* **选择题（60分，每题3分）**

1. 下面哪些是Thread类的方法（A）

A start()       B run()       C exit()       D getPriority()

2. 下面关于java.lang.Exception类的说法正确的是（A）

A 继承自Throwable      B Serialable      CD 不记得，反正不正确

3.下列说法正确的有（B）

A． class中的constructor不可省略

B． constructor必须与class同名，但方法不能与class同名

C． constructor在一个对象被new时执行

D．一个class只能定义一个constructor

下面程序的运行结果：（B）

public static void main(String args[]) {

Thread t = new Thread() {

public void run() {

pong();

}

};

t.run();

System.out.print("ping");

}

static void pong() {

System.out.print("pong");

}

A pingpong        B pongping       C pingpong和pongping都有可能       D 都不输出

5.下列说法正确的是（C）

A LinkedList继承自List // 都是实现接口

B AbstractSet继承自Set // 都是实现接口

C HashSet继承自AbstractSet

D WeakMap继承自HashMap

6、 实现批量数据导入的命令是( A )

 A.mysqldump B.mysql C.backup D.return

7、 创建用户的命令是( B)

A. join user B.create user C.create root    D.mysql   user

8、 修改自己的mysql服务器密码的命令是(C )

A.mysql B.grant C.set password D.change  password

9、 找回mysql服务器root密码的很重要的一步是跳过权限表的检查启动mysql,该命令是 ( D)

A. mysql  -u root –proot  B. mysqladmin   -uroot   -proot

C. net  start  mysql      D. mysqld-nt   --skip-grant-tables

10、联合查询使用的关键字是( B  )

A.UNION B.JOIN C.ALL D.FULL

11、在Linux中对文件夹itcast进行打包并压缩成itcast.tar.gz使用的命令是（ B）

A) tar -zxvf itcast itcast.tar.gz B) tar -zxvf itcast.tar.gz itcast

C) tar -zcvf itcast itcast.tar.gz D) tar -zcvf itcast.tar.gz itcast

12、Linux文件权限一共10位长度，分成四段，第三段表示的内容是（ D）

A) 文件类型 B) 文件所有者的权限

C) 文件所有者所在组的权限 D) 其他用户的权限

13、Linux生成ssh密钥的命令是（B ）

A) ssh-add

B) ssh-keygen

C) ssh-copy-id

D) ssh-agent

14、MySQL是一种基于( C )的数据库管理系统软件。

A) 层次模型 B)网状模型 C）关系模型 D）数据模型

15、主键约束（Primary Key）和唯一性约束（UNIQUE）都可以保证表中记录不重复，二者主要区别是（ A ）

A）主键约束不允许空值，唯一性约束允许出现一次空值

B）主键约束允许出现一次空值，唯一性约束不允许空值

C）二者都允许一次空值

D) 二者都不允许出现空值

16、如果要使用JDBC连接mysql的mydb数据库，那么url应该是（ D）

A) jdbc:mysql//:localhost:3306/mydb

B) jdbc:mysql://localhost:3306/mydb

C) mysql:jdbc//:localhost:3306/mydb

D) mysql:jdbc://localhost:3306/mydb

17用户表user中有字段age和count，数据类型都是int，数据如下

+------+------+-------+

| id | age | count |

+------+------+-------+

| u001 | 18 | 60 |

+------+------+-------+

执行select age + count from users where id = 'u001';结果为？( C )

A) 运行时将报错

B) 1860

C) 78

D) 18+60

18、账号表account中有字段money，现在要将钱数在100-200之间的增加50，以下sql语句正确的有( C)

A、update account set money + 50 where money >=100 and money <=200;

B、update account set money = money + 50 where money >=100 or money <=200;

C、update account set money = money + 50 where money between 100 and 200;

D、update account set money = money + 50 where money <=200 and money >100;

19、下列对JDBC驱动说法正确的是（ D ）

A　用JDBC连接数据库，必须使用连接池。

B　用JDBC连接数据库，使用连接池就不需要导入数据库驱动。

C　操作不同的数据库可以使用相同的驱动。

D　无论如何，只要使用JDBC，那么就要使用JDBC驱动。

20、哪个选项可以预防SQL攻击（ C ）

A　Connection

B　Statement

C　PreparedStatement

D　ResultSet

* **笔试题（40分）**

**SQL题（30分）**

1、（10分）

将测试数据表导入数据库，其中name字段代表“姓名”，score字段代表“分数”。

|  |
| --- |
| create table `t1` (     `id` double , // 数据库不存在double类型的东西的     `name` varchar (300),     `score` double  );  insert into `t1` (`id`, `name`, `score`) values('1','n1','59'); insert into `t1` (`id`, `name`, `score`) values('2','n2','66'); insert into `t1` (`id`, `name`, `score`) values('3','n3','78'); insert into `t1` (`id`, `name`, `score`) values('4','n1','48'); insert into `t1` (`id`, `name`, `score`) values('5','n3','85'); insert into `t1` (`id`, `name`, `score`) values('6','n5','51'); insert into `t1` (`id`, `name`, `score`) values('7','n4','98'); insert into `t1` (`id`, `name`, `score`) values('8','n5','53'); insert into `t1` (`id`, `name`, `score`) values('9','n2','67'); insert into `t1` (`id`, `name`, `score`) values('10','n4','88'); |

1. 单分数最高的人和单分数最低的人。（1分）

select name,max(score) ,min(score) from t1 group by name   
2. 两门分数加起来的第2至5名。（1分）  
select name ,sum(score) as score from t1 group by name order by score limit 1,5  
3. 两门总分数在150分以下的人。（2分）  
select name ,sum(score) as score from t1 group by name having score<=150  
4. 两门平均分数介于60和80的人。（2分）  
select name ,sum(score) as score from t1 group by name having score between 60 and 80   
5. 总分大于150分，平均分小于90分的人数。（2分）  
select name ,sum(score) as score ,avg(score ) as avg from t1 group by name having avg <90 and score>150  
6. 总分大于150分，平均分小于90分的人数有几个。（2分）  
select name ,sum(score) as score ,avg(score ) as avg ,count(DISTINCT(name)) from t1 group by name having avg <90 and score>150

2、（10分）

用一条SQL 语句 查询出每门课都大于80 分的学生姓名

|  |
| --- |
| name   kecheng   fenshu 张三    语文       81 张三     数学       75 李四     语文       76 李四     数学       90 王五     语文       81 王五     数学       100 王五     英语       90 |

对应的sql语句如下：

select \* from (

select name,

sum( case kechen when '语文' then fenshu else 0 end ) as m1,

sum( case kechen when '数学' then fenshu else 0 end) as m2,

sum( case kechen when '英语' then fenshu else 0 end ) as m3

from t2 group by name ) t where t.m1>80 and t.m2>80 and t.m3>80

1. （10分）

怎么把这样一个表儿

|  |
| --- |
| year   month amount 1991   1     1.1 1991   2     1.2 1991   3     1.3 1991   4     1.4 1992   1     2.1 1992   2     2.2 1992   3     2.3 1992   4     2.4 |

查成这样一个结果

|  |
| --- |
| year m1   m2   m3   m4 1991 1.1 1.2 1.3 1.4 1992 2.1 2.2 2.3 2.4 |

select year,

sum( case month when '1' then amount else 0 end ) as m1,

sum( case month when '2' then amount else 0 end) as m2,

sum( case month when '3' then amount else 0 end ) as m3,

sum(case month when '4' then amount else 0 end) as m4

from t3 group by year

**编程题（10分）**

用java代码完成下面题目,在一个日志文件2013-05-30.log中有如下数据:

27.19.74.143 - - [30/May/2013:17:38:25   
27.19.74.143 - - [30/May/2013:17:38:22   
27.19.74.143 - - [30/May/2013:17:38:23   
27.19.74.143 - - [30/May/2013:17:38:21   
27.19.74.143 - - [30/May/2013:17:38:28   
27.19.74.143 - - [30/May/2013:16:38:20   
27.19.74.143 - - [30/May/2013:18:38:20   
27.19.74.143 - - [30/May/2013:16:39:20   
27.19.74.143 - - [30/May/2013:16:41:20   
27.19.74.143 - - [30/May/2013:17:42:20   
27.19.74.143 - - [30/May/2013:19:38:21   
27.19.74.143 - - [31/May/2013:12:38:21   
27.19.74.143 - - [31/May/2013:16:38:21   
27.19.74.143 - - [31/May/2013:16:38:22   
27.19.74.143 - - [30/May/2013:14:38:21   
27.19.74.143 - - [30/May/2013:14:38:21   
  
该文件中的数据是一段截取自web服务器日志中的数据，每一行包含两个信息：   
1、网站访问者的ip   
2、网站访问者一次请求的时间

定义：连续的请求属于一次“独立访问”，如“30/May/2013:17:38:22”和“30/May/2013:17:38:25”两次请求属于同一次独立访问   
时间相邻的两次请求如果间隔超过30分钟，则视为分属两次不同的独立访问   
需求：统计出数据中的所有"独立访问"，输出每一次访问的起始请求时间，结束请求时间，及停留时长（毫秒）   
输出结果示例：   
31/May/2013:16:38:21    30/May/2013:16:41:20      181000

对应的java代码如下：

import com.alibaba.fastjson.JSON;  
import com.alibaba.fastjson.JSONArray;  
import com.alibaba.fastjson.JSONObject;  
import org.springframework.util.LinkedMultiValueMap;  
import org.springframework.util.MultiValueMap;  
  
import java.io.\*;  
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.\*;  
  
public class Answer {  
  
  
 public static void main(String[] args) throws IOException, ParseException {  
 BufferedReader fr=new BufferedReader(new FileReader(new File("C:\\Users\\Administrator\\Downloads\\wds-contact-master\\answer\\src\\main\\resources\\2013-05-30.txt")));  
 String content=null;  
 MultiValueMap<String, String> map = new LinkedMultiValueMap() ;  
 while((content=fr.readLine())!=null){  
 String[] result = content.split(" - - \\[");  
 List<String> values = map.get(result[0]);  
 if(values==null){  
 List<String > list=new ArrayList<String>();  
 list.add(result[1]);  
 map.put(result[0],list);  
 }else {  
 values.add(result[1]);  
 }  
 }  
 fr.close();  
 // 首先执行排序操作,按照时间进行排序操作处理  
 *sort*(map);  
 // 生成最终的计算结果  
 JSONArray jsonArray=*getFinalResults*(map);  
 jsonArray.stream().forEachOrdered(x->{  
 JSONObject cont=(JSONObject)x;  
 String start = (String) cont.get("start");  
 String stop = (String) cont.get("stop");  
 Long duration = (Long) cont.get("duration");  
 System.*out*.println(start+" " +stop+" "+duration);  
 });  
 }  
  
 */\*\*  
 \* 对应的一个ip地址对应的存在多个访问记录操作的，需要进行遍历操作管理  
 \* \*/* private static JSONArray getFinalResults(MultiValueMap<String, String> map) throws ParseException {  
 Set<String> keys = map.keySet();  
 Iterator<String> iterator = keys.iterator();  
 JSONArray result=new JSONArray();  
 while (iterator.hasNext()){  
 String key = iterator.next();  
 List<String> values = map.get(key);  
 // 对应的需要针对于单个ip形成多个数据记录操作  
 JSONArray res=*createJsonObject*(values);  
 if (res.size()>0){  
 result.addAll(res);  
 }  
 }  
 return result;  
 }  
  
 */\*\*  
 \* 判断和对应的生成jsonObject对象  
 \*  
 \** ***@param*** *values\*/* private static JSONArray createJsonObject(List<String> values) throws ParseException {  
 // 作为记录的起点数据  
 JSONArray jsonArray=new JSONArray();  
 String start=values.get(0);  
 if (values.size()>=2){  
 for(int i=1;i<values.size();i++){  
 String time = values.get(i);  
 String startTime = *getDestFormatValue*(start);  
 String endTime=*getDestFormatValue*(time);  
 // 超过了30分钟的话，对应的需要生成一个新的记录  
 Long duration = *pass30Minute*(startTime, endTime);  
 if (duration>=30\*60\*1000){  
 JSONObject content=new JSONObject();  
 content.put("start",start);  
 content.put("stop",time);  
 content.put("duration",duration);  
 start = time;  
 jsonArray.add(content);  
 }  
 }  
 }else {  
 System.*out*.println("记录时间太短了，无法生成相关的数据记录");  
 }  
 return jsonArray;  
 }  
  
 */\*\*  
 \* 记录超过了30分钟  
 \* \*/* private static Long pass30Minute(String startTime, String endTime) {  
 Long start = Long.*valueOf*(startTime);  
 Long end = Long.*valueOf*(endTime);  
 return end-start;  
 }  
  
 */\*\*  
 \* 排序操作实现  
 \* \*/* private static void sort(MultiValueMap<String, String> map) {  
 Set<String> keys = map.keySet();  
 Iterator<String> iterator = keys.iterator();  
 while (iterator.hasNext()){  
 String key = iterator.next();  
 List<String> values = map.get(key);  
 *sortAllValues*(values);  
 map.replace(key,values);  
 }  
 }  
  
 */\*\*  
 \* 排序指定key下面所有的value的  
 \* \*/* private static void sortAllValues(List<String> values) {  
 values.sort(new Comparator<String>() {  
 @Override  
 public int compare(String o1, String o2) {  
 try {  
 String d1 = *getDestFormatValue*(o1);  
 String d2 = *getDestFormatValue*(o2);  
 return d1.compareTo(d2);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 return 0;  
 }  
 });  
 }  
  
 */\*\*  
 \* 获取符合要求的时间格式数据  
 \* \*/* private static String getDestFormatValue(String time) throws ParseException {  
 String[] time\_prepare = time.split("/");  
 String[] dayMonthYear = time\_prepare[2].split(":");  
 String day=*getDay*(time\_prepare);  
 String month=*getMonth*(time\_prepare);  
 String year=*getYear*(dayMonthYear);  
 long timeFinal=*getFinalTime*(year,month,day,dayMonthYear[1],dayMonthYear[2],dayMonthYear[3]);  
 return timeFinal+"";  
 }  
  
 private static Long getFinalTime(String year, String month, String day, String hour, String minute, String second) throws ParseException {  
 DateFormat format=new SimpleDateFormat("yyyy-MM-dd-HH-mm-ss");  
 StringBuilder sb=new StringBuilder();  
 sb.append(year).append("-").append(month).append("-").append(day).append("-").append(hour)  
 .append("-").append(minute).append("-").append(second);  
 Date time = format.parse(sb.toString());  
 return time.getTime();  
 }  
  
 private static String getYear(String[] time) {  
 return time[0];  
 }  
  
 private static String getMonth(String[] time) {  
 switch (time[1]){  
 case "May":  
 return "5";  
 default:  
 return "0";  
 }  
 }  
  
 private static String getDay(String[] time) {  
 return time[0];  
 }  
}  
生成的结果显示如下：

