1.怎么实现输入x,找出数组中与x最接近的值(口述代码）

原始数据是乱序的：

package test;

import java.io.File;

public class Test {

/\*\* \* 假设原始数据是乱序 \*/

static int[] src = new int[] { 25, 68, 5, 38, 2, 15, 90, 55, 46 };

/\*\*\* 目标值\*/

static int x = 555;

public static void main(String[] args) {

System.out.println(getApproximate(x, src));

}

/\*\*\* 获取接近值 \*

\* @param x

\* @param src

\* @return

\*/

private static int getApproximate(int x, int[] src) {

if (src == null) {

return -1;

}

if (src.length == 1) {

return src[0];

}

int minDifference = Math.abs(src[0] - x);

int minIndex = 0;

for (int i = 1; i < src.length; i++) {

int temp = Math.abs(src[i] - x);

if (temp < minDifference) {

minIndex = i;

minDifference = temp;

}

}

return src[minIndex];

}

}

原始数据是有序的：

package test;

import java.io.File;

public class CopyOfTest {

/\*\*

\* 假设原始数据是有序

\*/

static int[] src = new int[] { 10, 12, 21, 22, 25, 30, 34, 37, 39, 46, 51, 56, 70, 85 };

/\*\*

\* 目标值

\*/

static int x = 55;

public static void main(String[] args) {

System.out.println(getApproximate(x, src));

}

/\*\*

\* 获取接近值

\* @param x

\* @param src

\* @return

\*/

private static int getApproximate(int x, int[] src) {

if (src == null) {

return -1;

}

if (src.length == 1) {

return src[0];

}

int index = -1;

for (int i = 0; i < src.length; i++) {

if (src[i] > x) {

index = i;

break;

} else if (src[i] == x) {

return x;

}

}

if (index == -1) {

return src[src.length - 1];

} else if (index == 0) {

return src[0];

} else {

return x - src[index - 1] < src[index] - x ? src[index - 1] : src[index];

}

}

}

2.Oracle去重查询

在oracle中，可以利用SELECT语句和DISTINCT关键字实现查询去重。 在 SELECT 语句中使用 DISTINCT 子句来过滤结果集中的重复行。

3.get和post两种请求的区别

1.get是从服务器上获取数据，post是向服务器传送数据，

2.get传送的数据量较小，不能大于2KB。post传送的数据

量较大，一般被默认为不受限制。

1. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

4.在进行文件上传时只能使用post而不能是get。

4.说说springboot中有哪些注解

@RequestMapping：处理请求地址映射，分为类、方法级别

@ResponseBody：返回json数据

@Autoware：自动装配，把配置好的Bean拿来用，完成属性方法的组装

@RestController：标注控制层，包含@ResponseBody和@Controller

@Service：标注业务层

5.左连接，右连接，group by

左外连接：可以简称为左连接（LEFT JOIN），它会返回左表中的所有记录和右表中满足连接条件的记录。

右外连接：可以简称为右连接（RIGHT JOIN），它会返回右表中的所有记录和左表中满足连接条件的记录。

Group By：从字面意义上理解就是根据“By”指定的规则对数据进行分组，所谓的分组就是将一个“数据集”划分成若干个“小区域”，然后针对若干个“小区域”进行数据处理

6.static静态方法中能不能使用非静态变量，为什么，如果想使用该怎么办

不能；

任何程序最终都是在内存中执行的，变量只有在内存中占有一席之地时才能被访问，不在内存中的变量就不能被访问。由于静态成员在类加载的时候就会被分配内存，而非静态成员则不会；

非静态成员（变量和方法）属于类的对象，所以只有在类的对象产生（创建类的实例）时才会分配内存，然后通过类的对象（实例）去访问。

7.抽象类和接口的区别

接口是一种规范，抽象类体现的是一种模板式设计。

接口里只能包含抽象方法、静态方法、默认方法和私有方法，接口不能为普通方法提供方法实现；抽象类则可以包含普通方法。

接口里只能定义静态常量；抽象类里则既可定义普通成员变量，也可以定义静态常量。

接口里不包含构造器；抽象类里可以包含构造器，抽象类里的构造器是让其子类调用这些构造器来完成属于抽象类的初始化操作。

接口里不能包含初始化块；抽象类可以包含初始化块。

一个类最多只能有一个直接父类，包括抽象类；但一个类可以直接实现多个接口，通过实现多个接口可以弥补Java单继承的不足。

8.ajax能不能请求xml文件

可以，

var xhr = new XMLHttpRequest;

xhr.open('get','./test.xml');

xhr.send()

xhr.onreadystatechange = function(){

if(xhr.readyState === 4){

if(xhr.status>=200 && xhr.status<300){

// var res = xhr.responseText;

// 如果ajax请求的是xml文件 - 接收响应的数据 - xhr.responseXML

// 获取到的是一个文档 - 类似于我们DOM中的document

var res = xhr.responseXML

// console.log(res);

// 可以像操作dom一样去操作响应的xml文档

var eles = res.getElementsByTagName('goods')

// console.log(eles);

// 遍历伪数组获取到每个数据

for(var i=0;i<eles.length;i++){

// console.log(eles[i]);

var goodsnameTag = eles[i].getElementsByTagName('goodsname')[0]

// console.log(goodsnameTag);

console.log(goodsnameTag.innerHTML);

}

}

}

}

9.xml文件的读取

Dom读取

Dom4j读取

JDom读取

Sax读取

properties的读取

10.多表连接查询

1.内连接 inner join 特点：两个表交叉的部分被查询出来

2.①左外连接 left[outer] join [outer]可以被省略

特点：左表中的记录全部出现在结果中，如果对应的某行记录在右表中没有对应关系，则右表会自动补空（值）

3.②右外连接 right[outer] join

特点：右表中的记录全部出现在结果中，如果对应的某行记录在左表中没有对应关系，则左表会自动补空

4.交叉连接 cross join

没有where条件的交叉连接将产生连接表所涉及的笛卡尔积，即TableA的行数\*TableB的行数的结果集。如果带where，返回或显示的是匹配的行数。

5.where外键条件多表查询 多个条件之间用and连接

特点：相当于内连接，效果一致