1. 使用数组编程，按下列格式输出数字：

1 3 6 10 15
2 5 9 14
4 8 13
7 12
11

```
public class Main
{
        public static void main(String[] args)
    {
        int a[][] = new int[][] { { 1, 3, 6, 10, 15 }, { 2, 5, 9, 14 }, { 4, 8, 13 }, { 7, 12 }, { 11 } };
        for (int i = 0; i < a.length; i++)
        {
            for (int k = 0; k < a[i].length; k++)
            {
                System.out.print(a[i][k] + " ");
            }
            System.out.println();
        }
    }
}
```

2. 定义一个学生类(Student)，属性有 private 的名字(name), public 的年龄(age);
设置 name 和 age 属性的方法：setName(),getName(); setAge(),getAge(),以及显示 name 和 age 值的方法 showName(),showAge();
编写 Application，创建一个学生对象，设置 name 和 age 属性值，并显示 name 和 age.

```
public class student {
    private String name;
    public int age;
    private void setName(String name)
    {
        this.name=name;
    }
    private void setAge(int age)
    {
        this.age=age;

    }
    public String getName()
    {
        return this.name;
    }
    public int getAge()
```

```
    {
        return this.age;
    }
    public void showName(String name)
    {
        System.out.println(name);
    }
        public void showAge(int age)
    {
        System.out.println(age);
    }
}
public class Main {
    public static void main(String[] args) {
        student astudent=new student();
        astudent.age=19;
        astudent.name="wanghua";
        astudent.showAge(astudent.getAge());
        astudent.showName(astudent.getName());

    }
}
```

1. 设计一个复数类 complex，包括：
   a. 成员变量：复数的实部和虚部;
   b. 构造函数：带参数和不带参数的构造函数，对成员变量进行初始化；
   c. 成员方法：复数的加、减、乘运算。

```
public class complex {
    private double shibu;
    private double xubu;
    public complex(){
        this(0.0,0.0);
    }
    public complex(double ashibu,double axubu)
    {
        shibu=ashibu;
        xubu=axubu;
    }
    public double getshibu(){
    return shibu;}
    public double getxubu(){
    return xubu;}
    public void setshibu(double newshibu){
    shibu=newshibu;}
    public void setxubu(double newxubu){
```

```
    xubu=newxubu;}
    public void showcom(complex a){
        if(a.xubu>0)
            System.out.println(a.shibu+"+"+a.xubu+"i");
        else
            System.out.println(a.shibu+a.xubu+"i");
    }
    public complex comadd(complex a,complex b,complex c){

        c.shibu=a.shibu+b.shibu;
        c.xubu=a.xubu+b.xubu;
        return c;
    }
    public complex comsub(complex a,complex b,complex c){
        c.shibu=a.shibu-b.shibu;
        c.xubu=a.xubu-b.xubu;
        return c;
    }
    public complex commul(complex a,complex b,complex c){

        c.shibu=(a.shibu*b.shibu)-(a.xubu*b.xubu);
        c.xubu=(a.shibu*b.xubu)+(a.xubu*b.shibu);
        return c;
    }
    public static void main(String[] args) {
        complex a,b,c;
        a=new complex(5.5,6.2);
        b=new complex(10.1,3.4);
        c=new complex(0,0);
        System.out.print("a+b=");
        c.comadd(a,b,c);
        c.showcom(c);
        System.out.print("a-b=");
        c.comsub(a, b, c);
        c.showcom(c);
        System.out.print("a*b=");
        c.commul(a, b, c);
        c.showcom(c);
    }
}
```

1．定义一个 Point 类。该类具有以下特点：

a. int 型变量 x,y 表示屏幕坐标系上的一个点;

b. 两种构造方法实现对象的初始化：Point(int x, int y)和 Point(Point p);

c. 计算两个 Point 对象之间距离的方法 distance(Point a, Point b);

d. 获得当前坐标 x,y 值的方法 getX(),getY();

e. 设置坐标 x,y 值的方法 setX(),setY();

f. 重写 Point 类的 toString()方法，以格式(x,y)输出当前点的字符串;

g. 统计应用程序中 Point 类对象的个数（使用 Point 类的类变量）。

```
package javaapplication1;
import static java.lang.Math.*;
public class Point {
    private int x,y;
    private static int count=0;
    public Point()
    {
        this(0,0);
        count++;
    }
    public Point(int a, int b)
    {
        x=a; y=b;
        count++;
    }
    public int getx(){
    return x;}
    public int gety(){
    return y;}
    public int getcount()
    { return count;}
    public void setx(int newx){
    x=newx;}
    public void sety(int newy){
    y=newy;}
    public static double distance(Point a, Point b){

        return sqrt(pow((a.x-b.x),2)+pow((a.y-b.y),2));
    }
    public String toString()
    {
        return("("+x+","+y+")");
    }
    public static void main(String[] args) {
          Point m,n;
        m=new Point(2,3);
        n=new Point(5,4);
        m.getx();
        m.gety();
```

```
            n.getx();
            n.gety();
           System.out.println("the distance:"+ distance(m,n));
            System.out.println(m);
            System.out.println(n);
              System.out.println(count);
        }
}
```

1. 声明一个 person 类，包含：

a) 属性：姓名，年龄，性别；

b) 方法：设置 person 信息，显示 person 信息(重写 toString 方法)；

```
public class person {
    private String name,sex;
    private int age;
    public void set(String name,int age,String sex)
    {
        this.name=name;
        this.sex=sex;
        this.age=age;
    }
    public String getName(){return name;}
    public int getAge(){return age;}
    public String getSex(){return sex;}
    public person()
    {}
    public person(String name,int age,String sex)
    {
        this.name=name;
        this.sex=sex;
        this.age=age;
    }
    public String toString()
    {
        return new String("His/Her name is:"+name+"\nHis/Her age is:"+age+"\nHis/Her sex is:"+sex);
    }
}
```

2. 声明一个 student 类，作为 person 类的子类。包含：

a) 属性：学号，英语成绩，数学成绩，物理成绩，平均成绩，总成绩；

b) 方法：设置学号，设置各科成绩，计算平均成绩，计算总成绩，显示 student 信息(重写 toString 方法)；

```
public class student extends person{
    private int number,EngGrade,MathGrade,PhyGrade,Aver,Total;
    public student(){}
```

```
    public student(String name,int age,String sex,int number,int EngGrade,int
MathGrade,int PhyGrade){
        this.set(name,age, sex);
        this.number=number;
        this.EngGrade=EngGrade;
        this.MathGrade=MathGrade;
        this.PhyGrade=PhyGrade;
    }
    public void setnumber(int number){this.number=number;}
    public void setEngGrade(int EngGrade){this.EngGrade=EngGrade;}
    public void setMathGrade(int MathGrade){this.MathGrade=MathGrade;}
    public void setPhyGrade(int PhyGrade){this.PhyGrade=PhyGrade;}
    public int getAver(){return this.Aver=(EngGrade+MathGrade+PhyGrade)/3;}
    public int getTotal(){return this.Total=EngGrade+MathGrade+PhyGrade;}
    public String toString(){
        return new String("The name is:"+this.getName()+"\nThe age
is:"+this.getAge()+"\nThe sex is:"
                +this.getSex()+"\nThe number is:"+number+"\nThe English grade
is:"+EngGrade+"\nThe Math grade is:"
                +MathGrade+"\nThe Physics grade is:"+PhyGrade+"\nThe average
grade is:"
                +this.getAver()+"\nThe total grade is:"+this.getTotal());
    }

}
```

3. 声明一个测试类 test, 生成若干个 student 类的对象，分别计算他们的各项成绩，并输出其信息。

```
public class Main {
    public static void main(String[] args)
    {
        person p1 = new person("panyy", 19, "girl");
        System.out.println(p1.toString());
        person p2 = new person();
        p2.set("lzx", 21, "gril");
        System.out.println(p2.toString());
        student stu1 = new student("zzy", 20, "boy", 1111000207, 80, 70, 60);
        System.out.println(stu1.toString());
        student stu2 = new student();
        stu2.set("lzy", 19, "girl");
        stu2.setnumber(111101234);
        stu2.setEngGrade(98);
        stu2.setMathGrade(88);
        stu2.setPhyGrade(89);
        System.out.println(stu2.toString());
```

```
    }
}
```

1. 声明一个抽象类 comparable, 用于对两个不同对象的比较，并满足：

a) 包含一个抽象方法 int compareTo(comparable b); 若小于对象 b 则返回-1,大于则返回 1, 相等则返回 0.

b) 声明两个子类，一个为复数类 complex，另一个为矩形类 rect; 在子类中分别实现父类 comparable 的抽象方法.

c) 复数类的大小比较以模为基准，而矩形类的比较以面积为基准。

```
public abstract class comparable {
    public abstract int compareTo(Object b);
}
public class complex extends comparable{
    private double real,image;
    public complex(){real=0.0;image=0.0;}
    public complex(double a,double b){real=a;image=b;}
    @Override
    public int compareTo(Object T){
        complex b=(complex)T;
        double temp=real*real+image*image-b.real*b.real-b.image*b.image;
        if(temp>0)return 1;
        else if(temp<0)return -1;
        else return 0;
    }
}
public class rect extends comparable{
    double length,width;
    public rect(){length=0.0;width=0.0;}
    public rect(double a,double b){length=a;width=b;}
    @Override
    public int compareTo(Object T){
        rect b=(rect)T;
        double temp=length*width-b.length*b.width;
        if(temp>0)return 1;
        else if(temp<0)return -1;
        else return 0;
    }
}
public class Main {
    public static void main(String[] args) {
        complex a=new complex(5,6);
        complex b=new complex(6,7);
        System.out.println(a.compareTo(b));
        System.out.println(b.compareTo(a));
```

```
        rect c=new rect(7,7);
        rect d=new rect(6,6);
        System.out.println(c.compareTo(d));
        System.out.println(d.compareTo(c));
        rect e=new rect(6,6);
        System.out.println(d.compareTo(e));
    }
}
```

1. 定义一个接口EqualDiag，表示具有等斜边的图形对象，其中包含：
   a. 方法getDiag，用来计算图形的斜边的长度;
   b. 方法getArea，用来计算图形的面积;
   c. 方法compareTo, 用来比较两个对象的大小(按照斜边的大小进行比较);
2. 定义一个矩形类Rectangle，再派生一个正方形类Square:
   a. 在矩形类和正方形类中实现接口EqualDiag;

```
public interface EqualDiag {
double getDiag();
double getArea();
int compareTo(Object b);
}
public class Rectangle implements EqualDiag {
    double length,width;
    public Rectangle(){length=0.0;width=0.0;}
    public Rectangle(double a,double b){length=a;width=b;}
    @Override
    public double getDiag(){return Math.sqrt(width*width+length*length);}
    @Override
    public double getArea(){return length*width;}
    @Override
    public int compareTo(Object T){
        Rectangle b=(Rectangle)T;
        double temp=getDiag()-b.getDiag();
        if(temp>0)return 1;
        else if(temp<0)return -1;
        else return 0;
    }
}
public class Square extends Rectangle {
    Square(){super();}
    Square(double a){width=length=a;}
}
public class Main {
    public static void main(String[] args) {
        Rectangle a=new Rectangle (4,6);
        Square b=new Square(5);
```

```
        System.out.println("Rectangle a'Daig="+a.getDiag());
        System.out.println("Rectangle a'Area="+a.getArea());
        System.out.println("Square b'Daig="+b.getDiag());
        System.out.println("Square b'Area="+b.getArea());
        System.out.println(a.compareTo(b));
        System.out.println(b.compareTo(b));
        System.out.println(b.compareTo(a));
    }
}
```

1. 编写求解几何图形（如直线、矩型，圆形）的周长和面积的应用程序，要求使用接口实现多重继承和多态技术。

提示：声明Iperimeter和IArea分别表示周长接口和面积接口；声明抽象类Shape.

```
public interface Iperimeter {
public double getDiag();
}
public interface IArea {
public double getArea();
}
public abstract class shape implements IArea,Iperimeter{
}
public class Rectangle extends shape{
    double length,width;
    public Rectangle(double length,double width){
        this.length=length;
        this.width=width;
    }
    public double getArea(){
    return length*width;
}
public double getDiag(){
    return 2*(length+width);
}
}
public class line extends shape{
    double l;
    public line(double l){
        this.l=l;
    }
    public double getArea(){
    return 0;
}
public double getDiag(){
    return l;
```

```
}
}
public class Circle extends shape{
double r;
public Circle(double r){
    this.r=r;
}
public double getArea(){
    return Math.PI*r*r;
}
public double getDiag(){
    return Math.PI*2*r;
}
}
public class Main {
    public static void main(String[] args) {
        shape a=new Circle(4);
        shape b=new line(5.3);
        shape c=new Rectangle(4,5);
        System.out.println("Circle a'Area="+a.getArea());
        System.out.println("Circle a'Diag="+a.getDiag());
        System.out.println("line b'Area="+b.getArea());
        System.out.println("line b'Diag="+b.getDiag());
        System.out.println("Rectangle c'Area="+c.getArea());
        System.out.println("Rectangle c'Diag="+c.getDiag());
    }
}
```

1. 编写一个矩形类rectangle：

   a. 成员变量包括：长和宽；

   b. 方法包括：计算面积、计算周长、设置长和宽的值（从键盘输入）；

   c. 重写toString()方法，显示rectangle对象的详细信息；

   d. 重写equals方法，如果两个矩形的面积和周长都相等，则认为它们是同一。

```
public class rectangle
{
    private double chang, kuan, area, zhchang;
    public void set(double chang, double kuan)
    {
        this.chang = chang;
        this.kuan = kuan;
    }
    public double getchang() { return chang; }
    public double getkuan() { return kuan; }
    public double getarea()
    {
```

```
        return this.area = chang * kuan;
    }
    public double getzhchang()
    {
        return this.zhchang = 2 * (chang + kuan);
    }
    public rectangle()
    { }
    public rectangle(double chang, double kuan)
    {
        this.chang = chang;
        this.kuan = kuan;
    }
    public String toString()
    {
        return new String("the chang=" + chang + "\nthe kuan=" + kuan + "\nthe
area=" + this.getarea() + "\nthe zhouchang=" + this.getzhchang());
    }
    public boolean equals(rectangle obj)
    {
        if  (this.getarea()  !=  obj.getarea()  ||  this.getzhchang()  !=
obj.getzhchang())
            return false;
        else
            return true;
    }
}
public class main
{
        public static void main(String[] args)
    {
        rectangle r1 = new rectangle();
        r1.set(5.5, 6.6);
        rectangle r2 = new rectangle(9.7, 7.4);
        System.out.println(r1.toString());
        System.out.println(r2.toString());
        if (r1.equals(r2))
            System.out.println("two rectangle are equal");
        else
            System.out.println("two rectangle are not equal");
    }
}
```

1. 模仿文本文件复制的程序，编写对二进制文件进行复制的程序。

```
import java.io.*;
```

```
import java.util.Scanner;
public class CreatebinarysystemFile {
    public static void main(String args[]) throws FileNotFoundException
    {
        String fileName;
        System.out.println("please input fileName: ");
        Scanner in=new Scanner(System.in);
        fileName=in.nextLine();
        int number;
        System.out.println("please input the number of int: ");
        number=in.nextInt();
        int min,max;
        System.out.print("please input the arrange of int:\nmin=");
        min=in.nextInt();
        System.out.print("max=");
        max=in.nextInt();
                        DataOutputStream     out=new     DataOutputStream(new
FileOutputStream(fileName));
        try
        {
            int value;
            int no=0;
            value=(int)(Math.random()*100);
            for(no=1;no<number;)
            {
             if(min<=value&&value<=max)
                {
                    out.writeInt(value);
                    no++;
                 }
                value=(int)(Math.random()*100);
            }
        }
        catch(IOException e)
        {
            System.out.println("error");
        }
        try
        {
        out.close();
        }
        catch(IOException e)
        {
            System.out.println("error close");
```

```
        }
        new CopyMaker().copy(fileName,"D:/a.dat");
    }
}
import java.io.*;
public class CopyMaker {
    String source,destnation;
    DataInputStream src;
    DataOutputStream dest;
    public boolean openFile()
    {
        try
        {
            src=new       DataInputStream(new       BufferedInputStream(new
FileInputStream(source)));
        }
        catch(IOException e)
        {
            System.out.println("Problem reading "+source);
            return false;
        }
        try
        {

            dest=new      DataOutputStream(new      BufferedOutputStream(new
FileOutputStream(destnation)));
        }
        catch(IOException e)
        {
            System.out.println("Problem reading "+destnation);
            return false;
        }
        return true;
    }
    public boolean CopyFile()
    {
        try
        {
            int line;
            line=src.read();
            while(line!=-1)
            {
                dest.write(line);
                line=src.read();
```

```
            }
        }
        catch(IOException iox)
        {
            System.out.println("Problem reading or writing");
            return false;
        }
        return true;
    }
    public boolean closeFile()
    {
        boolean v=true;
        try
        {
            src.close();
        }
        catch(IOException e)
        {
            System.out.println("Program closing "+source);
            v=false;
        }
        try
        {
            dest.close();
        }
        catch(IOException e)
        {
            System.out.println("Program closing "+destnation);
            v=false;
        }
        return v;
    }
    public boolean copy(String s,String d)
    {
        source=s;
        destnation=d;
        return openFile()&&CopyFile()&&closeFile();
    }
}
```

2. 创建一个存储若干随机整数的文本文件。其中文件名、整数的个数及其范围均由键盘输入。

```
import java.io.*;
import java.util.Scanner;
public class CreateTextfile {
```

```
public static void main(String args[]) throws IOException
{
    String fileName;
    System.out.println("please input fileName: ");
    Scanner in=new Scanner(System.in);
    fileName=in.nextLine();
    int number;
    System.out.println("please input the number of int: ");
    number=in.nextInt();
    int min,max;
    System.out.print("please input the arrange of int:\nmin=");
    min=in.nextInt();
    System.out.print("max=");
    max=in.nextInt();
    BufferedWriter input=new BufferedWriter(new FileWriter(fileName));
    try
    {
        int value;
        int no=0;
        value=(int)(Math.random()*100);
        for(no=1;no<number;)
        {
         if(min<=value&&value<=max)
            {
                input.write(value);
                input.newLine();
                no++;
             }
            value=(int)(Math.random()*100);
        }
    }
    catch(IOException e)
    {
        System.out.println("error");
    }
    try
    {
    input.close();
    }
    catch(IOException e)
    {
        System.out.println("error close");
    }
}
```

}

1. 创建一学生类（包括：姓名、年龄、所在班级、密码），创建若干该类的对象并保存到文件a.dat中（密码不用保存），从文件a.dat读取对象显示在屏幕上。

```
import java.io.Serializable;
class Student implements Serializable{
    String name;
    int age;
    int grade;
    transient String secret;
    public Student(String name,int age,int grade,String secret){
        this.name=name;
        this.age=age;
        this.grade=grade;
        this.secret=secret;
    }
}
import java.io.*;
public class Main {

    public static void main(String[] args) throws IOException,ClassNotFoundException{
        Student student[]={
                new Student("赵宗懿",18,101,"zzy"),
                new Student("李宗霞",19,103,"lzx"),
                new Student("刘兆英",18,105,"lzy"),
                new Student("潘妍妍",20,107,"pyy")};
        ObjectOutputStream oos=new ObjectOutputStream(
                new FileOutputStream("a.dat"));
        for(int i=0;i<student.length;i++)
            oos.writeObject(student[i]);
        oos.close();
          for(int i=0;i<student.length;i++)
            student[i]=null;
        ObjectInputStream ois=new ObjectInputStream(
                new FileInputStream("a.dat"));
        for(int i=0;i<student.length;i++)
            student[i]=(Student)ois.readObject();  ois.close();
        for(int i=0;i<student.length;i++){
            if(i==student.length-1){
                System.out.println(student[i].name);
                System.out.println(student[i].age);
                System.out.println(student[i].grade);
                System.out.println(student[i].secret);
                System.out.println();
```

```
            }
            else{
                System.out.println(student[i].name);
                System.out.println(student[i].age);
                System.out.println(student[i].grade);
                System.out.println(student[i].secret);
                System.out.println();
            }
            }
    }
}
```

1. 设计一个学生Student类，包含学号(Sno),姓名(Name),所在系(Dept)等属性。创建若干Student类的对象，添加到一个向量Vector中，并遍历输出该向量各元素值。

```
public class Student {
    String Sno;
    String Name;
    String Dept;
    public Student()
    {}
    public Student(String Sno,String Name,String Dept)
    {
        this.Sno=Sno;
        this.Name=Name;
        this.Dept=Dept;
    }
    public String toString(Student s)
    {
        return (s.Name+" "+s.Sno+" "+s.Dept);
    }
    public void setName(String name) {
        Name = name;
    }
    public void setSno(String sno) {
        Sno = sno;
    }
    public void setDept(String dept) {
        Dept = dept;
    }
    public String getSno() {
        return Sno;
    }
    public String getName() {
        return Name;
```

```
    }
    public String getDept() {
        return Dept;
    }
    public String toString()
    {
        return (this.Sno+" "+this.Name+" "+this.Dept);
    }
}
import java.util.Vector;
public class test {
    public static void main(String args[])
    {
       Student s[] = new Student[3];
       s[0]=new Student("203","Liuzhao","zn");
       s[1]=new Student("198","Lizong","zn");
       s[2]=new Student("207","Panyan","zn");
       Vector<Student> stu=new Vector<Student>();
       stu.add(s[0]);
       stu.add(s[1]);
       stu.add(s[2]);
        for(int i=0;i<s.length;i++)
       {
            System.out.println(stu.get(i));
       }
    }
}
```

2. 已知：哈希表HashTable<Integer,String> hTable = new HashTable<Integer,String>()。编程遍历hTable，要求写出两种方式。

```
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;
public class test {
   public static void main(String args[])
   {
       Hashtable<Integer,String> hTable =
               new Hashtable<Integer,String>();
       hTable.put(1,"Panyanyan");
       hTable.put(2,"Pa");
       hTable.put(3,"Pa");
       System.out.println("first:");
       System.out.println(hTable);
       System.out.println("second:");
    Vector<Hashtable> v=new Vector<Hashtable>();
```

```
        v.add(hTable);
        System.out.println(v);
        System.out.println("third:");
    Enumeration<Hashtable> e;
        e=v.elements();
        while(e.hasMoreElements())
        {
            System.out.print(e.nextElement());
        }
    }
}
```

1. 编写一个文本文件拷贝程序，将文本文件a.txt拷贝到b.txt,
同时在屏幕上输出文件a.txt中的每一个单词(每个单词一行).

(提示：使用StringTokenizer类提取单词, 参见课本P.140的例4-22)

```
class CopyMaker {
        String sourceName="d:/a.txt",destName="d:/b.txt";
        BufferedReader source;
        BufferedWriter dest;
        String line;
        private boolean openFiles(){
            try{
                source=new BufferedReader(new FileReader(sourceName));
            }
            catch(IOException iox){
                System.out.println("Problem opening"+sourceName);
                return false;
            }
            try{
                dest=new BufferedWriter(new FileWriter(destName));
            }
            catch(IOException iox){
                System.out.println("Problem opening"+destName);
                return false;
            }
            return true;
        }
        private boolean copyFiles(){
            try{
                line=source.readLine();
                while(line!=null){
                    dest.write(line);
                    dest.newLine();
                    line=source.readLine();
```

```
                }
                }
            catch(IOException iox){
                System.out.println("Problem reading or writing");
                return false;
            }
            return true;
        }
        private boolean closeFiles(){
            boolean retVal=true;
            try{
                source.close();
            }
            catch(IOException iox){
                System.out.println("Problem closing"+destName);
                retVal=false;
            }
            return retVal;
        }
        public boolean copy(String src,String dst){
            sourceName=src;
            destName=dst;
            return openFiles()&&copyFiles()&&closeFiles();
        }
}
import java.util.*;
import java.io.*;
public class FileCopy {
    public static void main(String[] args) {
         new CopyMaker().copy("d:/a.txt","d:/b.txt");
        String filename="d:/a.txt";
            String line;
            try{
            BufferedReader in =new BufferedReader(new FileReader(filename));
            line =in.readLine();
            while (line!=null){
                StringTokenizer st=new StringTokenizer(line);
                while (st.hasMoreTokens())
                {
                  System.out.println(st.nextToken());
                }
                line=in.readLine();
            }
            in.close();
```

```
            }
            catch(IOException iox){
            System.out.println("problem ");
            }
    }
}
```

1. 设计一个Person类，包含：姓名，年龄，性别。要求：该类至多只能创建一男、一女两个对象。

2. 设计一个测试类Test，创建若干个Person类对象，测试是否符合要求。

```
public class Person {
    private String Name,Sex;
    private int age;
    static private int man=0,woman=0;
    private Person(String Name ,String Sex,int age){
        this.Name=Name;this.age=age;this.Sex=Sex;
    }
    public void Print()
    {
        System.out.println("Create Succeed!"+"The name is:"+Name+"\n");
    }
    public static Person CreatePerson(String Name ,String Sex,int age){
        Person a=null;
        if(Sex=="woman")
            if(woman==0){
                a= new Person(Name ,Sex,age);woman++;a.Print();}
            else System.out.println("woman has already been created");
        else if(Sex=="man")
            if(man==0){
                a= new Person(Name ,Sex,age);man++;a.Print();}
            else System.out.println("man has already been created");
        else System.out.println("error.");
        return a;
    }
}
public class test {
    public static void main(String[] args) {
        Person a=Person.CreatePerson("panyanyan","woman",19);
        Person b=Person.CreatePerson("liuzhao", "man", 19);
        Person c=Person.CreatePerson("lizongxia", "woman", 21);
    }
}
```

设计一个线程Thread类的子类DataThread, 使用DataThread构建两个线程，分别输出50以内的奇数和偶数，并使两个线程并发执行。

```
public class DataThread extends Thread{
```

```
    private int n;
    public DataThread(){}
    public DataThread(int n){this.n=n;}
    public void run(){
        for(int i=0;i+n<=50;i++,i++)
            System.out.print((i+n)+" ");
    }
}
public class Main {
    public static void main(String[] args) {
        DataThread t1=new DataThread(0);
        DataThread t2=new DataThread(1);
        t1.start();
        t2.start();
    }
}
```

1. 设计一个数据单元类DataUnit, 它包含学号(Number)和姓名(Name)两个数据成员。

2. 设计两个线程，一个线程往数据单元里写信息，一个线程从数据单元里读信息。要求使用线程同步技术，使得每写一次就往外读一次。例如，写和读的数据序列为：

Write: 1, Name1
Read: 1, Name1
Write: 2, Name2
Read: 2, Name2
…

```
import java.util.Scanner;
public class DataUnit {
    String Name,Number;
    boolean available=false;
    Scanner in=new Scanner(System.in);
    public synchronized void read(){
        if(available)
            try{
                wait();
            }
            catch(Exception e){
            }
        System.out.printf("请输入学号: ");
        try{
            Number=in.next();
        }
        catch(Exception e){
            System.out.println("输入学号出错!");
        }
        System.out.printf("请输入姓名: ");
```

```
        try{
            Name=in.next();
        }
        catch(Exception e){
           System.out.println("输入姓名出错!");
        }
        System.out.println();
        available=true;
        notify();
    }
    public synchronized void write(){
        if(!available)
            try{
                wait();
            }
            catch(Exception e){ }
        System.out.println("输出学生学号："+Number+"姓名"+Name+"\n");
        available=false;
        notify();
    }
}
import java.util.Scanner;
public class Read extends Thread{
DataUnit d1=null;
public Read(DataUnit d){
    this.d1=d;
}
public void run(){
    while(true){
        d1.read();
    }
}
}
import java.util.Scanner;
public class Write extends Thread{
DataUnit d2=null;
public Write(DataUnit d){
    this.d2=d;
}
public void run(){
    while(true){
        d2.write();
    }
}
```

```
}
public class Main {
    public static void main(String[] args) {
        DataUnit data=new DataUnit();
        new Read(data).start();
        new Write(data).start();
    }

}
```