java 经典程序 100 例

1，编写程序，判断给定的某个年份是否是闰年。

闰年的判断规则如下：

（1）若某个年份能被 4 整除但不能被 100 整除，则是闰年。

（2）若某个年份能被 400 整除，则也是闰年。

```
import java.util.Scanner;
class Bissextile{
    public static void main(String[] arge){
        System.out.print("请输入年份");
    int year;       //定义输入的年份名字为“year”
    Scanner scanner = new Scanner(System.in);
    year = scanner.nextInt();
    if (year<0||year>3000){
        System.out.println("年份有误，程序退出！ ");
        System.exit(0);
        }
    if ((year%4==0)&&(year%100!=0)||(year%400==0))
        System.out.println(year+" is bissextile");
    else
        System.out.println(year+" is not bissextile ");
    }
}
```

2，给定一个百分制的分数，输出相应的等级。

90 分以上	A 级
80~89	B 级
70~79	C 级
60~69	D 级
60 分以下	E 级

```
import java.util.Scanner;
class Mark{
    public static void main(String[] args){
        System.out.println("请输入一个分数");
        //定义输入的分数为“mark”，且分数会有小数
        double mark;
        Scanner scanner = new Scanner(System.in);
        mark = scanner.nextDouble();

        //判断是否有输入错误。
```

```
            if(mark<0||mark>100){
                System.out.println("输入有误！  ");
                System.exit(0);
            }
            /*判断分数的等级
            90 分以上者 A 级， 80~89 分者 B 级，70~79 分者 C 级， 60~69 者 D 级，60 分以下 E 级 */
            if (mark>=90) System.out.println("this mark is grade \'A\' ");
            else if (mark>=80) System.out.println("this mark is grade \'B\' ");
            else if (mark>=70) System.out.println("this mark is grade \'C\' ");
            else if (mark>=60) System.out.println("this mark is grade \'D\' ");
            else    System.out.println("this mark is grade \'E\' ");
        }
}
```

3，编写程序求 1+3+5+7+……+99 的和值。

```
class he{
        public static void main(String[] args){
            int number = 1;   //初始值 1，以后再+2 递增上去
            int sum = 0;
            for ( ; number <100; number+=2 ){ sum += number; }
            System.out.println("1+3+5+7+……+99= " +sum);
        }
}
```

4、利用 for 循环打印 9*9 表?

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12   4*4=16
1*5=5   2*5=10   3*5=15   4*5=20   5*5=25
1*6=6   2*6=12   3*6=18   4*6=24   5*6=30   6*6=36
1*7=7   2*7=14   3*7=21   4*7=28   5*7=35   6*7=42   7*7=49
1*8=8   2*8=16   3*8=24   4*8=32   5*8=40   6*8=48   7*8=56   8*8=64
1*9=9   2*9=18   3*9=27   4*9=36   5*9=45   6*9=54   7*9=63   8*9=72   9*9=81
```

```
//循环嵌套，打印九九乘法表
public class NineNine{
        public static void main(String[]args){
```

```
        System.out.println();
        for (int j=1;j<10;j++){
            for(int k=1;k<10;k++) {     //老师的做法，判断语句里的 k<=j，省去下列的 if 语句。
                if (k>j) break;             //此处用 continue 也可以，只是效率低一点
                System.out.print(" "+k+"X"+j+"="+j*k);
             }
            System.out.println();
            }
        }
}
```

6、输出所有的水仙花数，把谓水仙花数是指一个数 3 位数，其各各位数字立方和等于其本身，

例如： 153 = 1*1*1 + 3*3*3 + 5*5*5

```
class DafodilNumber{
        public static void main(String[] args){
            System.out.println("以下是所有的水仙花数");
        int number = 100;         //  由于水仙花数是三位数，故由 100 开始算起

        int i, j, k;         // i    j    k    分别为 number 的百位、十位、个位
        for (int sum; number<1000; number++){
            i=number/100;    j=(number-i*100)/10;    k=number-i*100-j*10;
            sum=i*i*i+j*j*j+k*k*k;
            if (sum==number) System.out.println(number+" is a dafodil number! ");
            }
        }
}
```

7、求 a+aa+aaa+.....+aaaaaaaaa=?

其中 a 为 1 至 9 之中的一个数，项数也要可以指定。

```
import java.util.Scanner;
class Multinomial{
        public static void main(String[] args){
            int    a;          //定义输入的 a
            int    howMany;     //定义最后的一项有多少个数字
            Scanner scanner = new Scanner(System.in);
                System.out.println("请输入一个 1~9 的 a 值");
```

```
            a = scanner.nextInt();
                System.out.println("请问要相加多少项？");
            howMany = scanner.nextInt();
            int sum=0;
            int a1=a;   // 用来保存 a 的初始值
            for (int i=1; i<=howMany; i++){
                sum+= a;
                a = 10*a +a1;     // 这表示 a 的下一项
            // 每次 a 的下一项都等于前一项*10，再加上刚输入时的 a ；注意，这时的 a 已经变化了。
                }
            System.out.println("sum="+sum);
        }
}
```

8、求 2/1+3/2+5/3+8/5+13/8.....前 20 项之和？

```
class Sum{
      public static void main(Sting[] args){
            double sum=0;
            double fenZi=2.0, fenMu=1.0;       //初始的分子 (fenZi)＝2，分母(fenMu)＝1
            for(int i=1; i<=20; i++){
                sum += fenZi / fenMu ;
                fenMu = fenZi;                  //下一项的分母 ＝ 上一项的分子
                fenZi += fenMu;                //下一项的分子 ＝ 上一项的分子加分母
            }
            System.out.println("sum= "sum);
      }
}
```

9、利用程序输出如下图形:

```
    *
    * * *
    * * * * *
    * * * * * * *
    * * * * *
    * * *
    *
```

```
class Asterisk{
```

```
    public static void main(String[] args){
        for (int i=1; i<=13; i+=2){
            for(int j=1; j<=i && i+j<= 14; j++){System.out.print("* ");}
            System.out.println();   // 换行
        }
    }
}
```

11、计算圆周率

PI＝4－4/3+4/5-4/7.....

打印出第一个大于 3.1415 小于 3.1416 的值

```
class Pi {
    public static void main(String[] args){
        double pi =0;   //定义初始值
        double fenZi = 4;       //分子为 4
        double fenMu = 1;   //第一个 4，可看作分母为 1 的分式，以后的分母每次递增 2
        for (int i = 0; i < 1000000000; i++){ //运行老久，减少循环次数会快很多，只是精确
度小些
            pi += (fenZi/fenMu) ;
            fenZi *= -1.0;       //每项分子的变化是+4，－4，+4，－4 ....
            fenMu += 2.0;       //分母的变化是 1，3，5，7，  ....    每项递加 2
            }
        System.out.println(pi);
    }
}
```

输出结果为 pi = 3.1415926525880504，应该不精确

12、输入一个数据 n，计算斐波那契数列(Fibonacci)的第 n 个值

1 1 2 3 5 8 13 21 34

规律：一个数等于前两个数之和

```
//计算斐波那契数列(Fibonacci)的第 n 个值
public class Fibonacci{
    public static void main(String args[]){
        int n = Integer.parseInt(args[0]);
        int n1 = 1;//第一个数
        int n2 = 1;//第二个数
        int sum = 0;//和
        if(n<=0){
            System.out.println("参数错误!");
            return;
```

```
            }
            if(n<=2){
                sum = 1;
            }else{
                for(int i=3;i<=n;i++){
                    sum = n1+n2;
                    n1 = n2;
                    n2 = sum;
                }
            }
            System.out.println(sum);
        }
}


//计算斐波那契数列(Fibonacci)的第 n 个值
//并把整个数列打印出来
public class FibonacciPrint{
        public static void main(String args[]){
            int n = Integer.parseInt(args[0]);
            FibonacciPrint t = new FibonacciPrint();
            for(int i=1;i<=n;i++){
                t.print(i);
            }
        }
        public void print(int n){
            int n1 = 1;//第一个数
            int n2 = 1;//第二个数
            int sum = 0;//和
            if(n<=0){
                System.out.println("参数错误!");
                return;
            }
            if(n<=2){
                sum = 1;
            }else{
                for(int i=3;i<=n;i++){
                    sum = n1+n2;
                    n1 = n2;
                    n2 = sum;
                }
            }
            System.out.println(sum);
        }
```

}

13、求 1-1/3+1/5-1/7+1/9.....的值。
a,求出前 50 项和值。
b,求出最后一项绝对值小于 1e-5 的和值。

15、在屏幕上打印出 n 行的金字塔图案，如，若 n=5,则图案如下：

```
    *
   ***
  *****
 *******
*********
```

```
//打印金字塔图案
public class PrintStar{
    public static void main(String args[]){
        int col = Integer.parseInt(args[0]);
        for(int i=1;i<=col;i++){//i 表示行数
            //打印空格
            for(int k=0;k<col-i;k++){
                System.out.print(" ");
            }
            //打印星星
            for(int m=0;m<2*i-1;m++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

16、歌德巴赫猜想,任何一个大于六的偶数可以拆分成两个质数的和
打印出所有的可能

```
//任何一个大于六的偶数可以拆分成两个质数的和
//打印出所有的可能
public class Gedebahe{
    public static void main(String args[]){
        int num = Integer.parseInt(args[0]);
        if(num<=6){
            System.out.println("参数错误!");
            return;
```

```
        }
        if(num%2!=0){
            System.out.println("参数错误!");
            return;
        }
        Gedebahe g = new Gedebahe();
        //1 不是质数,2 是偶数,因此从 3 开始循环
        for(int i=3;i<=num/2;i++){
            if(i%2==0){//如果为偶数,退出本次循环
                continue;
            }
            //当 i 与 num-i 都为质数时,满足条件,打印
            if(g.isPrime(i) && g.isPrime(num-i)){
                System.out.println(i+" + "+(num-i)+" = "+num);
            }
        }
    }
```

第 4 章 数组

1. 定义一个 int 型的一维数组，包含 10 个元素，分别赋一些随机整数，然后求出所有元素的最大值，
最小值，平均值，和值，并输出出来。

```
class ArrayNumber{
    public static void main(String[] args){
        int[] arrayNumber;
        arrayNumber = new int[10];
    System.out.println("以下是随机的 10 个整数：");
        // 填入随机的 10 个整数
        for (int i =0; i<arrayNumber.length; i++){
            arrayNumber[i] = (int)(100*Math.random());
            System.out.print(arrayNumber[i]+" ");
```

```
            }
        System.out.println();
        int max = arrayNumber[0];
        int min = arrayNumber[0];
        int sum = 0;
        for (int i =0; i<arrayNumber.length; i++){
            if(max < arrayNumber[i])
                max = arrayNumber[i];    //求最大值
            if(min > arrayNumber[i])
                min = arrayNumber[i];     //求最小值
            sum += arrayNumber[i];
            }
    System.out.println("其中  Max="+max+",Min="+min+",Sum="+sum+",Avg="+sum/10.0);
    }
}
```

2.定义一个 int 型的一维数组，包含 10 个元素，分别赋值为 1~10， 然后将数组中的元素都向前移一个位置，
即，a[0]=a[1],a[1]=a[2],…最后一个元素的值是原来第一个元素的值，然后输出这个数组。

3. 定义一个 int 型的一维数组，包含 40 个元素，用来存储每个学员的成绩，循环产生 40 个 0~100 之间的随机整数，
将它们存储到一维数组中，然后统计成绩低于平均分的学员的人数，并输出出来。

4. （选做）承上题，将这 40 个成绩按照从高到低的顺序输出出来。

5,（选做）编写程序，将一个数组中的元素倒排过来。例如原数组为 1，2，3，4，5；则倒排后数组中的值
为 5，4，3，2，1。

6,要求定义一个 int 型数组 a,包含 100 个元素,保存 100 个随机的 4 位数。再定义一个 int 型数组 b，包含 10 个元素。统计 a 数组中的元素对 10 求余等于 0 的个数，保存到 b[0]中；对 10 求余等于 1 的个数，保存到 b[1]中，……依此类推。

```
class Remain{
    public    static void main( String[] args){
        int[] a = new int[100];

        //保存 100 个随机 4 位数到  a  中
        for (int i = 0;    i < a.length;    i++){
            a[i] = (int) (1000*Math.random());
        }
```

```
        //统计 a 数组中的元素对 10 求余的各个的数目
        int[] b = new int[10];
        int k,sum;
        for (int j = 0;  j < b.length;  j++){
            for (k=0,sum=0;  k < a.length;  k++){
                if ((a[k]%10)==j) sum++;
            }
            b[j] = sum;
            System.out.printf("b[%d]=%d\n",j,b[j]);
        }
    }
}
```

7,定义一个 20*5 的二维数组，用来存储某班级 20 位学员的 5 门课的成绩；这 5 门课按存储顺序依次为：core C++，coreJava，Servlet，JSP 和 EJB。

（1）循环给二维数组的每一个元素赋 0~100 之间的随机整数。

（2）按照列表的方式输出这些学员的每门课程的成绩。

（3）要求编写程序求每个学员的总分，将其保留在另外一个一维数组中。

（4）要求编写程序求所有学员的某门课程的平均分。

```
class Student{
    public static void main(String[] args ){
        int[][] mark = new int[20][5];
        // 给学生赋分数值，随机生成
        for ( int i = 0;  )
    }
}//未完成
```

8,完成九宫格程序

在井字形的格局中(只能是奇数格局)，放入数字(数字由)，使每行每列以及斜角线的和都相等

经验规则：从 1 开始按顺序逐个填写； 1 放在第一行的中间位置；下一个数往右上角 45 度处填写；

如果单边越界则按头尾相接地填；如果有填写冲突，则填到刚才位置的底下一格；

如果有两边越界，则填到刚才位置的底下一格。

个人认为，可以先把最中间的数填到九宫格的最中间位置；再按上面的规则逐个填写，而且

填的时候还可以把头尾对应的数填到对应的格子中。(第 n 个值跟倒数第 n 个值对应，格局上以最中

间格为轴心对应)

这样就可以同时填两个数，效率比之前更高；其正确性有待数学论证(但多次实验之后都没发现有错)。

九宫格的 1 至少还可以填在另外的三个位置，只是接下来的填写顺序需要相应改变；

再根据九宫格的对称性，至少可以有 8 种不同的填写方式

```
import java.util.Scanner;
class NinePalace{
    public static void main(String[] args){
        // 定义 N 为九宫格的行列数，需要输入
        System.out.println("请输入九宫格的行列规模(只能是奇数的)");
        Scanner n = new Scanner(System.in);
        int N;

        //判断格局是否奇数 （可判断出偶数、负数 及小数）
        double d;
        while (true){
            d = n.nextDouble();
            N = (int)d;
            if ((d-N)>1.0E-4||N%2==0||N<0)
                {System.out.println("输入出错,格局只能是正奇数。请重新输入");}
            else break;
        }

        //老师的九宫格填写方法
        int[][] result = new int[N][N];     //定义保存九宫格的数组
        int row = 0; //行 初始位置
        int col = N/2; //列 初始位置,因为列由 0 开始，故 N/2 是中间位置
        for (int i=1;   i<=N*N; i++){
            result [row][col] = i;
            row--;
            col++;
            if (row<0&&col>=N){col--;row+=2;} //行列都越界
            else if (row<0){ row = N-1;}      //行越界
            else if (col>=N){col = 0;}    //列越界
            else if (result[row][col] != 0){col--;row+=2;}    //有冲突
        }

        //打印出九宫格
        for (int i=0;   i<N;    i++){
            for(int j=0;    j<N; j++){System.out.print(result[i][j]+"\t");}
            System.out.println();
        }

        //我个人的填格方式
```

```
        int[][] result2 = new int[N][N];   //为免冲突，重新 new 一个数组
        result2[N/2][N/2] = (N*N+1)/2;   //先把中间值赋予中间位置
        row = 0;    //定义行及列的初始赋值位置。之前赋值的 for 对两个值有影响，故需重新定位
        col = N/2;
        for (int i=1; i<=N*N/2; i++){
            result2[row][col] = i;
            //下面这句是把跟 i 对应的值放到格局对应的位置上
            result2[N-row-1][N-col-1] = N*N+1-i;
            row--;
            col++;
            if (row<0){ row = N-1;}     //行越界
            else if (col>=N){col = 0;}   //列越界
            else if (result2[row][col] != 0){col--;row+=2;}   //有冲突
            //这方法不可能出现行列两边都越界的情况,详情需要数学论证
        }

        System.out.println();
        //再次打印出九宫格，以对比验证
        for (int i=0;   i<N;   i++){
            for(int j=0;   j<N; j++){System.out.print(result2[i][j]+"\t");}
            System.out.println();
        }

    }
}
```

9,求一个 3*3 矩阵对角线元素之和

10,打印杨辉三角

11. 约梭芬杀人法

把犯人围成一圈，每次从固定位置开始算起，杀掉第 7 个人，直到剩下最后一个。

11_2、用数组实现约瑟夫出圈问题。 n 个人排成一圈，从第一个人开始报数，从 1 开始报，报到 m 的人出圈，剩下的人继续开始从 1 报数，直到所有的人都出圈为止。对于给定的 n,m，求出所有人的出圈顺序。

12. 判断随机整数是否是素数

产生 100 个 0-999 之间的随机整数，然后判断这 100 个随机整数哪些是素数，哪些不是？

```
public class PrimeTest{
    public static void main(String args[]){
        for(int i=0;i<100;i++){
            int num = (int)(Math.random()*1000);
            PrimeTest t = new PrimeTest();
            if(t.isPrime(num)){
                System.out.println(num+"  是素数!");
            }else{
                System.out.println(num+"  不是素数!");
            }
            System.out.println();
        }
    }
    public boolean isPrime(int num){
        for(int i=2;i<=num/2;i++){
            if(num%i==0){
                System.out.println(num+"第一个被"+i+"整除!");
                return false;
            }
        }
        return true;
    }
}
```

冒泡排序法：

```
//按从大到小的排序
int tmp = a[0];
for (int i=0; i < a.length; i++){
    for (int j=0; j < a.length - i -1; j++){
        if (a[j] < a[j+1]) {
            tmp = a[j];
            a[j] = a[j+1];
            a[j+1] = tmp;
```

```
        }
    }
}
```

day06 练习

某公司的雇员分为以下若干类：

Employee：这是所有员工总的父类，属性：员工的姓名和生日月份。

方法：getSalary(int month) 根据参数月份来确定工资，如果该月员工过生日，则公司会额外奖励 100 元。

SalariedEmployee：Employee 的子类，拿固定工资的员工。属性：月薪

HourlyEmployee：Employee 的子类，按小时拿工资的员工，每月工作超出 160 小时的部分按照 1.5 倍工资发放

属性：每小时的工资、每月工作的小时数

SalesEmployee：Employee 的子类，销售人员，工资由月销售额和提成率决定

属性：月销售额、提成率

BasePlusSalesEmployee：SalesEmployee 的子类，有固定底薪的销售人员，工资由底薪加上销售提成部分　　属性：底薪。

```
public class TestEmployee{
    public static void main(String[]args){
        Employee[] es = new Employee[5];
        es[0] = new Employee("赵君",2);
        es[1] = new SalariedEmployee("宋婕", 1, 8000);
        es[2] = new HourlyEmployee("王超", 5, 10, 300);
        es[3] = new SalesEmployee("秋娥", 2, 200000, 0.05);
        es[4] = new BaseSalarySalesEmployee("郭镫鸿", 1, 1000000, 0.1, 10000);
        int month = 2;//本月为 2 月
        System.out.println("宇宙集团"+month+"月工资表： ");
        for(int i=0; i<es.length; i++){
            System.out.println(es[i].getName()+":"+es[i].getSalary(month));
        }
    }
}

class Employee{
    private String name;
    private int birth;
    public String getName(){
        return name;
    }
    public Employee(String name, int birth){
```

```
            this.name = name;
            this.birth = birth;
        }
        public double getSalary(int month){
            if(month==birth){
                return 100;
            }
            return 0;
        }
}

class SalariedEmployee extends Employee{
        private double salary;
        public SalariedEmployee(String name, int birth, double salary){
            super(name, birth);
            this.salary = salary;
        }
        public double getSalary(int month){
            return salary + super.getSalary(month);
        }
}

class HourlyEmployee extends Employee{
        private double hourSalary;
        private int hour;
        public HourlyEmployee(String name, int birth, double hourSalary, int hour){
            super(name, birth);
            this.hourSalary = hourSalary;
            this.hour = hour;
        }
        public double getSalary(int month){
            if(hour<=160){
                return hourSalary*hour+super.getSalary(month);
            }else{
                return 160*hourSalary+(hour-160)*hourSalary*1.5+super.getSalary(month);
            }
        }
}

class SalesEmployee extends Employee{
        private double sales;
        private double pre;
        public SalesEmployee(String name, int birth, double sales, double pre){
            super(name, birth);
```

```
            this.sales = sales;
            this.pre = pre;
        }
        public double getSalary(int month){
            return sales*pre+super.getSalary(month);
        }
}

class BaseSalarySalesEmployee extends SalesEmployee{
        private double baseSalary;
        public BaseSalarySalesEmployee(String name, int birth, double sales, double pre, double
baseSalary){
            super(name, birth, sales, pre);
            this.baseSalary = baseSalary;
        }
        public double getSalary(int month){
            return baseSalary+super.getSalary(month);
        }
}



/**
 * 在原有的雇员练习上修改代码
 * 公司会给 SalaryEmployee 每月另外发放 2000 元加班费,给
 * BasePlusSalesEmployee 发放 1000 元加班费
 * 改写原有代码,加入以上的逻辑
 * 并写一个方法,打印出本月公司总共发放了多少加班费
 * @author Administrator
 *
 */
public class EmployeeTest {

        /**
         * @param args
         */
        public static void main(String[] args) {
            Employee e[] = new Employee[4];
            e[0] = new SalariedEmployee("魏威",10,5000);
            e[1] = new HourlyEmployee("段利峰",8,80,242);
            e[2] = new SalesEmployee("林龙",11,300000,0.1);
            e[3] = new BasedPlusSalesEmployee("华溪",1,100000,0.15,1500);
            for(int i=0;i<e.length;i++){
                System.out.println(e[i].getName()+": "+e[i].getSalary(11));
```

```
            }

            //统计加班费
            int result = 0;
//              for(int i=0;i<e.length;i++){
//                  if(e[i] instanceof SalariedEmployee){
//                      SalariedEmployee s = (SalariedEmployee)e[i];
//                      result += s.getAddtionalSalary();
//                  }
//                  if(e[i] instanceof BasedPlusSalesEmployee){
//                      BasedPlusSalesEmployee b = (BasedPlusSalesEmployee)e[i];
//                      result += b.getAddtionalSalary();
//                  }
//              }

            for(int i=0;i<e.length;i++){
                result += e[i].getAddtionalSalary();
            }
            System.out.println("加班费: "+result);
        }
}

interface AddtionalSalary{
        int getAddtionalSalary();
}

class Employee implements AddtionalSalary{
        private String name;//员工姓名
        private int birth;//员工生日月份
        public Employee(String name,int birth){
            this.name = name;
            this.birth = birth;
        }
        public int getSalary(int month){
            int result = 0;
            if(month==birth)
                result = 100;
            return result;
        }
        public String getName(){
            return name;
        }

        public int getAddtionalSalary(){
```

```
            return 0;
        }
}

class SalariedEmployee extends Employee{
        private int salaryPerMonth;
        public SalariedEmployee(String name,int birth,int salaryPerMonth){
            super(name,birth);
            this.salaryPerMonth = salaryPerMonth;
        }
        public int getSalary(int month){
            return this.salaryPerMonth + super.getSalary(month)+
                this.getAddtionalSalary();
        }
        public int getAddtionalSalary(){
            return 2000;
        }
}

class HourlyEmployee extends Employee{
        private int salaryPerHour;
        private int hoursPerMonth;
        public HourlyEmployee(String name,int birth,int salaryPerHour,int hoursPerMonth){
            super(name,birth);
            this.salaryPerHour = salaryPerHour;
            this.hoursPerMonth = hoursPerMonth;
        }
        public int getSalary(int month){
            int result = 0;
            if(this.hoursPerMonth<=160){
                result = hoursPerMonth*salaryPerHour;
            }else{
                result = 160*salaryPerHour +
                (int)((hoursPerMonth-160)*1.5*salaryPerHour);
            }
            return result+super.getSalary(month);
        }
}

class SalesEmployee extends Employee{
        private int sales;
        private double rate;
        public SalesEmployee(String name,int birth,int sales,double rate){
            super(name,birth);
```

```
            this.sales = sales;
            this.rate = rate;
        }
        public int getSalary(int month){
            return (int)(sales*rate)+super.getSalary(month);
        }
}

class BasedPlusSalesEmployee extends SalesEmployee{
        private int basedSalary;
        public BasedPlusSalesEmployee(String name,int birth,int sales,double rate,int basedSalary){
            super(name,birth,sales,rate);
            this.basedSalary = basedSalary;
        }
        public int getSalary(int month){
            return this.basedSalary+super.getSalary(month) +
            this.getAddtionalSalary();
        }
        public int getAddtionalSalary(){
            return 1000;
        }
}
```

经典算法：

1. 某学校为学生分配宿舍，每 6 个人一间房（不考虑性别差异），问需要多少房？

答案： (x+5)/6

注意理解 int 类型数值。

2. 让数值在 0～9 之间循环。

```
public class test{
        public static void main(String[] args){
            int i=0;
            while(true){
                i = (i+1)%10;
                System.out.println(i);
            }
        }
}
```

作业：

1. 写一个数组类（放对象）：

功能包括：添加(添加不限制多少项)、修改、插入、删除、查询

```
class MyArray{
        private Object[] os = new Object[10];
        public void add(Object o);
        public void set(int index, Object o);
        public void insert(int index, Objecto);
        public void remove(int index);
        public void remove(Object o);
        public Object get(int index);

}

public class TestMyArray{
    public static void main(String[]args){
        MyArray ma = new MyArray();
        ma.add("aaa");
        ma.add("bbb");
        ma.add("ccc");
        Object o = ma.get(1);
        Iterator it = ma.iterator();
        while(it.hasNext()){
            Object o1 = it.next();
            System.out.println(o1);
        }
    }
}
```

作业 10-08

1. 随机产生 20 个整数(10 以内的)，放入一个 ArrayList 中， 用迭代器遍历这个 ArrayList
2. 并删除其中为 5 的数
3. 再产生 3 个整数，插入到位置 4 处
4. 把所有值为 1 的数都变成 10

```
import java.util.ArrayList;
class ArrayList{
        private Object[] os = new Object[20];

}

public class TestArray{
    public static void main(String[]args){
```

```
            ArrayList a = new ArrayList();

            ma.add("aaa");
            ma.add("bbb");
            ma.add("ccc");
            Object o = ma.get(1);
            Iterator it = ma.iterator();
            while(it.hasNext()){
                Object o1 = it.next();
                System.out.println(o1);
            }
        }
    }
```

1. 产生 3000 个 10 以内的数，放入 hashSet
2. 遍历它，打印每一个值

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
public class TestHashSet {
    public static void main(String[] args) {
        Random r = new Random();
        HashSet hs1 = new HashSet();
        for(int i=0; i<3000; i++){
            hs1.add(r.nextInt(10));
        }
        Iterator it1 = hs1.iterator();
        while(it1.hasNext()){
            System.out.print(it1.next()+" ");
        }
    }
}
```

//由于 HashSet 不能重复，所以只有 10 个数在里面，按哈希排序

2 4 9 8 6 1 3 7 5 0

```
/*
 * 测试 TreeSet 的比较器，
```

```
 * 在有自己的比较器的情况下，如何实现 Comparable 接口
 */

import java.util.*;
class Teacher{
	int id;
	String name;
	int age;
	public Teacher() {}
	public Teacher(int id, String name, int age) {
		this.id = id;
		this.name = name;
		this.age = age;
	}
	public int getId() {	return id;	}
	public void setId(int id) {this.id = id;	}
	public String getName() {	return name;}
	public void setName(String name) {	this.name = name;}
	public int getAge() {return age;}
	public void setAge(int age) {this.age = age;}

	public int TeacherComparator(Object o){
		Teacher t1 = (Teacher) o;
		if(t1.getId() > id){return 1;}
		else if (t1.getId() < id){return -1;}
		return 0;
	}
}
class TreeSet{

}

class Test {
	public static void main(String[] args) {
		String s1 = new String("aaa");
		String s2 = new String("bbb");
		String s3 = new String("aaa");
		System.out.println(s1==s3);
		System.out.println(s1.equals(s3));

		HashSet hs = new HashSet();
		hs.add(s1);
		hs.add(s2);
		hs.add(s3);
```

```
            Iterator it = hs.iterator();
            while(it.hasNext()){
                System.out.println(it.next());
            }
            System.out.printf("%x\n",s1.hashCode());
            System.out.printf("%x\n",s2.hashCode());
            System.out.printf("%x\n",s3.hashCode());
        }
}
```

1. 在 Map 中，以 name 作 Key，以 Student 类 作 Velue，写一个 HashMap

```
import java.util.*;
class Student{
        int id;
        String name;
        int age;
        public Student() {}
        public Student( int id, String name, int age) {
            this.id = id;
            this.name = name;
            this.age = age;
        }
        public int getId() {return id;}
        public void setId(int id) {this.id = id;}
        public String getName() {return name;}
        public void setName(String name) {this.name = name;}
        public int getAge() {return age;}
        public void setAge(int age) {this.age = age;}
}

class TestHashMap{
        public static void main(String[] args) {
            HashMap hm = new HashMap();
            Student s1 = new Student(1,"jacky",19);
            hm.put("jacky",s1);
            hm.put("tom",new Student(2,"tom",21));
            hm.put("kitty",new Student(3,"kitty",17));

            Iterator it = hm.keySet().iterator();
            while(it.hasNext()){
                Object key =    it.next();
                Student value = (Student) hm.get(key);
```

```
            System.out.println(key+":id="+value.id+",age="+value.age);
        }
        System.out.println("============================");

        //比较 KeySet() 和 entrySet() 两种迭代方式
        for(Iterator i1 = hm.entrySet().iterator(); i1.hasNext(); )
        { Map.Entry me = (Map.Entry) i1.next();
       Student s = (Student) me.getValue();
            System.out.println(me.getKey()+": id="+s.id+" age="+s.age);
        }
    }
}
```

day13 homework

1.

```
/*********************************************************************************
****
```

自己写一个栈:　　(先进后出)

　　建议底层用 LinkedList 实现

参照 java.util.Stack

方法:　boolean empty()　测试堆栈是否为空。

　　E　peek()　　查看栈顶对象而不移除它。

　　E　pop()　　移除栈顶对象并作为此函数的值返回该对象。

　　E　push(E item)　把项压入栈顶。

　　int　　search(Object o)　　返回对象在栈中的位置，以 1 为基数。

```
*********************************************************************************
********/
//不能用继承，因为它破坏封装。只需调用即可
import java.util.LinkedList;
class MyStack<E>{
    private LinkedList<E> list = new LinkedList<E>();
    public  boolean empty()      {return list.isEmpty();}
    public  E peek()              {return list.peek();    }
    public  E pop()               {return list.poll();    }
    public  void push(E o)       {list.addFirst(o);      }

    //int     indexOf(Object o) 返回此列表中首次出现的指定元素的索引，如果此列表中不包含该元素，则返回 -1。
    public  int search(Object o){return list.indexOf(o);}
}
```

2.

/**********

定义以下类，完成后面的问题，并验证。

Exam 类　考试类

属性：　若干学生　一张考卷

提示：学生采用 HashSet 存放

Paper 类　考卷类

属性：若干试题

提示：试题采用 HashMap 存放，key 为 String，表示题号，value 为试题对象

Student 类　　学生类

属性：姓名　一张答卷　一张考卷　考试成绩

Question 类　　试题类

属性：题号 题目描述　　若干选项　　正确答案

提示：若干选项用 ArrayList

AnswerSheet 类　　答卷类

属性：每道题的答案

提示：答卷中每道题的答案用 HashMap 存放，key 为 String，表示题号，value 为学生的答案

问题：为 Exam 类添加一个方法，用来为所有学生判卷，并打印成绩排名（名次、姓名、成绩）

**********/

3.

```
/*****************************************************************************
*********
项目：商品管理系统
功能：增删改查 （可按各种属性查）
商品属性：名称、价格（两位小数）、种类
*****************************************************************************
********/
```

day17 图形界面

1. 计算器

```
/*****************例题 画出计算器的界面*****************************
界面如下：
        1     2     3     +
        4     5     6     -
        7     8     9     *
        0     .     =     /
  ********************/
import java.awt.*;
import javax.swing.*;
```

```
class Calculator {
    public static void main(String[] args){
        JTextField text = new JTextField();
        JFrame f = new JFrame("计算器");
        Font font = new Font("宋体", Font.BOLD, 25);//"宋体"想写成默认，则写“null”
        text.setFont(font); //定义字体
        text.setHorizontalAlignment(JTextField.RIGHT);//令 text 的文字从右边起
        text.setEditable(false);//设置文本不可修改，默认可修改(true)
        f.add(text, BorderLayout.NORTH);//Frame 和 Dialog 的默认布局管理器是 Border Layout
        ButtonActionListener listener = new ButtonActionListener(text);//事件反应在 text 中
        JPanel buttonPanel = new JPanel();//设法把计算器键盘放到这个 Jpanel 按钮上
        String op = "123+456-789*0.=/";
        GridLayout gridlayout = new GridLayout(4,4,10,10);
        buttonPanel.setLayout(gridlayout);//把计算器键盘放到 buttonPanel 按钮上
        for(int i=0; i<op.length(); i++){
            char c = op.charAt(i); //拿到字符串的第 i 个字符
            JButton b = new JButton(c+"");//把字符放到按钮上
            b.addActionListener(listener);//在按钮上放置监听器，每次按都会有反应
            buttonPanel.add(b);//把按钮放到 buttonPanel 上
        }//这个循环很值得学习，很常用
        f.add(buttonPanel/*, BorderLayout.CENTER*/);   //默认添加到 CENTER 位置
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(300, 250);
        f.setVisible(true);//这句要放到最后，等事件完成后再显示
}}

//监听者
class ButtonActionListener implements ActionListener{
    private JTextField textField;
    public ButtonActionListener(JTextField textField) {
        this.textField = textField;
    }
    public void actionPerformed(ActionEvent e) {//必须覆盖它的 actionPerformed()

        textField.append("哈哈，放了几个字\n");
}}
/**********************未实现计算器的具体功能*******************************/
```

2. 扫雷游戏

3. 俄罗斯方块

day19 多线程

写两个线程，一个线程打印 1~52，另一个线程打印字母 A-Z。打印顺序为 12A34B56C……5152Z。要求用线程间的通信。

注：分别给两个对象构造一个对象 o，数字每打印两个或字母每打印一个就执行 o.wait()。

在 o.wait()之前不要忘了写 o.notify()。

```
class Test{
    public static void main(String[] args) {
        Printer p = new Printer();
        Thread t1 = new NumberPrinter(p);
        Thread t2 = new LetterPrinter(p);
        t1.start();
        t2.start();
    }
}
class Printer{
    private int index = 1;//设为 1，方便计算 3 的倍数
    //打印数字的构造方法，每打印两个数字，等待打印一个字母
    public synchronized void print(int i){
        while(index%3==0){try{wait();}catch(Exception e){}}
        System.out.print(" "+i);
        index++;
        notifyAll();
    }
    //打印字母，每打印一个字母，等待打印两个数字
    public synchronized void print(char c){
        while(index%3!=0){try{wait();}catch(Exception e){}}
        System.out.print(" "+c);
        index++;
        notifyAll();
    }
}
//打印数字的线程
class NumberPrinter extends Thread{
```

```
        private Printer p;
        public NumberPrinter(Printer p){this.p = p;}
        public void run(){
            for(int i = 1; i<=52; i++){
                p.print(i);
            }
        }
}
//打印字母的线程
class LetterPrinter extends Thread{
        private Printer p;
        public LetterPrinter(Printer p){this.p = p;}
        public void run(){
            for(char c='A'; c<='Z'; c++){
                p.print(c);
            }
        }
}
/*如果这题中，想保存需要打印的结果，可在 Printer 类里定义一个成员变量
String s = ""; //不写“”的话是 null，null 跟没有东西是不一样的，它会把 null 当成字符 =_=
然后在两个 print()方法里面，while 循环后分别加上 s = s + " "+i; 以及 s = s +" "+ c;*/
```