## 一 简答题

1 InputStream的read方法返回值的意义是？

|  |
| --- |
| 答：read()方法返回的是ASCLL码值（整数），调用一次，就读取一个字节。返回读到的字节。当文件读取完毕，返回-1。也可以将整数通过char类型强制转换成字符串。 |

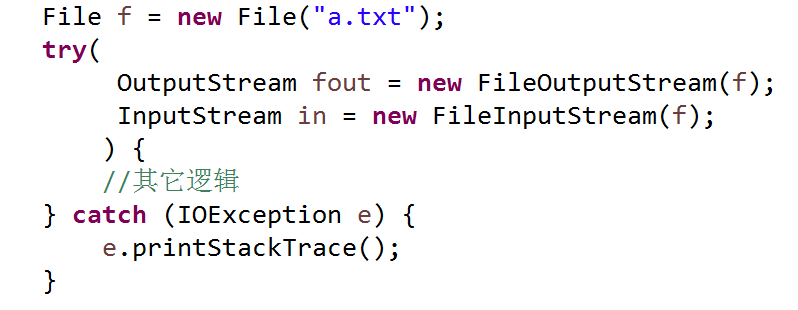
2 如何使System.out.println("HelloWorld");执行后，将HelloWorld字符串写入到a.txt文件内？

|  |
| --- |
| 答：PrintWriter out= new PrintWriter(new FileOutputStream(“D://a.txt”));  System.out.println("HelloWorld"); |

3 某一用户密码要求必须是3位字母开头，6位数字结尾的字符串。请写出一个正则表达式校验该用户密码字符串。

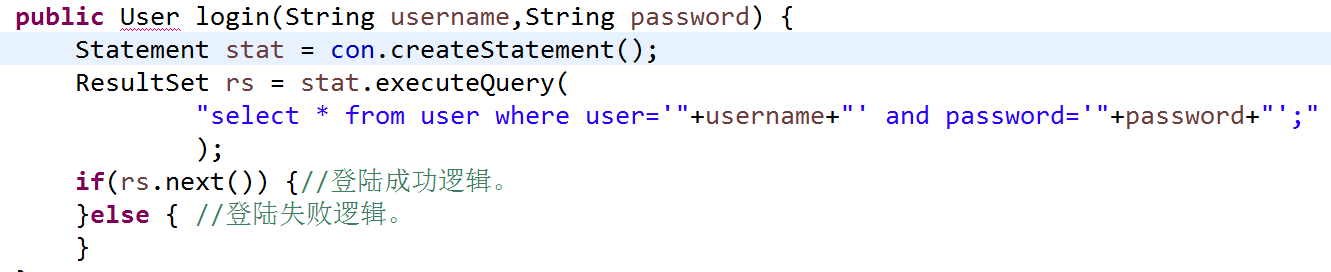
|  |
| --- |
| 答：^([a-zA-Z]{3}).{0,}([0-9]{6})$ |

4 以下代码为什么不需要对流进行关闭？



|  |
| --- |
| 答：因为字节流输出流不会把数据放入缓冲区，在不关流情况下，数据也会写入到目标终端。 |

5 一java应用程序进行登陆的逻辑如下，请对其进行SQL注入攻击。



|  |
| --- |
| 答：在"user"或是"password"中插入特殊符号，以改程序SQL的条件判断，  用户名：1  密码：1' OR '1' = '1  当程序接收到参数后，SQL语句就变成：select \* from user where user = '1' and password = '1' or '1' = '1'; |

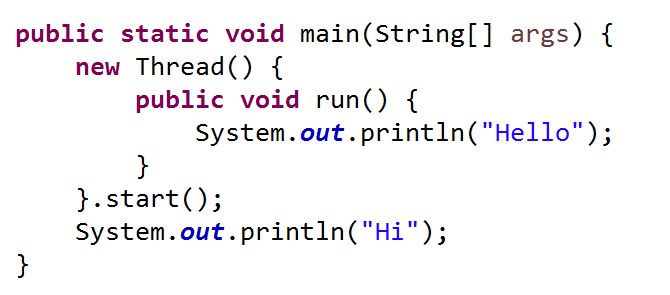
6 现有一字符串str。第一次采用Unicode字符集进行URL编码得到str1，第二次采用GBK字符集对str1进行编码得到str2。str2内容如下：

%25FE%25FF%2559%2529%2573%258B%2576%25D6%2557%2530%2586%254E%25FF%250C%255B%259D%2558%2554%2595%2547%256C%25B3%2559%2596%2530%2502

请用相应的java代码将其还原。（5行代码以内）

|  |  |
| --- | --- |
| 还原代码 | 请贴代码或截图String str1 = new String(str2.getBytes(),”GBK”); |
| 原始str为 | String oldStr = URLDecoder.decode(str1); |

7 下列代码如何确保Hi字符串永远在打印Hello之后被打印出来？



|  |
| --- |
| 答：贴代码或截图  **public** **static** **void** main(String[] args) **throws** InterruptedException  {  **new** Thread()  {  **public** **void** run()  {  System.***out***.println ("Hello");    }  }.start ();  Thread.*sleep*(8);  System.***out***.println("Hi");    } |

## 二 编程题

1 欲获取一个5000~9000之间的随机偶数，请写出一种实现方案。

|  |
| --- |
| 粘贴代码或截图  **public** **class** RandomText {  **public** **static** **void** main(String[] args) {  **int** [] num = **new** **int**[20];  Random r = **new** Random();  **for**(**int** i = 0;i<20;i++) {  **int** a = r.nextInt(9000);    **if**(a>5000) {    **if** (a%2==0)  System.***out***.println(a+"是偶数");  }  }  }  } |

2 请用定义一个类模拟实现ArrayList类，并实现ArrayList的add、remove、forEach、clear、isEmpty、set、get、length方法（请在100行代码以内实现）。

|  |
| --- |
| 粘贴代码或截图  **public** **static** **void** main(String[] args) {  ArrayList list = **new** ArrayList();  list.add(12);  list.add(1);  list.add("asdc");  System.***out***.println(list);  list.remove(0);  System.***out***.println(list);  list.set(11, 55);  System.***out***.println(list);  list.get(1);  System.***out***.println(list.get(1));  list.size();  System.***out***.println(list);  list.clear();  System.***err***.println(list);  System.***out***.println(list.isEmpty());    } |

3 请用多线程实现一个一对多的“生产者-消费者”模式的案例。（100行代码以内）

|  |
| --- |
| 粘贴代码或截图  //生产者  **class** producer **extends** Thread{  **private** Monitor s;  producer(Monitor s){  **this**.s = s;  }  **public** **void** run(){  **for**(**char** ch = 'A';ch <= 'E';ch++){  **try**{  Thread.*sleep*((**int**)Math.*random*()\*400);//线程休眠  }**catch**(InterruptedException e){  }  s.recordProduct(ch);  System.***out***.println(ch + "product has been produced by producer");  }  }  }  //消费者  **class** Consumer **extends** Thread{  **private** Monitor s;  Consumer(Monitor s){  **this**.s = s;  }  **public** **void** run1(){  **char** ch;  **do**{  **try**{  Thread.*sleep*((**int**)Math.*random*()\*400);//线程休眠  }**catch**(InterruptedException e){  }  ch = s.getProduct();  System.***out***.println(ch+"product has been consumed by consumer!");  }**while**(ch != 'E');  }  }  **class** Monitor{  **private** **char** c;  **private** **boolean** flag = **true**;  **public** **synchronized** **void** recordProduct(**char** c) {  **if**(!flag) {  **try** {  wait();  } **catch** (InterruptedException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  **this**.c = c;  flag = **false**;  notify();  }  **public** **synchronized** **char** getProduct() {  **if**(flag) {  **try** {  wait();  } **catch** (InterruptedException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }    }  flag = **true**;  notify();  **return** **this**.c;    }  }  测试类：  **public** **class** b {  **public** **static** **void** main(String[] args) {  Monitor s = **new** Monitor();  **new** producer(s).start();  **new** Consumer(s).start();  }    } |

4 对象属性复制。现有两个实体类A、B。它们没有继承任何父类也没有实现任何接口。A类有10个属性以及相应的setter和getter方法。B类有9个属性以及相应的setter和getter方法。A类的属性名、类型和B类完全一样。现在需要将B类的属性值复制到A类中。请使用反射实现BeanUtils.copy方法。（50行代码以内）

A a = new A();

B b= new B();

BeanUtils.copy(b,a);

|  |
| --- |
| 答：粘贴代码或截图 |

5 请用多线程 + ServerSocket实现一个最简单的HTTP服务器。要求如下：

* 只需要响应GET请求即可，不支持其它请求。
* 根据GET请求的路径，只返回.html , .css , .js ,. jpg , .png , .jpeg , .gif, .json 静态资源文件。
* 不支持长连接、Cookie、Session等功能。
* 80行代码以内。

|  |
| --- |
|  |

6 请写出对Map的遍历方式，尽可能多。案例自选。

|  |
| --- |
| 答：贴代码或截图  Map<String, Integer> map1 = **new** HashMap<String, Integer>();  map1.put("jack", 20);  map1.put("rose", 18);  map1.put("lucy", 17);  map1.put("java", 25);  //通过 map1.keySet() 获取key 通过key 找到value  **for** (String key : map1.keySet()) {  Integer value = map1.get(key);  System.***out***.println("key : "+key+" value : "+value);  }  //Entry方法：  Map<String, Integer> map2 = **new** HashMap<String, Integer>();  map2.put("jack", 20);  map2.put("rose", 18);  map2.put("lucy", 17);  map2.put("java", 25);    //通过Map.Entry(String,Integer) 获取，然后使用entry.getKey()获取到键，通过entry.getValue()获取到值  **for**(Map.Entry<String, Integer> entry : map2.entrySet()){  System.***out***.println("键 key ："+entry.getKey()+" 值value ："+entry.getValue());  } |

7 请对一张.jpg图片进行翻转，效果如下：左边为原图，右边为翻转后图片。（50行代码内）



|  |
| --- |
| 答：贴代码或截图  **public** **static** **void** main(String[] args) **throws** IOException {  *drawImg*();  }  **public** **static** **void** drawImg() **throws** IOException {  BufferedImage src = ImageIO.*read*(**new** File("多多.jpg"));  **int** w = src.getWidth();  **int** h = src.getHeight();  BufferedImage dest = **new** BufferedImage(w,h,BufferedImage.***TYPE\_INT\_RGB***);  Graphics g = dest.getGraphics();  g.drawImage(src, 0, 0, w,h,w,0,0,h, **null**);  ImageIO.*write*(dest, "jpg", **new** File("a.jpg"));  } |