**软件学院2019级高级程序设计语言试卷A**

**参考答案与评分标准**

**一.填空（30分，每题2分）**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **A** | **B** | **D** | **D** | **A** | **B** | **C** | **A** | **C** | **D** |
| **11** | **12** | **13** | **14** | **15** |  | | | | |
| **A** | **B** | **B** | **D** | **C** |

**二. 阅读题（40分）**

**1.**

**评分标准（共8分）**

sduds

\*sds\*

\*\*s\*\*

\*sds\*

sduds

输出：5行5列 2分

\*和sdu组成：2分

漏斗形状：2分

sdu顺序正确：2分

**2.**

2020

2021Java

2020

Kbwb3131

**评分标准（共8分）**

每行2分，格式不对扣1到2分

**3.**

1 3

2 3

1 2 3

3 2 1

**评分标准（共8分）**

每行2分，格式不对扣1到2分

**4.**

1 2 3

1 2 4

1 3 4

2 3 4

**评分标准（共8分）**

每行2分，错一个数字扣1分

**5.**

Finally in f()

12

Finally in f()

Caught OtherException

Improper score value

Finally in f()

-1

Caught OtherException

**评分细则：（共8分）**

每组2分，有错扣1分，全错扣2分，如果3行只有Finally in f()，不得分

**三. 写程序**

**1. 参考答案：**

import java.util.Scanner;

public class First {

public static void main(String[] args) {

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

int start=scan.nextInt();

int end=scan.nextInt();

int gap=(end/100\*60+end%100)-(start/100\*60+start%100);

System.*out*.println(gap/60\*100+gap%60);

}

}

**评分细则：共6分**

完成输入2分，gap计算正确2分，正确输出2分

**2.**自行车（Bike）和汽车（Car）都是一种交通工具（Vehicle），不同交通工具重量不同，能够一次行驶的里程也不同。请完善如下程序，使程序能够正确运行，并得到正确的输出。（**12分**）

abstract class Vehicle {

private int weights; // 交通工具重量，单位：千克

public Vehicle(int weights) {

this.weights = weights;

}

public int getWeights() {

return this.weights;

}

//定义抽象方法distance()，能够被子类使用（2分）

|  |
| --- |
| public abstract int distance(); |

}

class Bike extends Vehicle {

//写出类Bike的构造器，能够被TestW2类的main（）方法正确使用（2分）

|  |
| --- |
| public Bike(int weights) {  super(weights);  } |

public int distance() {//自行车能够一次行驶50公里

return 50;

}

}

class Car extends Vehicle {

private int volume; // 汽油容量，单位：升

//写出类Car的构造器，能够被TestW2类的main（）方法正确使用（2分）

|  |
| --- |
| public Car(int weights, int volume) {  super(weights);  this.volume = volume;  } |

//写出类Car的distance()方法，计算行驶里程，算法为volume乘以10（2分）

|  |
| --- |
| public int distance() {  return volume \* 10;  } |

}

public class TestW2 {

public static void main(String[] args) {

//定义Vehicle类型的数组ve\_arr（2分）

|  |
| --- |
| Vehicle[] ve\_arr = new Vehicle[2]; |

ve\_arr[0] = new Bike(20);

ve\_arr[1] = new Car(1000, 50);

for(int i = 0; i<ve\_arr.length; i++) {

//通过方法计算，得到正确的控制台输出结果，直接字符串输出不得分（2分）

|  |
| --- |
| System.out.println("ve\_arr["+i+"]的重量为："+ve\_arr[i].getWeights()+ "千克，能够行驶"+ve\_arr[i].distance()+"公里"); |

}

}

}

控制台程序输出结果：

ve\_arr[0]的重量为：20千克，能够行驶：50公里

ve\_arr[1]的重量为：1000千克，能够行驶：500公里

**3.**根据一个具有2个整数类型作为元素的链表，完成所需要的方法（**12分**）。示例如下，



class ListNode{ //为方便，属性定义为public，可以不用getter方法直接使用。

public int data1;

public int data2;

public ListNode link;

}

public class List {

private ListNode head;// 指向首元素

public int sum() {

//返回链表中所有data1和data2元素的数值和。（**2分**）

//如图例中，返回整数42

|  |
| --- |
| int count = 0;  ListNode position = head;  while (position != null) {  count += (position.data1 + position.data2); **—————1分，其余部分1分**  position = position.link;  }  return count; |

}

public void deleteItem(int m, int n) {

//删除链表中“data1==m并且data2==n”的Node（**4分**）

//如图例中，调用deleteItem（2,12），删除第二个Node

|  |
| --- |
| ListNode position = head;  ListNode current = position;// **初始化1分**  while (position!=null && position.link != null) {  current = position; // **循环 1分**  if (position.link.data1 == m && position.link.data2 == n)  position.link = position.link.link;  else  position = position.link;// **if语句 1分**  }  if(head!= null && head.data1 == m && head.data2 == n)  head = head.link;  if (position!=null && position.data1 == m && position.data2 == n)  current.link = null; // 头尾删除，**1分** |

}

public void insertItem(int m, int n, int k) {

//新建“data1==m，data2==n”的Node,并插入“data1==k”的第一个node后面（**3分**）

//如图例中，调用insertItem(4, 10, 2），将在第二个Node后面插入一个新的Node

|  |
| --- |
| ListNode position = head;  while (position != null) { //**循环1分**  if (position.data1 == k) {  position.link = new ListNode(m, n, position.link); //**if语句1分**  break;  }  position = position.link; //**1分**  } |

}

public void exchangeItem() {

//找出data1数值最大和data2数值最大的Node（若不同Node中的数值一样大，找出其中一个即可），将这两个Node在List中互换（**3分**）

//如图例中，第一个Node和第三个Node互换

|  |
| --- |
| if(head == null)  return;  ListNode Max\_data1 = head, Max\_data2 = head ;  ListNode position = head;  int temp\_data1 = position.data1, temp\_data2 =position.data2;//**初始化1分**  while (position != null) {  if (temp\_data1 < position.data1) {  temp\_data1 = position.data1;  Max\_data1 = position;  }  if (temp\_data2 < position.data2) {  temp\_data2 = position.data2;  Max\_data2 = position;  }  position = position.link; //**循环1分**  }  int temp = Max\_data1.data2;  Max\_data1.data1=Max\_data2.data1;  Max\_data1.data2=temp\_data2;  Max\_data2.data1=temp\_data1;  Max\_data2.data2=temp; //**互换 1分** |

}

}