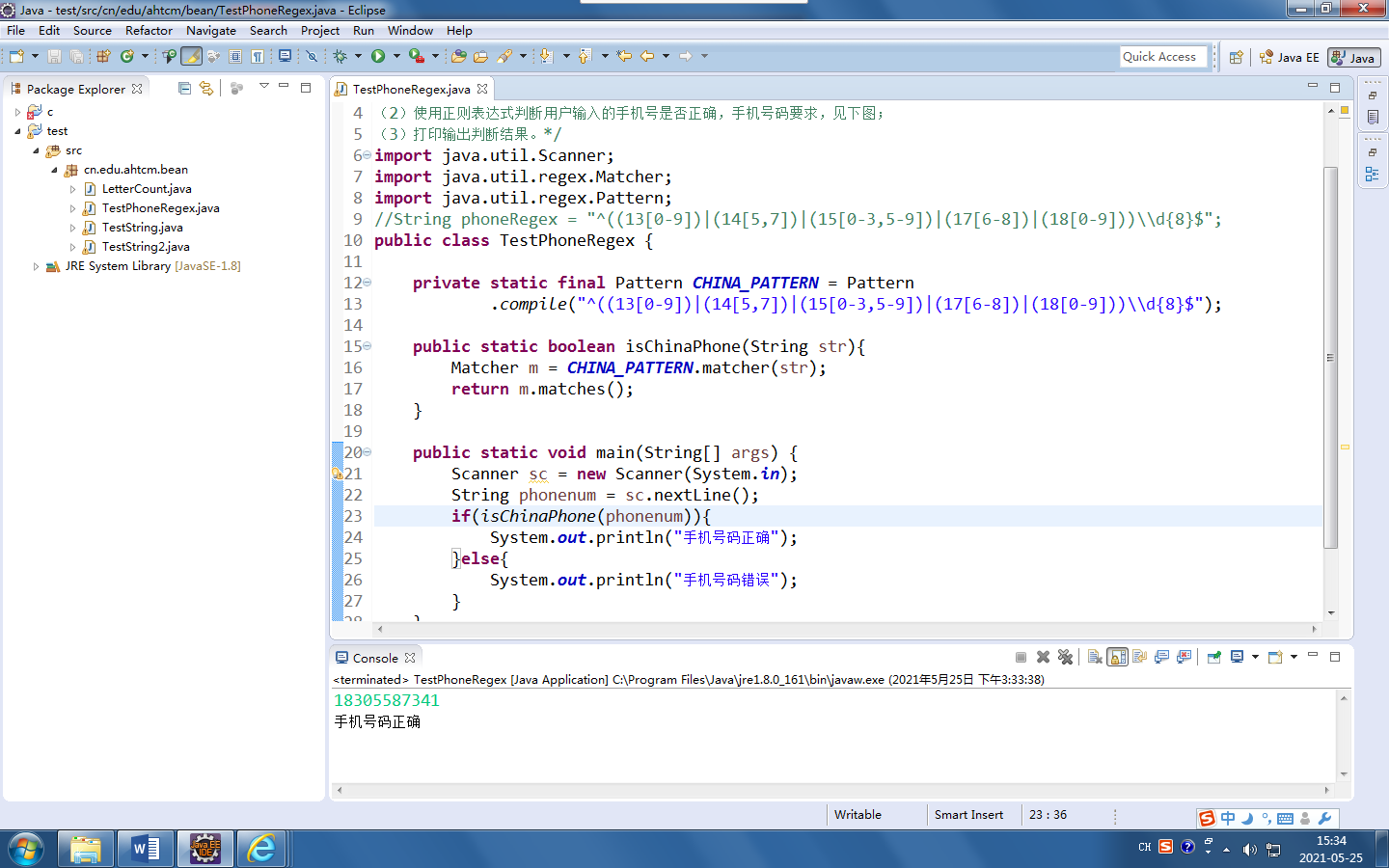
System.***out***.println("手机号码错误");

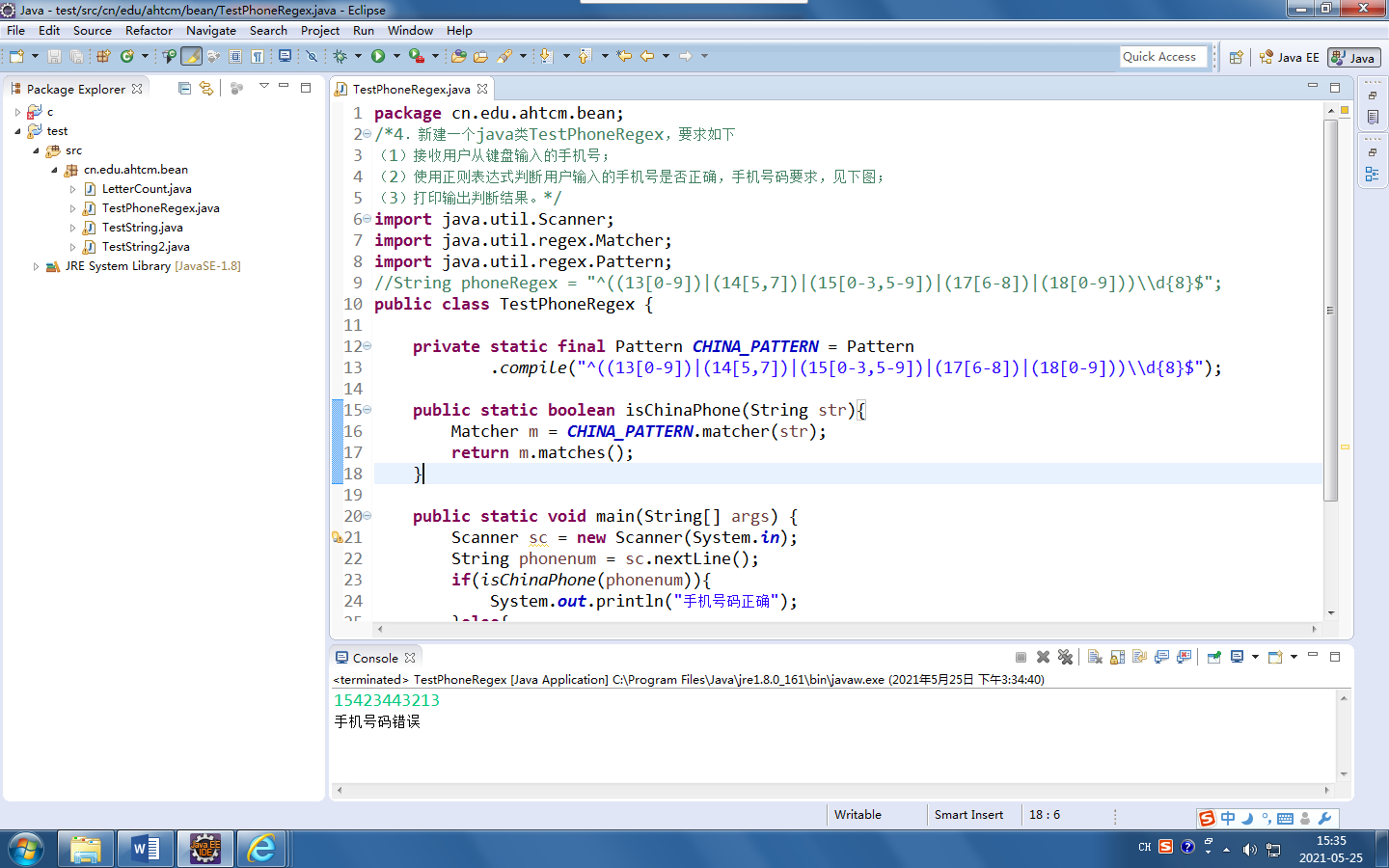
}

}

}

**【运行结果】**





**5.实验-5：**

**【源码】**

**package** cn.edu.ahtcm.bean;

**import** java.util.Scanner;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

/\*5．新建一个java类TestEmailRegex，要求如下

（1）接收用户从键盘输入的邮箱；

（2）使用正则表达式判断用户输入的邮箱是否正确；

（3）打印输出判断结果。\*/

// String EmailRegex = "[a-zA-Z0-9\_]+@[a-zA-Z0-9]+(\\.[a-zA-Z]+){1,3}";

**public** **class** TestEmailRegex {

**private** **static** **final** Pattern ***EMIAL\_PATTERN*** = Pattern

.*compile*("[a-zA-Z0-9\_]+@[a-zA-Z0-9]+(\\.[a-zA-Z]+){1,3}");

**public** **static** **boolean** isEmail(String str){

Matcher m = ***EMIAL\_PATTERN***.matcher(str);

**return** m.matches();

}

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

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

String email = sc.nextLine();

**if**(*isEmail*(email)){

System.***out***.println("email格式正确");

}**else**{

System.***out***.println("email格式错误");

}

}

}

**【运行结果】**

