**中软国际考题-笔试**

1. 单选题

1、下列表示正整数的正则表达式为（B）

A. ^-?[1-9]\d\*$ B. ^[1-9]\d\*$ C. ^[1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*|0?\.0+|0$ D.以上均不是

2、正则表达式^[1-9]\d\*\.\d\*|0\.\d\*[1-9]\d\*|0?\.0+|0$表示验证的选项是（A）

A. 非负浮点数 B. 正浮点数 C. 正整数 D. 非正浮点数

3、\\W能匹配（D）。

A.非数字 B.非数字 C.单词字符 D.单词字符

4、正则表达式中，表示字符出现0次或多次的字符是？（A）

A．\* B．？ C．+ D．-

5、（C）可以判断手机号码是否合法。

A．1{358}\\d{9} B．1[358]\d{9} C．1[358]\\d{9} D．1[358]\\D{9}

6、下列选项中，哪些不属于Java语句正则表达式的元字符（B）

A.\w B.\a C.\d D.\D

7、非正整数的正则表达式是：（B）

A.^-?\d+$ B.^(-\d+|(0+))$

C.^-[0-9]\*[1-9][0-9]\*$  D.^(-\d+|(9-))$

8、简单的身份证号验证的正则表达式是：（C）

A. ^[a-zA-Z]\w{7,10}$ B.^[\u4e00-\u9fa5]{0,}$

C.\d{15}|\d{18}$ D.^(-?\d+)(\.\d+)?$

9、正则表达式 /a+(bab)?(caac)\*/ ，下列选项中是该正则表达式的子集是？（C）

A．/(bab)(caca)/

B．/a(bab){2}(caac)\*/

C．/a{2}/

D．/a+(bab){0,1}(ca)+(ca)/

10、 验证字符串的组成规则，第一个须为数字，后面可以是字母、数字、下划线，总长度为5-20位，以下正确的是：( C )

A./^\d{1}[\w|\_|\d]{4,19}/ B./^[a-zA-Z]{1}[\w|\_|\d]{4,19}/ C./^\d{1}[\w|\_|\d]{4,19}$/ D./^[a-zA-Z]{1}[\w|\_|\d]{4,19}$/

11、正则表达式中可以表示所有的单词字符信息的是下面哪个规则：(D )

A． \W B． \w C． [a-zA-Z] D．[a-zA-Z\_0-9]

12、下表列出了的元字符和对它们的一个简短的描述错误的是（C）。

A、^ 匹配输入字行首。。

B、$ 匹配输入行尾。

C、\* 匹配前面的子表达式1次。

D、+ 匹配前面的子表达式一次或多次(大于等于1次）。。

13、求非负整数（B）

A.\d B.^\d+$ C.\d+$ D.^\d

1. 多选题
2. 如下选项中关于正则表达式使用的场景描述错误的是？（A、B）
3. 在数据库查询中使用正则表达式筛选查询数据
4. 使用正则表达式来抓取http请求
5. 正则表达式可以根据匹配规则在文本中查询符合规则的文本
6. 正则表达式常用语验证数据是否合法
7. 正则表达式可以在java运行环境下执行

2、以下和正则表达式相关的类有：（A、C）

A.Pattern B.Driver C.Matcher D.Reg

3、下面哪些项，属于Java正则表达式的数字个数语法？（A、B、C、D）

A. \* B. + C. ?

D. {n} E. {n:} F . {n:m}

4、下列关于java的正则表达式说法正确的是？（A、B、E）

A. 正则表达式，是用来匹配字符串的，即检查一定的格式，还能按格式进行分组，替换...

B. 正则表达式可以提高你编程的效率。

C. 正则不能检查一个字符串是否是IP地址类型的

D. 正则可以验证手机号码格式以及手机号码是否存在

E. 正则进行数据合法性验证。可以用在所有需要进行数据验证的地方

5、匹配中国邮政编码的正则表达式（A、B）

A.[1-9]\d{5}(?!\d) B./^[1-9][0-9]{5}$/

C./^[1-9][0-9]{5}/ D./^[1-9][0-9]{6}$/

6、下面代表只能输入多位数字的是( AB )

A.^[0-9]\*$ B^\d{n}$

C.^(0|[1-9][0-9]\*) D.^[0-9]+\.{0,1}[0-9]{0,2}$

7、 正则表达式的特点有哪些？（A、BCD）

A灵活性、逻辑性教强；

B可以迅速地用极简单的方式达到字符串的复杂控制。

C对于刚接触的人来说，比较晦涩难懂。

D功能性非常强；

8、下面哪些项正确？（A、B、C、D）

A.电子邮箱验证：^\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$

B.验证输入只能是汉字 ： ^[\u4e00-\u9fa5]{0,}$

C.长度为8-10的用户密码 ：（以字母开头、数字、下划线）^[a-zA-Z]\w{7,10}$

D.URL地址验证：^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]\*)?$;

1. 填空题（每空1分，共10分）
2. Java中使用\_\_\_\_\_\_\_\_\_\_类来构建正则表达式对象。(Pattern)
3. 正则表达式中，表示“匹配结尾”的字符是\_\_\_\_\_\_\_\_。($)
4. Java中表示一个或多个汉子的正则表达式是：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。(^[\u0391-\uFFE5]+$)
5. Java中表示邮政编码的正则表达式是：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。(^[1-9]\d{5}$)
6. Java中表示手机号码的正则表达式是：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。(^1[3|4|5|8][0-9]\d{8}$)
7. 如下代码用来检验一个四则运算式中数据项的数目，请填写划线部分缺少的代码。  
   注意：只填写缺少代码，不要写任何多余内容，例如，已有的双引号。

public class A{

public static int f(String s){

return s.split("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_").length;//以运算符号分割

}

public static void main(String[] args){

System.out.println(f("12+35\*5-2\*18/9-3")); //7

System.out.println(f("354\*12+3-14/7\*6")); //6

}

}([\\p{Punct}](\\\\p{Punct}))

1. 写出身份证号正确的正则表达式为（）（isIDCard=/^[1-9]\d{9}(0[1-9]|1[0,2])(0[1-9]|[1,2]\d|3[0,1])\d{3}(\d|[x,X])$/;）
2. 长度为8-10的用户密码（以字母开头、数字、下划线）\_\_\_\_\_\_\_\_\_。(^[a-zA-Z]\w{7,10}$)
3. 只能输入最少m位，最多n位数字的正则表达式是\_\_\_\_\_\_\_\_\_( ^\d{m,n}$ )
4. 正则引擎主要可以分为两大类：一种是\_\_\_\_\_，一种是\_\_\_\_\_。(DFA,NFA)
5. 匹配正整数的正则表达式是：\_\_\_\_\_\_\_\_\_\_ 。(^[1-9]\*[1-9][0-9]\*$)
6. 判断题
7. 正则表达式一般在web前端的网页开发中使用，在java中无法使用正则表达式。（ F ）
8. 字符‘大家好’也是正则表达式，只是验证规则为字符匹配大家好。（T ）
9. 正则表达式中，^用于匹配开头。（ T ）
10. 字符“.”在Java语言的正则表达式有特殊含义，表示任何字符。（ T ）
11. String  regex = "^((13[0-9])|(14[5|7])|(15([0-3]|[5-9]))|(18[0,5-9]))\\d{8}$";  此表达式是验证手机号码的Java正则表达式。（ T ）
12. 让2013-6-7 变成 2013.6.7的是: let str = '2013-6-7'; let regExp = /-/g; （ T ）
13. 正则表达式使用^代表开始，$代表结束 ( T )
14. 2.正则表达式不光可以限定能输入什么，还能限定输入多少位( T )
15. 正则表达式是对[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6" \t "https://baike.baidu.com/item/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F/_blank)串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。（ T ）
16. 负整数的正则表达式是：^-[0-9]\*[1-9][0-9]\*$ 。（ T ）
17. 有数字、26个英文字母组成的字符串的正则表达式是：^[A-Za-z0-9]+$。（ T ）

五、简答题

1、请简单介绍正则表达式，并描述它的作用？

答案：

又称规则表达式。正则表达式通常被用来检索、替换那些符合某个模式(规则)的文本。

有如下的重要使用场景

验证数据是否正确。

局部或全文检索文本。

2、简述正则表达式及其用途。（5分）

答案：在编写处理字符串的程序时，经常会有查找符合某些复杂规则的字符串的需要。正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。

3、Java中是如何支持正则表达式操作的？（5分）

答案：Java中的String类提供了支持正则表达式操作的方法，包括：matches()、replaceAll()、replaceFirst()、split()。此外，Java中可以用Pattern类表示正则表达式对象，它提供了丰富的API进行各种正则表达式操作，请参考下面面试题的代码。

4、在Java语言中，正则表达式的作用有哪些?(分别写出2条即可)（5分）

答案：

字符串匹配（字符匹配）

字符串查找

字符串替换

字符切割

5、正则表达式是什么？（5分）

答案：正则表达式是一种可以用于模式匹配和替换的规范， 一个正则表达式就是由普通的字符（例如字符a到z） 以及特殊字符（元字符） 组成的文字模式， 它用以描述在查找文字主体时需要匹配的一个或多个字符串。 正则表达式作为一个模板， 将某个字符模式与所搜索的字符串进行匹配 。

6、验证字符串的组成规则，第一个须为数字，后面可以是字母、数字、下划线，总长度为5-20位

答案：/^\d{1}[\w|\_|\d]{4,19}$/

7、 让2013-6-7 变成 2013.6.7

答案：var str = '2013-06-07'

str.replace(/-/g,'.') 或者 let regExp = /-/g; str.replace(regExp, '.')

8、 var str = "get-element-by-id";改成驼峰命名

答案：let regExp = /-(\w)/g;

str= str.replace(regExp, function(word) {

return word.substring(1).toUpperCase();})

9、单词首字母都大写

答案：var name = ' han mei mei';

name = name.replace(/\b\w+\b/g, function(word){

return word.substring(0,1).toUpperCase()+word.substring(1);});

10、判断字符串是否包含数字

答案：var str = 'sji7ak8';

var regExp = /\d/;

regExp.test(str)

11、字符调换

答案：name = "Doe, John";name.replace(/(\w+)\s\*, \s\*(\w+)/, "$2 $1");

12、判断是否符合指定格式

1、XXX-XXX-XXXX 2、其中 X 为 Number 类型\*\*

答案：/^(\d{3}-){2}\d{4}$/.test(str);

13、简单匹配电话

答案：/^1[34578]\d{9}$/.test(tele);

14、给一个连字符串例如：get-element-by-id转化成驼峰形式？

var str = "get-element-by-id";

var reg = /-\w/g; // 匹配横杆以及之后的一个字符，全局匹配

console.log(str.replace(reg,function($0){

return $0.slice(1).toUpperCase();

// 匹配到到是-e -b -i 形式截取后一个字符转成大写

}));

15、 匹配二进制数字

var str = "10101111";

var reg = /^[01]+$/g;

console.log(reg.test(str));

16、 非零的十进制数字 (有至少一位数字, 但是不能以0开头)

var str = "81";

var reg = /^[1-9][0-9]?$/g;

console.log(reg.test(str));

17、 匹配一年中的12个月

var str = "12";

var reg = /^(0?[1-9]|1[0-2])$/g;

console.log(reg.test(str));

18、 匹配qq号最长为13为

var str ="10009093283333";

var reg = /^[1-9][0-9]{4,12}$/g;

console.log(reg.test(str));

19、 匹配常见的固定电话号码

var str = "000-12344562";

// \(? 匹配左括号一次或0次然后以0开头后面加两个数字，再匹配右括号或空格或减号一次或0次，随后匹配8个数字

var reg = /\(?0\d{2}[) -]?\d{8}/g;

console.log(str.match(reg));

20、 判断字符串是否包含数字

let str1 = 'abc9efh';

let str2 = 'abcefg';

let regExp = /\d/;

console.log(regExp.test(str1)); // true

console.log(regExp.test(str2)); // false

21、 匹配ip地址

var str = "255.221.221.12";

// [01]?\d\d?表示匹配小于199的数，可以说两位数或一位数，2[0-4]\d表示从200到249，配合25[0-5]就表示小于255的数了。

var reg = /(([01]?\d\d?|2[0-4]\d|25[0-5])\.){3}([01]?\d\d?|2[0-4]\d|25[0-5])/g;

console.log(str.match(reg));

22、 匹配用尖括号括起来的以a开头的字符串

var str = "<a herf='www.baidu.com'>";

var reg = /<a[^>]+>/g;

console.log(str.match(reg));

23、 分割数字每三个以一个逗号划分

var str = "12345678901";

function numSplit(str){

var re = /(\d)(?=(\d{3})+$)/g;

//(\d{3})+$ 的意思是连续匹配 3 个数字，且最后一次匹配以 3 个数字结尾。

//要找到所有的单个字符，这些字符的后面跟随的字符的个数必须是3的倍数，并在符合条件的单个字符后面添加,

return str.replace(re,'$1,');

}

console.log(numSplit(str));

24、find()和match()有什么区别？举例说明？

答案：public void test10(){

String str="a中A3";

String reg="[a-zA-Z]";

Pattern compile = Pattern.compile(reg);

Matcher matcher = compile.matcher(str);

if(matcher.matches()){

System.out.println("是全字母");

}

if (matcher.find()){

System.out.println("有字母");

}

match():是全匹配正则，find():是部分匹配

25、正则表达式的目的是什么？

1. 给定的字符串是否符合正则表达式的过滤逻辑（称作“匹配”）：

2. 可以通过正则表达式，从字符串中获取我们想要的特定部分。

26、正则表达式的概念？

答案：正则表达式（Regular Expression）是用于描述一组字符串特征的模式，用来匹配特定的字符串。

27、正则表达式的常见应用？

答案：1)验证字符串，即验证给定的字符串或子字符串是否符合指定的特征，例如，验证是否是合法的邮件地址、验证是否是合法的HTTP地址等等。

2)查找字符串，从给定的文本当中查找符合指定特征的字符串，这样比查找固定字符串更加灵活。

3)替换字符串，即查找到符合某特征的字符串之后将之替换。

4)提取字符串，即从给定的字符串中提取符合指定特征的子字符串。

28、元字符的概念？

答案：元字符（Metacharacter）是一类非常特殊的字符，它能够匹配一个位置或者字符集合中的一个字符。 如.、\w等都是元字符。

29、处理选项正则表达式有几种不同的模式？

答案：忽略大小写、多行模式、单行模式、忽略空白、显示补获。

六、程序实现题

1、正则表达式：定义正则表达式，并返回匹配正则表达式的文本

（10分）

|  |
| --- |
| import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class RegexUtil {  public static void main(String[] args) {  String str = "北京市(朝阳区)(西城区)(海淀区)";  Pattern p = Pattern.compile(".\*?(?=\\()");  Matcher m = p.matcher(str);  if(m.find()) {  System.out.println(m.group());  }  }  } |

2、写一段代码，使能够验证电子邮箱。测试数据：

apple@163.com 有效

banana@qq.com.cn 有效

cat@hotmail.99 无效

|  |
| --- |
| import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class 邮箱验证 {  public static void main(String[] args) {  String[] sArray = null;  String sRegEx = null;  sArray = new String[] { "apple@163.com", "banana@qq.com.cn", "cat@hotmail.99" };  sRegEx = "^\\w+@\\w+.[a-zA-Z]{2,3}(.[a-zA-Z]{2,3})?$";  validate(sArray, sRegEx);  }  static void validate(String[] sArray, String sRegEx) {  Pattern \_pattern = Pattern.compile(sRegEx);  Matcher matcher = null;  for (String s : sArray) {  if (matcher == null) {  matcher = \_pattern.matcher(s);  } else {  matcher.reset(s);  }  String result = s + (matcher.matches() ? "\t有效" : "\t无效");  System.out.println(result);  }  }  } |

3、对IP地址进行排序：192.168.1.200 10.10.10.10 3.3.50.3 127.0.0.1

（10分）

|  |
| --- |
| public class Test {  public static void main(String[] args) {  // 对IP地址进行排序：192.168.1.200 10.10.10.10 3.3.50.3 127.0.0.1  String temp = "192.168.1.200 10.10.10.10 3.3.50.3 127.0.0.1";  temp=temp.replaceAll("(\\d+)", "00$1");  temp=temp.replaceAll("0+(\\d{3})", "$1");//将ip地址补成同样位数  String[] str=temp.split(" +");  Arrays.sort(str);  for(String ips:str)  System.out.println(ips.replaceAll("0+(\\d+)", "$1"));  }  } |

4、需求： 将“我我我、、、我我、、我要、我要要、、、要要要、、要要、、学学学、、、、学学编、、、学编编编、、编编编程、、程程”还原成：我要学编程

（10分）

|  |
| --- |
| public class RegexTest1 {  public static void main(String[] args) {  String temp = "我我我、、、我我、、我要、要要、、、要要要、、要要、、学学学、、、、学学编、、、编编编、、编编编程、、程程";  temp=temp.replaceAll("(、+)","");  temp=temp.replaceAll("(.)\\1+", "$1");  System.out.print(temp);  }  } |

5、需求：对IP地址进行排序

（10分）

|  |
| --- |
| import java.util.Arrays;  public class RegexSortIP {  public static void main(String[] args) {  // 对IP地址进行排序：192.168.1.200 10.10.10.10 3.3.50.3 127.0.0.1  String temp = "192.168.1.200 10.10.10.10 3.3.50.3 127.0.0.1";  temp=temp.replaceAll("(\\d+)", "00$1");  temp=temp.replaceAll("0+(\\d{3})", "$1");//将ip地址补成同样位数  String[] str=temp.split(" +");  Arrays.sort(str);  for(String ips:str)  System.out.println(ips.replaceAll("0+(\\d+)", "$1"));  }  }  输出为  3.3.50.3  10.10.10.10  127.0.0.1  192.168.1.20 |

6、需求：检验email地址

（10分）

|  |
| --- |
| public class RegexTest {  public static void main(String[] args) {  String mail="zsy@sina.com.cn";  String regex="\\w+@[a-zA-Z0-9]+(\\.[a-zA-Z]{2,3}){1,2}";//注意：这种匹配方式适用于国内部分邮件地址，不能匹配所有的邮件  boolean b=mail.matches(regex);  System.out.println(mail+" is "+b);  }  } |

7、非捕获组，匹配每行字母个数是偶数个的数据，每行数据不为空，正则不能存在分组1

源串：

a

ab

abc

abcd

预期：

匹配得到 ab 和 abcd，不包含分组1

答案：\b(?:\w{4}|\w{2})\b

8、匹配由 A/S/D/F 4个字母(区分大小写)组成的长度为3字符串

源串：

ABC

ASD

ADS

ASF

BBC

A|S

A|D

ASDF

9、预期：以[]元字符获得3个字母的或集，匹配 ASD/ADS/ASF 3组数据

答案：\b[ASDF]{3}\b

匹配每行数据中以.jpg/.jpeg/.png/.gif结尾的图片名称（含后缀）

源串：

image.jpg

image.jpeg

image.png

image.gif

not\_image.txt

not\_image.doc

not\_image.xls

not\_image.ppt

预期：匹配 image.jpg/image.jpeg/image.png/image.gif 4个结果

答案：\b\w\*.(?:jpg|jpeg|png|gif)\b

10、校验密码必须包含字母、数字和特殊字符，6-16位，假定特殊字符为 -\_= 三个字符

源串：

12345

123456

1234561234561234

12345612345612345

a1234

a12345

-1234

-12345

a-123

a-1234

a-1234a-1234a-12

a-1234a-1234a-1234

aaaaa

aaaaaa

-\_=-\_

-\_=-\_=

预期：匹配

a-1234

a-1234a-1234a-12

答案：(?=^.{6,16}$)(?=.\*\d)(?=.\*[a-zA-Z])(?=.\*[-\_=])^.\*

11、分割数字每三个以一个逗号划分（var str = "12345678901";）

function numSplit(str){

var re = /(\d)(?=(\d{3})+$)/g;

return str.replace(re,'$1,');

}

console.log(numSplit(str));

12、请使用正则取出：Tech,Sales,内容1,内容2 要求最好返回值是一个二维数组：如：a[0][0] = ‘Tech’;a[0][1] = ‘Sales’; let str = 'Tech,Sales,内容1,内容2';

|  |
| --- |
| let regExp = /Tech|Sales|内容1|内容2/g;  let myArray;  let array = [  []  ];  while ((myArray = regExp.exec(str)) !== null) {  array[0].push(myArray[0]);  }  console.log(array); |

13、对输入的qq号进行匹配（qq匹配规则：长度为5-10位，纯数字组成，且不能以0开头。）

（10分）

|  |
| --- |
| public class regexTest {  public static void main(String[] args) {  //测试：  String qq1 = "1832137835";  System.out.println(qq1+"是qq号码吗？"+b1);  }    public static boolean isQQ(String qq) {  //定义匹配规则：  String regex = "[1-9][0-9]{4,9}";  //判断是否符合规则  boolean b = qq.matches(regex);    return b;  }  } |

14. 对输入的电话号码进行匹配（匹配要求：匹配成功的电话号码位数为11位的纯数字，且以1开头，第二位必须是：3、7、8中的一位，即只匹配13\*\*\*\*\*\*\*\*\*、17\*\*\*\*\*\*\*\*\*、18\*\*\*\*\*\*\*\*\*的电话号码）。

（10分）

|  |
| --- |
| public class regexTest {  public static void main(String[] args) {  //测试：  String t1 = "13745678901";    System.out.println(t1+"是电话号码吗？"+b1);  }    public static boolean isQQ(String qq) {  //定义匹配规则：  String regex = "1[378][0-9]{9}";    //判断是否符合规则  boolean b = qq.matches(regex);  return b;  }  } |

15、实现时间的计算: 要求用户输入身份证号，若格式有误，要求其重新输入。然后根据身份证号码输出20岁生日 所在周的周三的日期。 例如:

\* 出生日期:1992-07-15。 20岁生日:2012-07-15 当周的周三为:2012-07-18

（10分）

|  |
| --- |
| while (true) {  Scanner scan = new Scanner(System.in);  System.out.println("请输入你的身份证号");  String str = scan.next();  String str1 = "[0-9]+x";  String str2 = "[0-9]+";  if ((str.matches(str1) || str.matches(str2)) && str.length() == 18) {  String str3 = str.substring(6, 14);  SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");  Date date = sdf.parse(str3);  Calendar cale = Calendar.getInstance();  cale.add(Calendar.YEAR, 20);  cale.setTime(date);  cale.set(Calendar.DAY\_OF\_WEEK, 4);  date = cale.getTime();  System.out.println(date);  break;  } else {  System.out.println("身份证格式错误,请重新输入");  }  } |

16、三种方式正则表达式验证手机号是否合法。

|  |
| --- |
| public class RegexTest {  public static void main(String[] args) {  // TODO Auto-generated method stub  //方式1  String regex ="^((13[0-9])|(15[^4])|(18[0,2,3,5-9])|(17[0-8])|(147))\\d{8}$";  Pattern pattern = Pattern.compile(regex);  String phoneid= "13820396489";  Matcher matcher = pattern.matcher(phoneid);  System.out.println("结果是："+matcher.matches());    //方式2  System.out.println("结果是："+Pattern.matches(regex, phoneid));    //方式3  System.out.println(phoneid.matches(regex));  }  } |