Cat类定义：

public class Cat {

private String name;

private int weight;

public Cat(String name,int weight){

this.name=name

this.weight=weight:

}

public String toString(){

return “我是”+name+“体重：“+weight+”喵喵喵”；}

1.使用ArrayList集合存储5个Cat类对象，并查找是否包含目标对象，如果存在则删除，不存在则输出“没有该猫咪”

解答步骡:

(1)创建ArrayList类对象，添加5个Cat类元素，并使用Iterator类遍历输出，注意使用泛型，使代码洁。

(2)通过遍历的方式，依次比较每个Cat对象的name属性是否为"Ann，，如果是，则删除;如果遍历结束都没有找到，则输出“没有该猫咪”

2.使用HashSet集合存储5个Cat类对象，给出Cat类的equals和hashCode函数，实现相同猫咪的判断，

3.使用TreeSet集合存储5个Cat类对象，给出Cat类的compareTo函数，实现按照体重排序。

4.使用HashMap存储5个oat类对象及每只猫咪对应的品种，实现遍历输出，查找修改品种信息等操作。



1.从键盘循环录入学生信息，输入格式为:学号名字:

将学生信息保存到“stu.txt”文件中，一个学生信息占据一行数据，当用户输入”over’时停止输入。要求使用字符缓冲输出流写出数据。结束输入后用缓冲流将文件内容输出。

解答步骤:

(1)使用Scanner类进行键盘录入数据。(2)创建字符缓冲输出流对象关联“stu.txt”文件(BufferedWriter)类

(3)使用循环不停的接收输入的学生数据。

(4)判断输入的内容是否是“over，是则终止循环停止输入。否则就数据写出到文件中。

(5)注意使用try-catch-finally进行异常处理

如果是输入流，该如何实现。



2、编写一个完整的 Java 程序，具体要求如下:

(1)定义银行卡 Bankcard，该类包含:

a.3 个 private 属性的成员变量:卡号，户主，余额。

b.无参构造函数，不进行任何操作。

c.带3个参数的构造方法，用于初始化卡号，户主，余额。

d.重写 tostring()方法用于返回 Bankcard 的详细信息。

e.定义余额的读写方法 Get()和 set()，实现存款和职款操作，注意取款时候

余额不足要有提示。

(2)定义信用卡 &reditcard，继承 Bankcard类，该类增加:

a.1个 private 的属性:信用卡额外有透支额度。

b.带4个参数的构造方法，用于初始化卡号，户主，余额和透支额度。

c.重写 tostring()方法用于返回 creditcard 的详细信息。

d.重写取款方法，信息卡可透支，超出余额和透支金额之和时，要有提示。

(3)编写 Test 测试类程序，在 main 函数中实现如下要求:

创建 &ankcard 对象 b1，进行测试。

创建 creditcard 对象 c1，进行测试。

// Bankcard类

class Bankcard {

private String cardNumber;

private String cardHolder;

private double balance;

public Bankcard() {}

public Bankcard(String cardNumber, String cardHolder, double balance) {

this.cardNumber = cardNumber;

this.cardHolder = cardHolder;

this.balance = balance;

}

@Override

public String toString() {

return "Bankcard [cardNumber=" + cardNumber + ", cardHolder=" + cardHolder + ", balance=" + balance + "]";

}

public double getBalance() {

return balance;

}

public void setBalance(double newBalance) {

this.balance = newBalance;

}

public void deposit(double amount) {

this.balance += amount;

}

public void withdraw(double amount) {

if (amount > balance) {

System.out.println("余额不足，取款失败！");

} else {

this.balance -= amount;

}

}

}

// Creditcard类

class Creditcard extends Bankcard {

private double creditLimit;

public Creditcard(String cardNumber, String cardHolder, double balance, double creditLimit) {

super(cardNumber, cardHolder, balance);

this.creditLimit = creditLimit;

}

@Override

public String toString() {

return "Creditcard [cardNumber=" + super.cardNumber + ", cardHolder=" + super.cardHolder + ", balance=" + super.getBalance() + ", creditLimit=" + creditLimit + "]";

}

@Override

public void withdraw(double amount) {

if (amount > getBalance() + creditLimit) {

System.out.println("超出透支额度，取款失败！");

} else {

setBalance(getBalance() - amount);

}

}

}

// Test测试类

public class Test {

public static void main(String[] args) {

Bankcard b1 = new Bankcard("1234567890", "Alice", 1000);

System.out.println(b1.toString());

b1.deposit(500);

b1.withdraw(1200);

System.out.println(b1.toString());

Creditcard c1 = new Creditcard("0987654321", "Bob", 2000, 500);

System.out.println(c1.toString());

c1.deposit(300);

c1.withdraw(2500);

System.out.println(c1.toString());

}

}