第五讲--数组

任务目标

1、数组的初始化

2、英文字符统计、数字统计

3、数组的排序（冒泡排序、选择排序、插入排序）

相关知识

1、数组与循环、排序、频率统计。

1、新建数组

```
import java.lang.Math;
import java.util.Scanner;

public class Test1
{
    public static void main(String[] args)
    {
        int[] array = new int[]{1,2,3,4,5};
        for(int i=0;i<array.length;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```

2、数组的初始化

1、通过随机数产生数组

```
import java.lang.Math;
import java.util.Scanner;

public class Test1
{
    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*51);
        }
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
```

```
    }
}
```

2、将字符串转化为字符型数组

```
import java.util.Arrays;
import java.util.Scanner;

public class Test2
{
    public static void main(String[] args)
    {
        String paragraph = "There are moments in life when you miss someone so
much that you just want to pick them from your dreams and hug them for real Dream
what you want to dream; go where you want to go;be what you want to be,because
you have only one life and one chance to do all the things you want to do May you
have enough happiness to make you sweet,enough trials to make you strong,enough
sorrow to keep you human, enough hope to make you happy Always put yourself in
others'shoes.If you feel that it hurts you,it probably hurts the other person,
too. The happiest of people don't necessarily have the best of everything;they
just make the most of everything that comes along their way.Happiness lies for
those who cry,those who hurt, those who have searched,and those who have
tried,for only they can appreciate the importance of people who have touched
their lives.Love begins with a smile,grows with a kiss and ends with a tear.The
brightest future will always be based on a forgotten past, you can't go on well
in lifeuntil you let go of your past failures and heartaches. When you were
born,you were crying and everyone around you was smiling.Live your life so that
when you die,you're the one who is smiling and everyone around you is crying.
Please send this message to those people who mean something to you,to those who
have touched your life in one way or another,to those who make you smile when you
really need it,to those that make you see the brighter side of things when you
are really down,to those who you want to let them know that you appreciate their
friendship.And if you don't, don't worry,nothing bad will happen to you,you will
just miss out on the opportunity to brighten someone's day with this message.";
        char[] array = paragraph.toCharArray();
        int[] table = new int[26];
        for(int i=0;i<array.length;i++)
        {
            if(array[i]>='a' && array[i]<='z')
            {
                table[array[i]-'a']++;
            }
            if(array[i]>='A' && array[i]<='Z')
            {
                table[array[i]-'A']++;
            }
        }
        for(int i=0;i<table.length;i++)
        {
            System.out.print((char)(i+97)+":");
            System.out.print(table[i]+"\t");
        }
    }
}
```

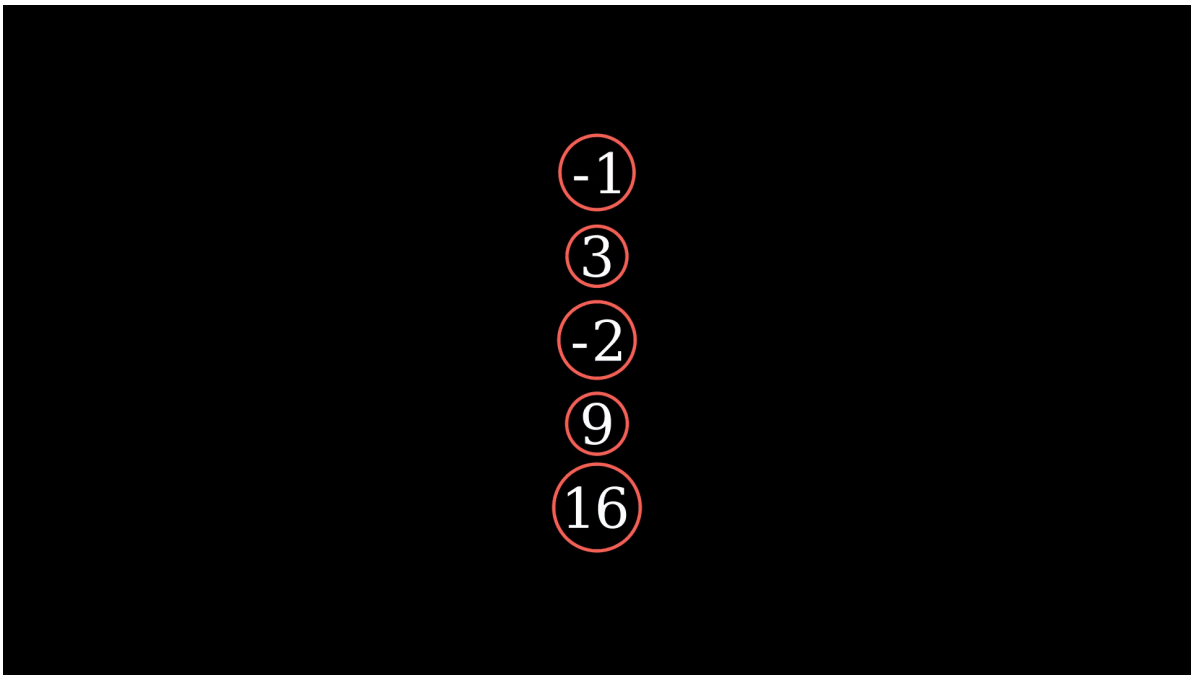
3、排序操作

1、冒泡排序。

- 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
- 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。这步做完后，最后的元素会是最大的数。
- 针对所有的元素重复以上的步骤，除了最后一个。
- 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

```
import java.lang.Math;
import java.util.Scanner;

public class Test3
{
    public static int[] bubble1(int[] a)
    {
        int temp;
        for(int i=0;i<a.length-1;i++)
        {
            for(int j=a.length-1;j>i;j--)
            {
                if(a[j]<a[j-1])
                {
                    temp=a[j];
                    a[j]=a[j-1];
                    a[j-1]=temp;
                }
            }
        }
        return a;
    }

     public static int[] bubble2(int[] a)
    {
        int temp;
         for(int i=a.length-1;i>0;i--)
          {
            for(int j=0;j<i;j++)
            {
                if(a[j]>a[j+1])
                {
                    temp=a[j+1];
                    a[j+1] =a[j];
                    a[j]=temp;
                }
            }
        }
        return a;
    }

    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*51);
```

```
        }
        array = bubble(array);
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```

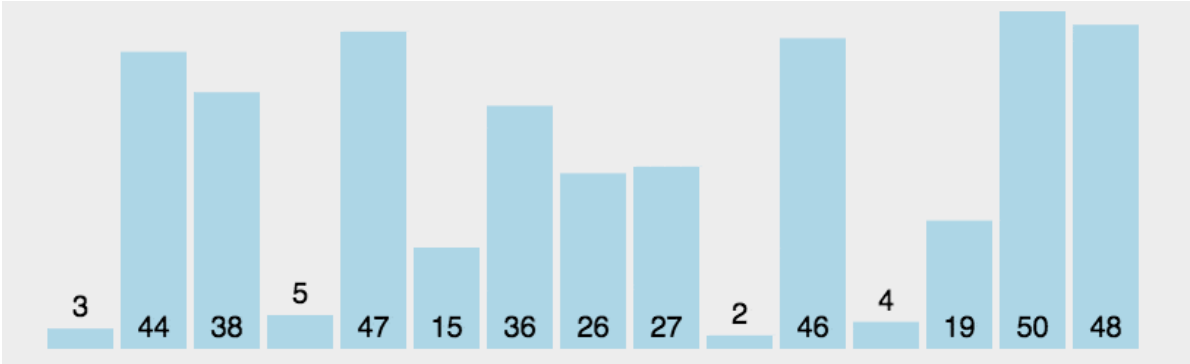


2、选择排序。

- 首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置。
- 再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。
- 重复第二步，直到所有元素均排序完毕。

```
import java.lang.Math;
import java.util.Scanner;

public class Test1
{
    public static int[] select(int[] a)
    {
        int k,t;
        for(int i=0;i<a.length-1;i++)
        {
            k = i;
            for(int j=i+1;j<a.length-1;j++)
            {
                if(a[j]<a[k])
                {
                    k=j;
                }
            }
            if(i!=k)
            {
                t = a[k];
                a[k] =a[i];
                a[i] =t;
```

```
            }
        }
        return a;
    }
    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*51);
        }
        array = select(array);
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```

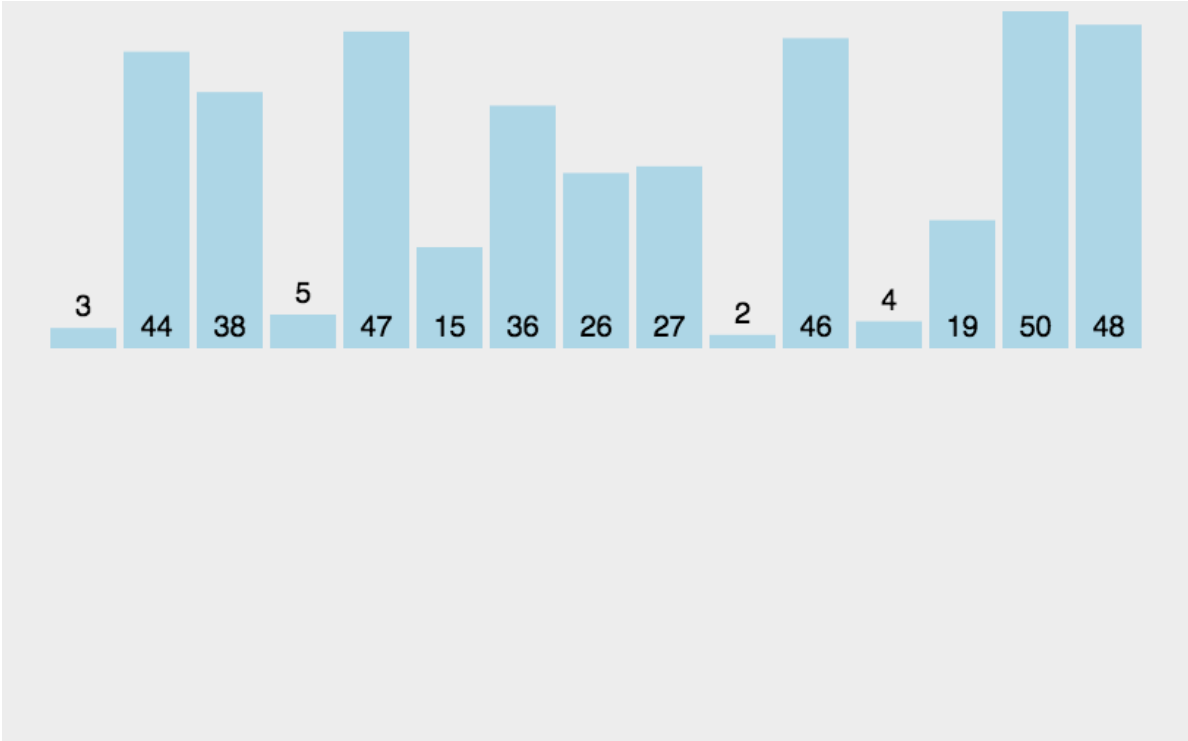


3、插入排序

- 将第一待排序序列第一个元素看做一个有序序列，把第二个元素到最后一个元素当成是未排序序列。
- 从头到尾依次扫描未排序序列，将扫描到的每个元素插入有序序列的适当位置。（如果待插入的元素与有序序列中的某个元素相等，则将待插入元素插入到相等元素的后面。）

```
import java.lang.Math;
import java.util.Scanner;

public class Test1
{
    public static int[] insert(int[] a)
    {
        int j,t;
        int i = 1;
        while(i<a.length)
        {
            j = i;
            while(j>0 && a[j]<a[j-1])
            {
                t = a[j];
                a[j]=a[j-1];
                a[j-1]=t;
                j--;
            }
```

```
            i++;
        }
        return a;
    }
    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*101-50);
        }
        array = insert(array);
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```

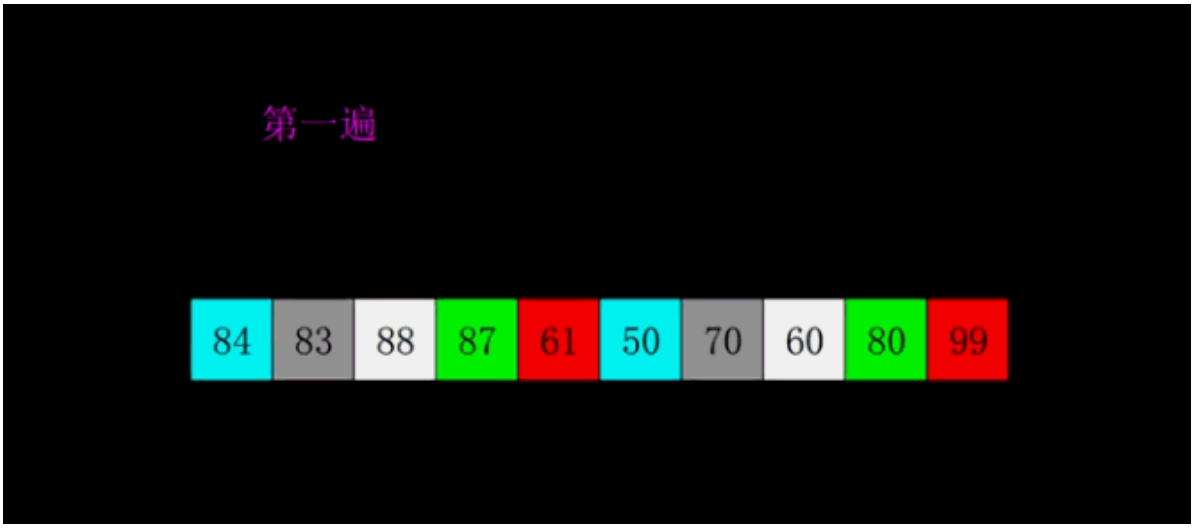


4、Shell 排序

- 选择一个增量序列 $t_1, t_2, \cdots, t_k$，其中 $t_i > t_j, t_k = 1$；
- 按增量序列个数 k，对序列进行k趟排序；
- 每趟排序，根据对应的增量 t_i，将待排序列分割成若干长度为m的子序列，分别对各子表进行直接插入排序。仅增量因子为 1 时，整个序列作为一个表来处理，表长度即为整个序列的长度。

```
import java.lang.Math;
import java.util.Scanner;
import java.util.Arrays;

public class Test1
{
    public static int[] shell(int[] a)
    {
        int length = a.length;
```

```
        int j, t, i, gap;
        j = t = 0;
        gap = 1;
        while(gap<length)
        {
            gap= gap*3+1;
        }
        while(gap>0)
            {
                i=gap;
                while(i<length)
                {
                    t = a[i];
                    j = i-gap;
                    while(j>=0 && a[j]>t)
                    {
                        a[j+gap] = a[j];
                        j-=gap;
                    }
                    a[j+gap]=t;
                    i++;
                }
                gap = Math.round(gap/3);
            }
        return a;
    }

    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*101-50);
        }
        array = shell(array);
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```

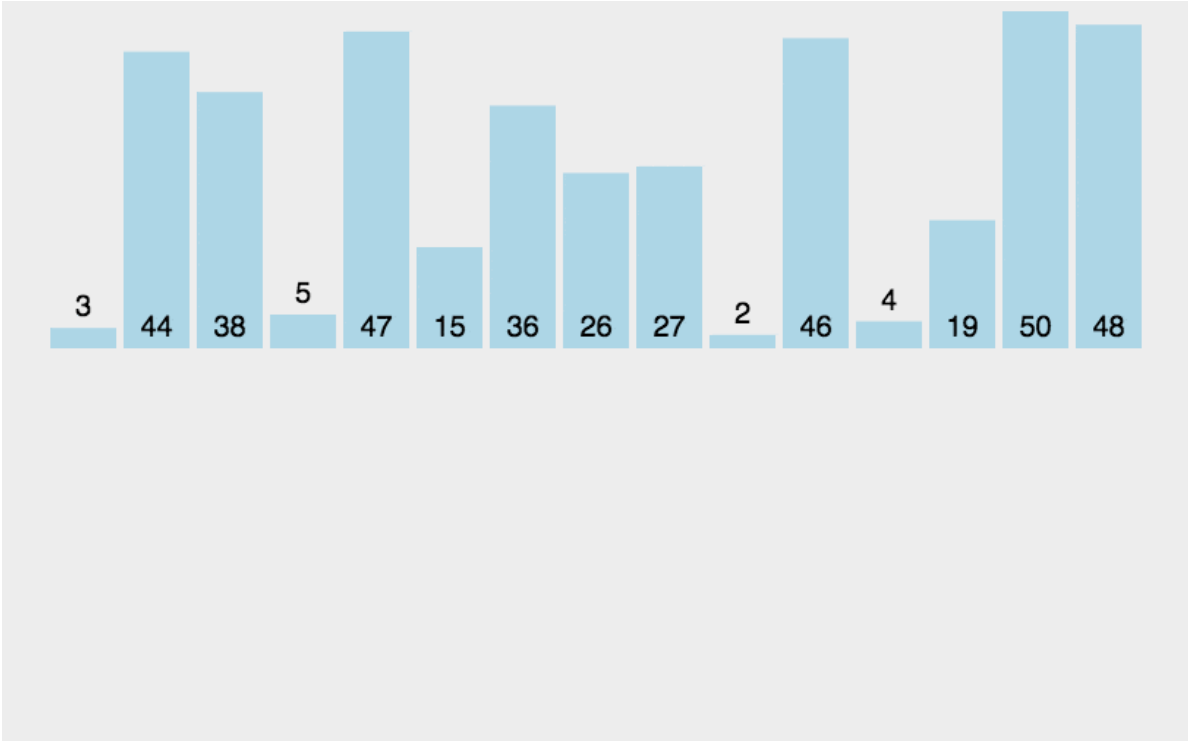


5 、归并排序

- 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列；
- 设定两个指针，最初位置分别为两个已经排序序列的起始位置；
- 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置；
- 重复步骤 3 直到某一指针达到序列尾；
- 将另一序列剩下的所有元素直接复制到合并序列尾。

```
import java.lang.Math;
import java.util.Scanner;
import java.util.Arrays;

public class Test1
{
        public static int[] mergeSort(int[] a)
        {
        int[] arr = Arrays.copyOf(a, a.length);
        if (arr.length < 2)
        {
            return arr;
        }
        int middle = (int) Math.floor(arr.length/2);
        int[] left = Arrays.copyOfRange(arr, 0, middle);
        int[] right = Arrays.copyOfRange(arr, middle, arr.length);
        return merge(mergeSort(left), mergeSort(right));
        }

        public static int[] merge(int[] left, int[] right)
        {
        int[] result = new int[left.length + right.length];
        int i = 0;
        while (left.length > 0 && right.length > 0)
        {
            if (left[0] <= right[0])
            {
                result[i++] = left[0];
                left = Arrays.copyOfRange(left, 1, left.length);
            }
            else
            {
                result[i++] = right[0];
                right = Arrays.copyOfRange(right, 1, right.length);
            }
        }
        while (left.length > 0)
        {
            result[i++] = left[0];
            left = Arrays.copyOfRange(left, 1, left.length);
        }
        while (right.length > 0)
        {
            result[i++] = right[0];
            right = Arrays.copyOfRange(right, 1, right.length);
        }
        return result;
    }
```

```
    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*101-50);
        }
        array = mergeSort(array);
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```

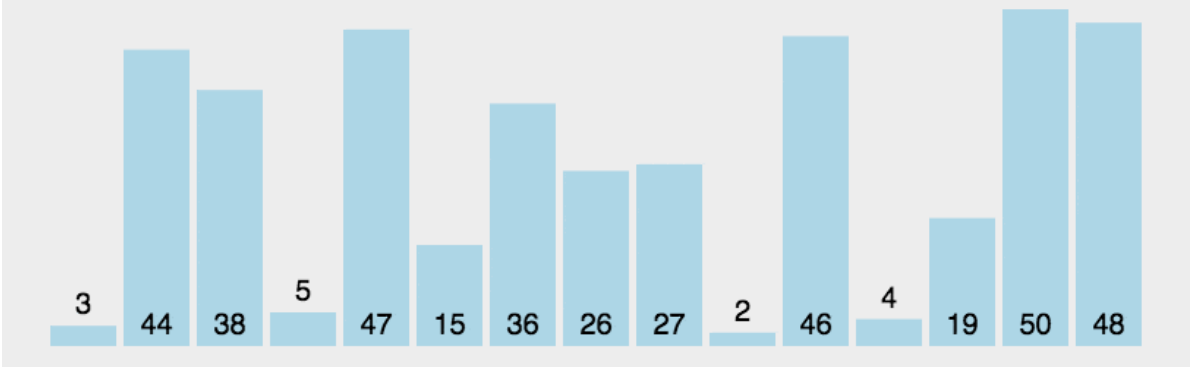


6、快速排序

- 从数列中挑出一个元素，称为 "基准"（pivot）；
- 重新排序数列，所有元素比基准值小的摆放在基准前面，所有元素比基准值大的摆在基准的后面（相同的数可以到任一边）。在这个分区退出之后，该基准就处于数列的中间位置。这个称为分区（partition）操作；
- 递归地（recursive）把小于基准值元素的子数列和大于基准值元素的子数列排序；

```
import java.lang.Math;
import java.util.Scanner;
import java.util.Arrays;

public class Test1
{
    public static int[] quickSort(int[] a, int left, int right) {
        if (left < right) {
            int partitionIndex = partition(a, left, right);
            quickSort(a, left, partitionIndex - 1);
            quickSort(a, partitionIndex + 1, right);
        }
        return a;
```

```
    }

    public static int partition(int[] a, int left, int right) {
        int pivot = left;
        int index = pivot + 1;
        for (int i = index; i <= right; i++) {
            if (a[i] < a[pivot]) {
                swap(a, i, index);
                index++;
            }
        }
        swap(a, pivot, index - 1);
        return index - 1;
    }

    public static void swap(int[] a, int i, int j) {
        int t = a[i];
        a[i] = a[j];
        a[j] = t;
    }

    public static void main(String[] args)
    {
        int[] array = new int[5];
        for(int i=0;i<5;i++)
        {
            array[i] = (int)(Math.random()*101-50);
        }
        array = quickSort(array,0,array.length-1);
        for(int i=0;i<5;i++)
        {
            System.out.print(array[i]+"\t");
        }
    }
}
```




```
import java.util.Scanner;
class Test9
{
    public static int reg_product(int a)
    {
        if(a==1)
```

```
            return 1;
        else
            return reg_product(a-1)*a;
    }

    public static int fab(int a)
    {
        if(a==1)
        {
            return 1;
        }
        else if (a==2)
        {
            return 1;
        }
        else
        {
            return fab(a-1)+fab(a-2);
        }
    }

    public static int[] fab_array(int[] array,int a)
    {
        for(int i=1;i<=a;i++)
        {
            array[i-1]=fab(i);
        }
        return array;
    }

    public static int[] fabonacci(int[] a)
    {
        for(int i=0;i<a.length;i++)
        {
            if(i==0)
                a[i] =1;
            else if(i==1)
                a[i] = 1;
            else
                a[i] = a[i-1]+a[i-2];
        }
        return a;
    }

    public static void printarray(int[] a)
    {
        for(int i=0;i<a.length;i++)
        {
            System.out.print(a[i]+"\t");
        }
    }
    public static void main(String[] args)
    {
        int a =10;
        int[] array = new int[10];
        printarray(fabonacci(array));
        // System.out.print(reg_sum(a));
        // printarray(fab_array(array,a));
```

```
    }
}
```

```
abstract class Person
{
    public abstract void run();
}

class Student extends Person
{
    public void run()
    {
        System.out.print("Student\n");
    }
}

class Employee extends Person
{
    public void run()
    {
        System.out.print("Employee\n");
    }
}

public class Test11
{
    public static void main(String[] args)
    {
        Person[] p = new Person[3];
        p[0] = new Student();
        p[1] = new Employee();
        p[2] = new Student();
        for(int i=0;i<3;i++)
        {
            p[i].run();
        }
    }
}
```

```
import java.util.Arrays;

public class Test14
{
    public static void main(String[] args)
    {
```

```
        int[] a = {1,2,-34,45,89,100,-8};
        Arrays.sort(a);
        for(int c: a)
        {
            System.out.print(c+"\t");
        }
    }
}
```

7、二维数组的运算

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

```
import java.lang.Math;
import java.util.Scanner;

public class Test4
{
    public static int[][] times(int[][] a, int[][] b)
    {
        int[][] t;
        if(a[0].length==b.length && a.length == b[0].length)
        {
            t = new int[a.length][b[0].length];
            for(int i= 0;i <a.length;i++)
            {
                for(int j = 0; j<b[0].length;j++)
                {
                    for(int k = 0; k<a[0].length; k++)
                    {
                        t[i][j]+=a[i][k]*b[k][j];
                    }
                }
            }
            return t;
        }
        else
        {
            return null;
        }

    }

    public static void main(String[] args)
    {
        int[][] a = {{1,1,1},{0,0,0},{1,0,0}};
        int[][] b = {{1,0,0},{1,0,0},{1,0,0}};
        int[][] c;
        c = times(a,b);
        for(int i=0;i<c.length;i++)
```

```
        {
            for(int j=0;j<c[0].length;j++)
            {
                System.out.print(c[i][j]+"\t");
            }
            System.out.print("\n");
        }
    }
}
```