1. 编写一个学生类 Student ，要求：

(1) 学生类 Student 属性有：

id : long型，代表学号

name : String类对象，代表姓名

age : int型，代表年龄

sex : boolen型，代表性别（其中：true表示男，false表示女）

phone : String类对象，代表联系电话

(2) 学生类 Student的方法有：

Student(long i , String n , int a , boolean s , String p) : 有参构造函数，形参表中的参数分别初始化学号、姓名、年龄、性别和联系电话。

int getAge() ( ) : 获取年龄作为方法的返回值。

boolean getSex( ) : 获取性别作为方法的返回值。

String getPhone ( ) : 获取联系电话作为方法的返回值。

public String toString( ) : 以 姓名：联系电话 学号 的形式作为方法的返回值。

(3)学生类Student有main方法，可以实例化一个学生“张三”，张三的学号是10001，年龄为23岁，性别男，联系电话0411-88888888；然后输出toString()方法；

**public** **class** Student

{

**long** id; //学号

String name; //姓名

**int** age; //年龄

**boolean** sex; //性别 true：男；false：女；

String phone; //联系电话

Student(**long** i, String n, **int** a, **boolean** s, String p) {

id = i;

name = n;

age = a;

sex = s;

phone = p;

}

**int** getAge() {

**return** age;

}

**boolean** getSex() {

**return** sex;

}

String getPhone() {

**return** phone;

}

**public** String toString() {

**return** name + " :联系电话 " + phone + "，学号为 "+id;

}

**public** **static** **void** main(String[] args){

Student s = **new** Student(10001,"张三",23,**true**,"0411-88888888");

System.***out***.println(s);

}

}

2. 设计一个类，该类用于表示现代中国的一个典型家庭：father,mother,child.并为其添加一个无参的构造函数和带三个参数的构造函数以及三个读写方法和一个get(int member )方法,

当member=FATHER时，返回父亲的姓名；

member=MOTHER时，返回母亲的姓名；

member=CHILD时，返回孩子的姓名；

其中FATHER,MOTHER,CHILD 为常数。

**public** **class** Family

{

**private** String father;

**private** String mather;

**private** String child;

**public** **static** **final** **int** ***FATHER*** = 1;

**public** **static** **final** **int** ***MATHER*** = 2;

**public** **static** **final** **int** ***CHILD*** = 3;

**public** Family(){}

**public** Family(String father, String mather, String child){

**this**.father = father;

**this**.mather = mather;

**this**.child = child;

}

**public** String get(**int** member){

String s=**null**;

**if**(member == ***FATHER***){

s = father;

}

**if**(member == ***MATHER***){

s = mather;

}

**if**(member == ***CHILD***){

s = child;

}

**return** s;

}

**public** String getFather() {

**return** father;

}

**public** **void** setFather(String father) {

**this**.father = father;

}

**public** String getMather() {

**return** mather;

}

**public** **void** setMather(String mather) {

**this**.mather = mather;

}

**public** String getChild() {

**return** child;

}

**public** **void** setChild(String child) {

**this**.child = child;

}

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

Family family = **new** Family("1","2","3");

String str = family.get(***CHILD***);

System.***out***.println(str);

}

}

3. Josephu问题为：设编号为1,2,3……n的n个人围坐一圈，约定编号为k(1<=k<=n)的人从1开始报数，到m的那个人出列，它的下一位又从1开始报数，数到m的人又出列。以此类推，直到所有人出列为止，由此产生一个出对编号的序列。

解题：

用一个不带头结点的循环链表来处理Josephu问题，先构成一个有n个结点的单循环链表，然后由k结点起从1开始计数，计到m时，对应结点的人从链表中删除，然后再从被删除结点的下一个结点又从1开始计数，直到最后一个结点从链表中删除算法结束。

**public** **class** JosePhu

{

**public** **static** **void** main(String[] args) {

CycLink cyclink=**new** CycLink();

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

System.***out***.println("请输入小孩的个数:");

**int** length = scan.nextInt();

cyclink.setLen(length);//链表长度

cyclink.createLink();

cyclink.setK(2);//从第几个人开始数

cyclink.setM(2);//数几下

cyclink.show();

cyclink.play();

}

}

**class** Child{

**int** no;

Child nextChild=**null**;

**public** Child(**int** no){

//给一个编号

**this**.no=no;

}

}

//单向环形链表

**class** CycLink{

//先定义一个指向链表第一个小孩的引用

//指向第一个小孩的引用，不能动

Child firstChild=**null**;

Child temp=**null**;

**int** len=0;//表示共有多少个小孩

**int** k=0;

**int** m=0;

//设置m数几下

**public** **void** setM(**int** m){

**this**.m=m;

}

//设置环形链表大小

**public** **void** setLen(**int** len){

**this**.len=len;

}

//设置从第几个人开始数数

**public** **void** setK(**int** k){

**this**.k=k;

}

//开始play

**public** **void** play(){

Child temp=**this**.firstChild;

//1.先找到开始数数的人

**for**(**int** i=1;i<k;i++){

temp=temp.nextChild;

}

**while**(**this**.len!=1){

//2.数m下

**for**(**int** j=1;j<m;j++){

temp=temp.nextChild;

}

//找到要出圈的前一个小孩

Child temp2=temp;

**while**(temp2.nextChild!=temp){

temp2=temp2.nextChild;

}

//3.将数到m的小孩，退出圈

temp2.nextChild=temp.nextChild;

//让temp指向下一个数数的小孩

temp=temp.nextChild;

**this**.len--;

}

//最后一个小孩

System.***out***.println("最后出圈的小孩:"+temp.no);

}

//初始化单向环形链表

**public** **void** createLink(){

**for**(**int** i=1;i<=len;i++){

**if**(i==1){

//创建第一个小孩

Child ch=**new** Child(i);

**this**.firstChild=ch;

**this**.temp=ch;

}**else**{

//创建最后一个小孩

**if**(i==len){

Child ch=**new** Child(i);

temp.nextChild=ch;

temp=ch;

temp.nextChild=**this**.firstChild;

}**else**{

//继续创建小孩

Child ch=**new** Child(i);

temp.nextChild=ch;

temp=ch;

}

}

}

}

//打印该环形链表

**public** **void** show(){

//定义一个跑龙套

Child temp=**this**.firstChild;

**do**{

System.***out***.print(temp.no+" ");

temp=temp.nextChild;

}**while**(temp!=**this**.firstChild);

}

}

4. 设计一个星座类，并为其添加一个带参数的构造函数，

class Constellation{

private String birthday;

public Constellation(String birthday){

}

String getConstellation(){

返回 星座

}

int getBornYear(){

}

int getBornMonth(){

}

int getBornDay(){

}

}

出生日期与星座关系如下：

1.20---2.18 水瓶座

2.19----3.20 双鱼座

3.21----4.19. 牧羊座

4.20---5.20 金牛座

5.21-----6.21双子座

6.22.----7.22巨蟹座

7.23----8.22 狮子座

8.23----9.22 处女座

9.23.----10.23天秤座

10.24-----11.22天蝎座

11.23---12.21 射手座

12.22.----1.19 山羊座

**public** **class** ConstellationPractice

{

**public** **static** **void** main(String[] args) {

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

System.***out***.println("请输入你的出生年份：");

**int** year = scan.nextInt();

System.***out***.println("请输入你的出生月份：");

**int** month = scan.nextInt();

System.***out***.println("请输入你的出生日期：");

**int** day = scan.nextInt();

String birthday = year+"-"+month+"-"+day;

Constellation constellation = **new** Constellation(year,month,day);

System.***out***.println("您的星座为 "+constellation.getConstellation()+"!");

}

}

**class** Constellation{

**private** String birthday;

**private** **int** year;

**private** **int** month;

**private** **int** day;

**public** Constellation(**int** year,**int** month,**int** day){

**this**.year = year;

**this**.month = month;

**this**.day = day;

birthday = year+"-"+month+"-"+day;

System.***out***.println("您的生日为"+birthday+"!");

}

String getConstellation(){

//返回 星座

String constellation = "";

**if**((month==1 && day >=20 ) || (month ==2 && day<=18)){

constellation= "水瓶座";

}**else** **if**((month==2 && day >=19 ) || (month ==3 && day<=20)){

constellation= "双鱼座";

}**else** **if**((month==3 && day >=21 ) || (month ==4 && day<=19)){

constellation= "牧羊座";

}**else** **if**((month==4 && day >=20 ) || (month ==5 && day<=20)){

constellation= "金牛座";

}**else** **if**((month==5 && day >=21 ) || (month ==6 && day<=21)){

constellation= "双子座";

}**else** **if**((month==6 && day >=22 ) || (month ==7 && day<=22)){

constellation= "巨蟹座";

}**else** **if**((month==7 && day >=23 ) || (month ==8 && day<=22)){

constellation= "狮子座";

}**else** **if**((month==8 && day >=23 ) || (month ==9 && day<=22)){

constellation= "处女座";

}**else** **if**((month==9 && day >=23 ) || (month ==10 && day<=23)){

constellation= "天秤座";

}**else** **if**((month==10 && day >=24 ) || (month ==11 && day<=22)){

constellation= "天蝎座";

}**else** **if**((month==11 && day >=23 ) || (month ==12 && day<=21)){

constellation= "射手座";

}**else** **if**((month==12 && day >=22 ) || (month ==1 && day<=19)){

constellation= "山羊座";

}

**return** constellation;

}

**int** getBornYear(){

**return** year;

}

**int** getBornMonth(){

**return** month;

}

**int** getBornDay(){

**return** day;

}

}