

**【Scanner类、Random类、ArrayList 类】**

**内容**

API概述

Scanner 类

Random类

ArrayList 类

**目标**

能够明确API的使用步骤

能够使用Scanner 类获得键盘录入数据

能够使用Random类生成随机数

能够使用数组存储自定义类型并遍历

能够使用ArrayList 集合的构造方法创建ArrayList 集合对象

能够使用ArrayList 集合存储数据

能够使用ArrayList 集合中常用的方法

能够使用ArrayList 集合存储字符串并遍历

能够使用ArrayList 集合存储自定义对象并遍历

能够使用ArrayList 类作为形式参数和返回值类型

**第1章 API**

**概述**

API(Application Programming Interface)，应用程序编程接口。Java API是一本程序员的 字典 ，是JDK中提供给 我们使用的类的说明文档。这些类将底层的代码实现封装了起来，我们不需要关心这些类是如何实现的，只需要学 习这些类如何使用即可。所以我们可以通过查询API的方式，来学习Java提供的类，并得知如何使用它们。

**API使用步骤**

1. 打开帮助文档。
2. 点击显示，找到索引，看到输入框。
3. 你要找谁？在输入框里输入，然后回车。
4. 看包。java.lang 下的类不需要导包，其他需要。
5. 看类的解释和说明。
6. 学习构造方法。



 Scanner  sc = new Scanner(System.in); int i = sc.nextInt();



 import 包名.类名;



 java.util.Scanner;



 数据类型  变量名  =  new 数据类型(参数列表);



 Scanner  sc = new Scanner(System.in);



 变量名.方法名();

7. 使用成员方法。

**第2章 Scanner类**

了解了API的使用方式，我们通过Scanner 类，熟悉一下查询API，并使用类的步骤。

**2.1 什么是Scanner类**

一个可以解析基本类型和字符串的简单文本扫描器。 例如，以下代码使用户能够从 System.in 中读取一个数：

备注：System.in 系统输入指的是通过键盘录入数据。

**2.2 引用类型使用步骤**

**导包**

使用import关键字导包，在类的所有代码之前导包，引入要使用的类型，java.lang 包下的所有类无需导入。 格 式：

举例：

**创建对象**

使用该类的构造方法，创建一个该类的对象。 格式：

举例：

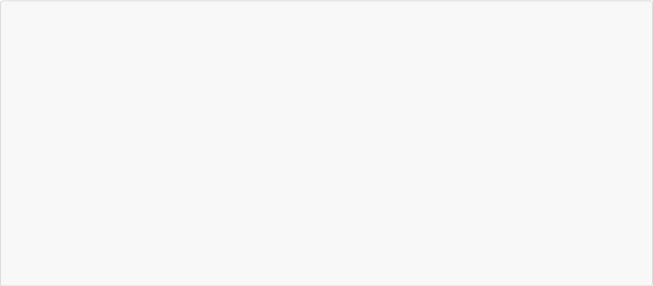
**调用方法**

调用该类的成员方法，完成指定功能。 格式：

举例：



 int i = sc.nextInt(); // 接收一个键盘录入的整数



 //1. 导包

import java.util.Scanner ;

public class Demo01\_Scanner {

    public static void main(String[] args) {         //2. 创建键盘录入数据的对象

        Scanner sc = new Scanner (System.in);

        //3. 接收数据

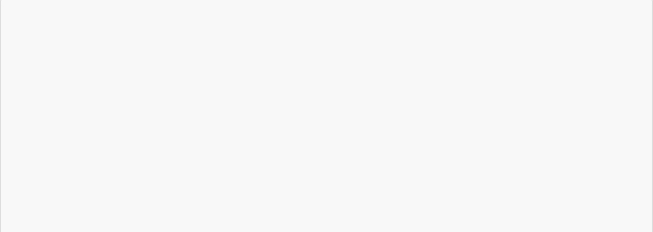
        System.out.println ("请录入一个整数：");         int i = sc.nextInt();

        //4. 输出数据

        System.out.println ("i:"+i);

    }

}



**2.3 Scanner使用步骤**

**查看类**

java.util.Scanner ：该类需要import导入后使用。

**查看构造方法**

public Scanner(InputStream source) : 构造一个新的 Scanner ，它生成的值是从指定的输入流扫描的。 **查看成员方法**

public int nextInt() ：将输入信息的下一个标记扫描为一个 int 值。

使用Scanner 类，完成接收键盘录入数据的操作，代码如下：

**2.4 练习**

**求和**

键盘录入两个数据并求和，代码如下：

 import  java.util.Scanner ;

public class Test01Scanner {

  public static void main(String[] args) {     // 创建对象

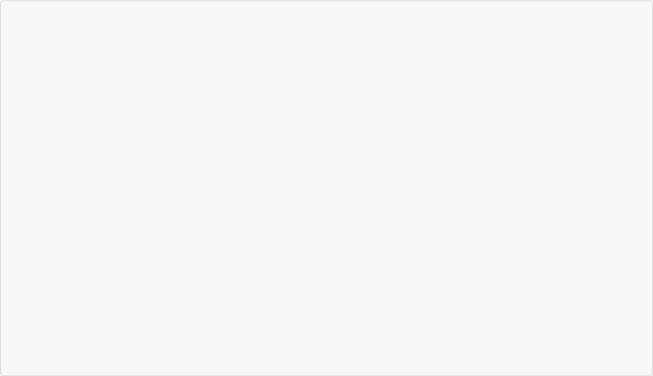
    Scanner sc = new Scanner (System.in);     // 接收数据

    System.out.println ("请输入第一个数据：");     int a = sc.nextInt();

    System.out.println ("请输入第二个数据：");     int b = sc.nextInt();

    // 对数据进行求和

    int sum = a + b;



 new 类名(参数列表)；



 new Scanner(System.in)；

    System.out.println ( sum:  + sum);

  }

}

**取最值**

键盘录入三个数据并获取最大值，代码如下：

 import java.util.Scanner ;

public class Test02Scanner {

  public static void main(String[] args) {

    // 创建对象

    Scanner sc = new Scanner (System.in);

    // 接收数据

    System.out.println ("请输入第一个数据：");

    int a = sc.nextInt();

    System.out.println ("请输入第二个数据：");

    int b = sc.nextInt();

    System.out.println ("请输入第三个数据：");

    int c = sc.nextInt();

    // 如何获取三个数据的最大值

    int temp = (a > b ? a :b);

    int max = (temp > c ? temp : c);

    System.out.println ("max:" + max);

  }

}

**2.5 匿名对象【了解】**

**概念**

创建对象时，只有创建对象的语句，却没有把对象地址值赋值给某个变量。虽然是创建对象的简化写法，但是应用 场景非常有限。

**匿名对象** ：没有变量名的对象。

格式：

举例：

**应用场景**

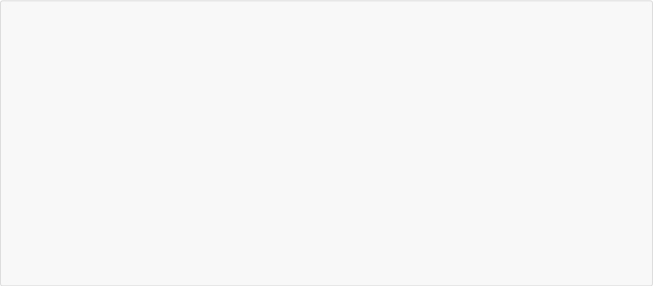
1. 创建匿名对象直接调用方法，没有变量名。



 new Scanner(System.in).nextInt();



 new Scanner(System.in).nextInt(); new Scanner (System.in).nextInt();



 class Test {

    public static void main(String[] args) {         // 普通方式

        Scanner sc = new Scanner(System.in);          input(sc);

        //匿名对象作为方法接收的参数

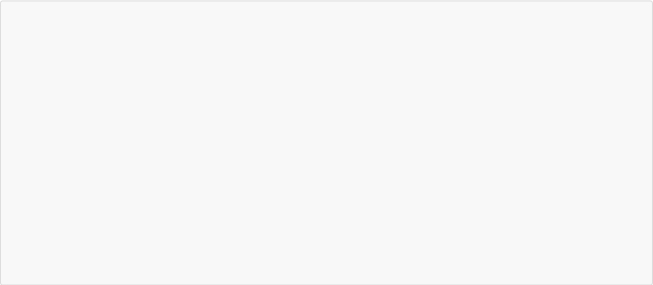
        input(new Scanner (System.in));

    }

   public static void input(Scanner  sc){         System.out.println (sc);

    }

}



 class Test2 {

     public static void main(String[] args) {         // 普通方式

        Scanner sc = getScanner();

    }

    public static Scanner getScanner(){         //普通方式

        //Scanner sc = new Scanner(System.in);

        //return sc;

        //匿名对象作为方法返回值

        return new Scanner (System.in);

    }

}

2. 一旦调用两次方法，就是创建了两个对象，造成浪费，请看如下代码。

小贴士：一个匿名对象，只能使用一次。

3. 匿名对象可以作为方法的参数和返回值

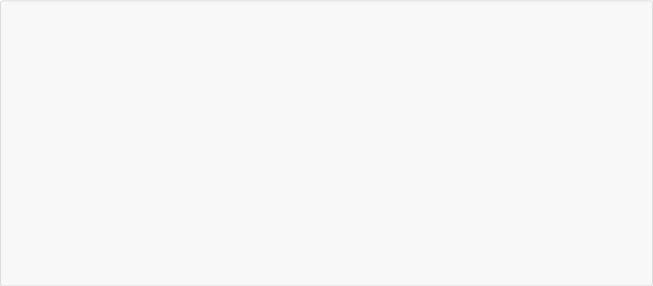
作为参数：

作为返回值

**第3章 Random类 3.1 什么是Random类**



 Random r = new Random(); int i = r.nextInt ();



 //1. 导包

import java.util.Random;

public class Demo01\_Random {

    public static void main(String[] args) {         //2. 创建键盘录入数据的对象

        Random r = new Random();

        for(int i = 0; i < 3;i++){

            //3. 随机生成一个数据

            int number = r.nextInt(10);             //4. 输出数据

            System.out.println ("number:" + number);         }

    }

}



此类的实例用于生成伪随机数。

例如，以下代码使用户能够得到一个随机数：

**3.2 Random使用步骤**

**查看类**

java.util.Random ：该类需要 import 导入使后使用。

**查看构造方法**

public Random() ：创建一个新的随机数生成器。

**查看成员方法**

：返回一个伪随机数，范围在 0 （包括）和 指定值 n （不包括）之间的

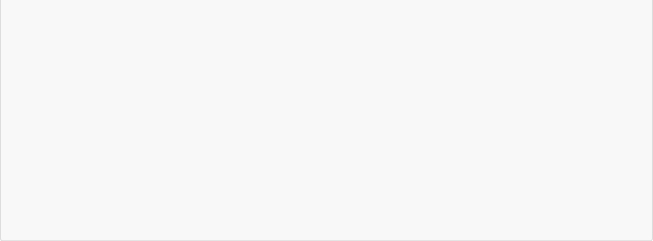
|  |  |
| --- | --- |
| public int nextInt(int n) | |
| int | 值。 |

使用Random类，完成生成3个10以内的随机整数的操作，代码如下：

备注：创建一个 Random 对象，每次调用 nextInt() 方法，都会生成一个随机数。 **3.3 练习**

**获取随机数**

获取1-n之间的随机数，包含n，代码如下：



 // 导包

import java.util.Random;

public class Test01Random {

  public static void main(String[] args) {     int n = 50;

    // 创建对象

    Random r = new Random();

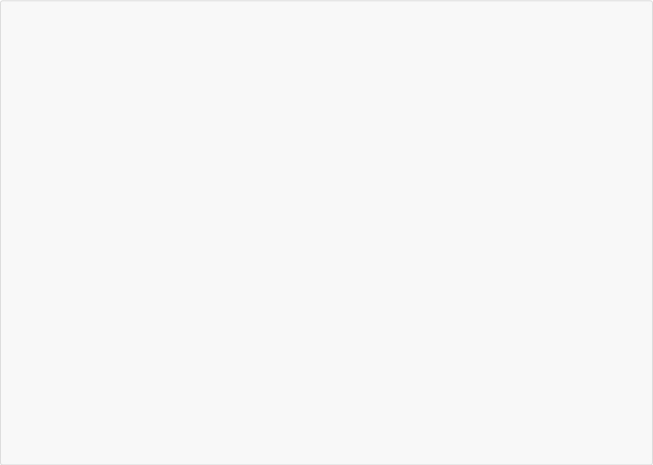
    // 获取随机数

    int number = r.nextInt(n) + 1;

    // 输出随机数

    System.out.println ("number:"  + number);   }

}



**猜数字小游戏**

游戏开始时，会随机生成一个1-100 之间的整数 number 。玩家猜测一个数字 guessNumber ，会与 number 作比 较，系统提示大了或者小了，直到玩家猜中，游戏结束。

小贴士：先运行程序代码，理解此题需求，经过分析后，再编写代码

 // 导包

import java.util.Random;

public class Test02Random {

  public static void main(String[] args) {

    // 系统产生一个随机数1‐100之间的。

    Random r = new Random();

    int number = r.nextInt(100) + 1;

    while(true){

      // 键盘录入我们要猜的数据

      Scanner sc = new Scanner(System.in);

      System.out.println("请输入你要猜的数字(1‐100)：");

      int guessNumber  = sc.nextInt();

      // 比较这两个数据(用if语句)

      if (guessNumber > number) {

        System.out.println ("你猜的数据" + guessNumber + "大了");

      } else if (guessNumber < number) {

        System.out.println ("你猜的数据" + guessNumber + "小了");

      } else {

        System.out.println ("恭喜你,猜中了");

        break;

      }

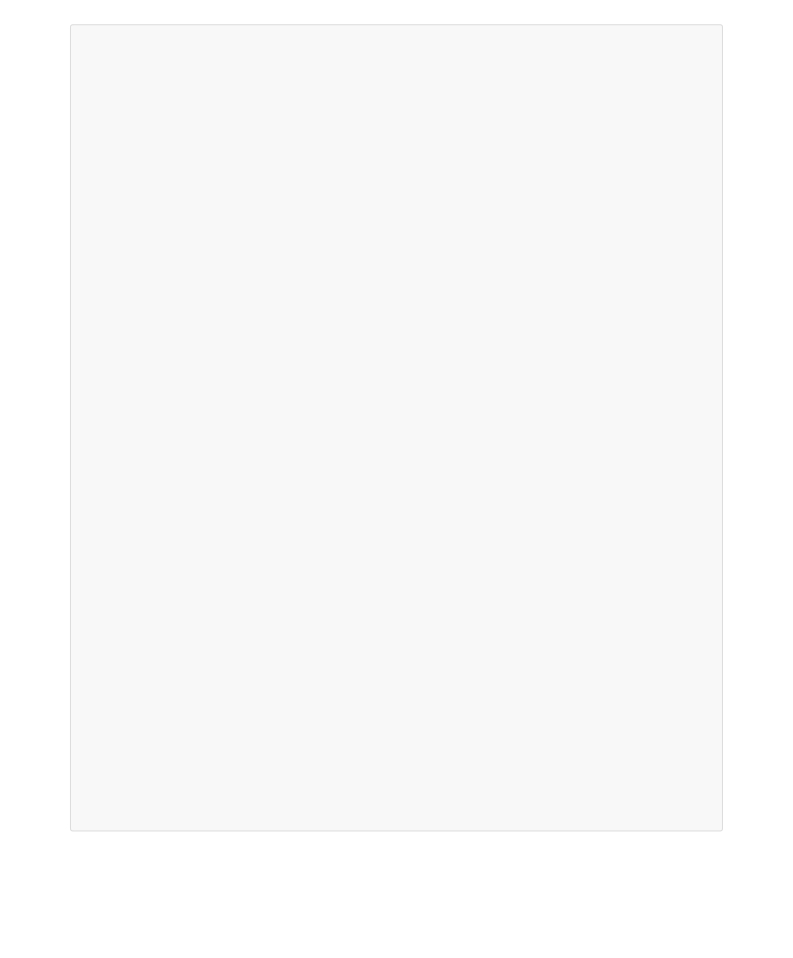
    }

  }

}

**第4章 ArrayList类**

**4.1 引入——对象数组**



使用学生数组，存储三个学生对象，代码如下：

 public class Student {

  private String name;

  private int age;

  public Student() {

  }

  public Student(String name, int age) {

    this.name = name;

    this.age = age;

  }

  public String getName() {

    return name;

  }

  publicvoid setName(String name) {

    this.name = name;

  }

  publicint getAge() {

    return age;

  }

  publicvoid setAge(int age) {

    this.age = age;

  }

}

public class Test01StudentArray {

  public static void main(String[] args) {

    //创建学生数组

    Student[] students = new Student [3];

    //创建学生对象

    Student s1 = new Student ("曹操",40);

    Student s2 = new Student ("刘备",35);

    Student s3 = new Student ("孙权",30);

    //把学生对象作为元素赋值给学生数组

    students [0] = s1;

    students [1] = s2;

    students [2] = s3;

    //遍历学生数组

    for(int x=0; x<students.length; x++) {

      Student s = students[x];

      System.out.println(s.getName ()+"‐‐‐"+s.getAge());

    }

  }

}

到目前为止，我们想存储对象数据，选择的容器，只有对象数组。而数组的长度是固定的，无法适应数据变化的需 求。为了解决这个问题，Java提供了另一个容器 java.util.ArrayList 集合类,让我们可以更便捷的存储和操作对 象数据。

**4.2 什么是ArrayList类**



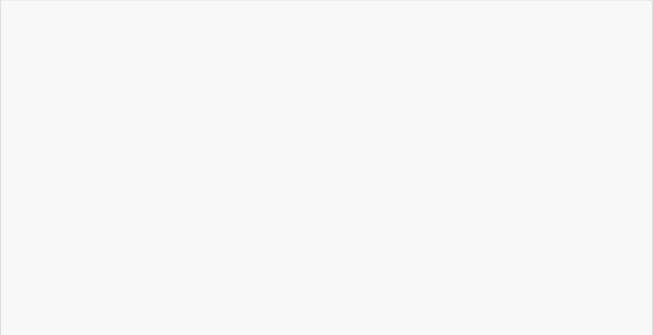
 ArrayList<String>，ArrayList<Student>



 ArrayList<String> list = new ArrayList<String>();



 ArrayList<String> list = new ArrayList<>();



java.util.ArrayList 是大小**可变的数组**的实现，存储在内的数据称为元素。此类提供一些方法来操作内部存储 的元素。 ArrayList 中可不断添加元素，其大小也自动增长。

**4.3 ArrayList使用步骤**

**查看类**

java.util.ArrayList <E> ：该类需要 import 导入使后使用。

<E> ，表示一种指定的数据类型，叫做泛型。 E ，取自Element （元素）的首字母。在出现 E 的地方，我们使 用一种引用数据类型将其替换即可，表示我们将存储哪种引用类型的元素。代码如下：

**查看构造方法**

public ArrayList() ：构造一个内容为空的集合。

基本格式:

在JDK 7后,右侧泛型的尖括号之内可以留空，但是<>仍然要写。简化格式：

**查看成员方法**

public boolean add(E e) ： 将指定的元素添加到此集合的尾部。

参数 E e ，在构造ArrayList 对象时， <E> 指定了什么数据类型，那么 add(E e) 方法中，只能添加什么数据 类型的对象。

使用ArrayList 类，存储三个字符串元素，代码如下：

 public  class Test02StudentArrayList {

  public static void main(String[] args) {

    //创建学生数组

    ArrayList <String> list = new ArrayList<>();

    //创建学生对象

    String s1 = "曹操";

    String s2 = "刘备";

    String s3 = "孙权";

    //打印学生ArrayList集合

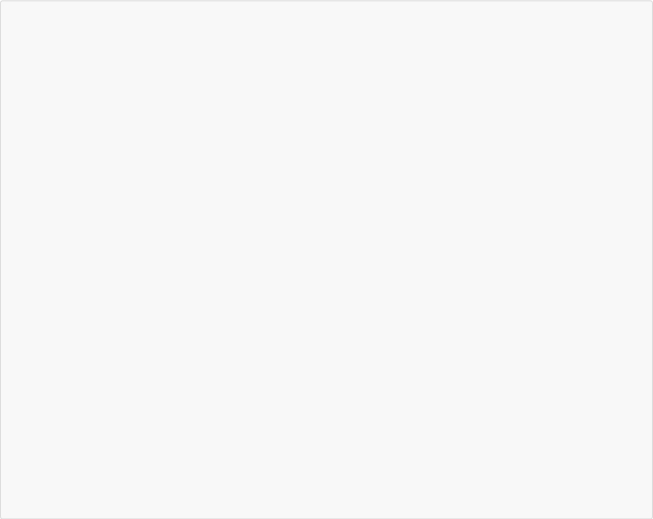
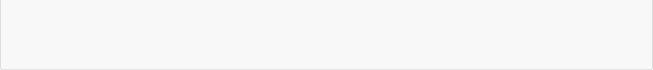
    System.out.println (list);

    //把学生对象作为元素添加到集合

    list.add(s1);

    list.add(s2);

    list.add(s3);



    //打印学生ArrayList集合

    System.out.println (list);

  }

}

**4.4 常用方法和遍历**

对于元素的操作,基本体现在——增、删、查。常用的方法有：

public boolean add(E e) ：将指定的元素添加到此集合的尾部。

public E remove(int index) ：移除此集合中指定位置上的元素。返回被删除的元素。 public E get(int index) ：返回此集合中指定位置上的元素。返回获取的元素。

public int size() ：返回此集合中的元素数。遍历集合时，可以控制索引范围，防止越界。

这些都是最基本的方法，操作非常简单，代码如下:

 public class Demo01ArrayListMethod {

  public static void main(String[] args) {

    //创建集合对象

    ArrayList <String> list = new ArrayList<String>();

    //添加元素

    list.add("hello" );

    list.add("world" );

    list.add("java");

    //public E get(int index):返回指定索引处的元素

    System.out.println ("get:"+list.get(0));

    System.out.println ("get:"+list.get(1));

    System.out.println ("get:"+list.get(2));

    //public int size(): 返回集合中的元素的个数

    System.out.println ("size:" +list.size());

    //public E remove(int index):删除指定索引处的元素，返回被删除的元素

    System.out.println ("remove:" +list.remove(0));

    //遍历输出

    for(int i = 0; i < list.size(); i++){

      System.out.println(list.get(i));

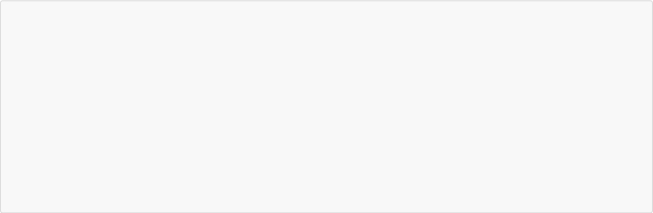
    }

  }

}

**4.5 如何存储基本数据类型**

ArrayList 对象不能存储基本类型，只能存储引用类型的数据。类似 **<int> 不能写**，但是存储基本数据类型对应的 包装类型是可以的。所以，想要存储基本类型数据， <> 中的数据类型，必须转换后才能编写，转换写法如下：



 public class Demo02ArrayListMethod {

  public static void main(String[] args) {

    ArrayList <Integer > list = new ArrayList<Integer>();     list.add(1);

    list.add(2);

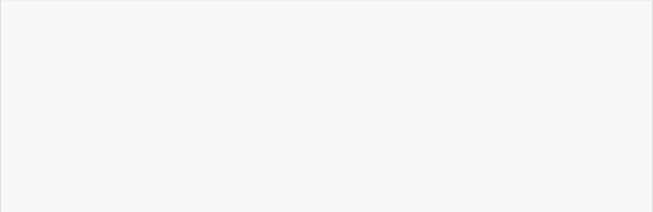
    list.add(3);

    list.add(4);

    System.out.println (list);

  }

}



short

Short

long

Long

double

Double

boolean

Boolean

|  |  |
| --- | --- |
| **基本类型** | **基本类型包装类** |
| byte | Byte |
|  |  |
| int | Integer |
|  |  |
| ﬂoat | Float |
|  |  |
| char | Character |
|  |  |

我们发现，只有 Integer 和 Character 需要特殊记忆，其他基本类型只是首字母大写即可。那么存储基本类型数 据，代码如下：

**4.6 ArrayList练习**

**数值添加到集合**

生成6个1~33之间的随机整数,添加到集合,并遍历  public class Test01ArrayList {

  public static void main(String[] args) {     // 创建Random 对象

    Random random = new Random();

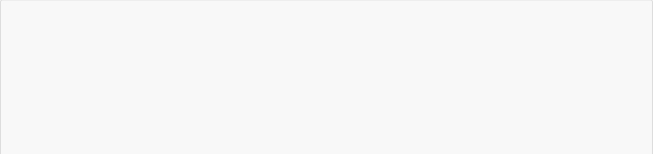
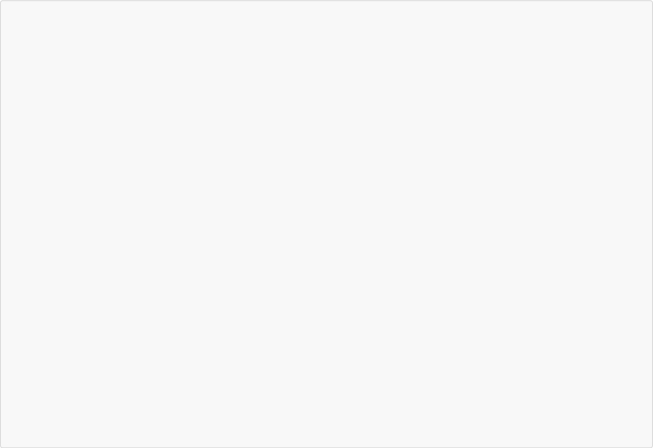
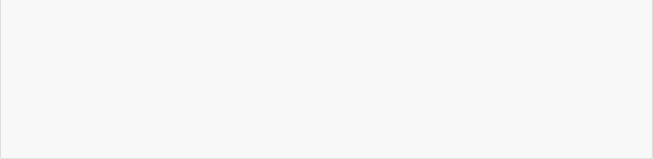
    // 创建ArrayList 对象

    ArrayList <Integer > list = new ArrayList<>();

    // 添加随机数到集合

    for (int i = 0;i < 6;i++) {

      int r = random.nextInt(33) + 1;



      list.add(r);

    }

    // 遍历集合输出

    for (int i = 0;i < list.size(); i++) {

      System.out.println(list.get(i));

    }

  }

}

**对象添加到集合**

自定义4个学生对象,添加到集合,并遍历

 public class Test02ArrayList {

  public static void main(String[] args) {

    //创建集合对象

    ArrayList <Student > list = new ArrayList<Student>();

    //创建学生对象

    Student s1 = new Student ("赵丽颖",18);

    Student s2 = new Student ("唐嫣",20);

    Student s3 = new Student ("景甜",25);

    Student s4 = new Student ("柳岩",19);

    //把学生对象作为元素添加到集合中

    list.add(s1);

    list.add(s2);

    list.add(s3);

    list.add(s4);

    //遍历集合

    for(int x = 0; x < list.size(); x++) {

      Student s = list.get(x);

      System.out.println(s.getName ()+"‐‐‐"+s.getAge());

    }

  }

}

**打印集合方法**

定义以指定格式打印集合的方法(ArrayList 类型作为参数)，使用{}扩起集合，使用@分隔每个元素。格式参照 {元素 @元素@元素}。

 public class Test03ArrayList {

  public static void main(String[] args) {

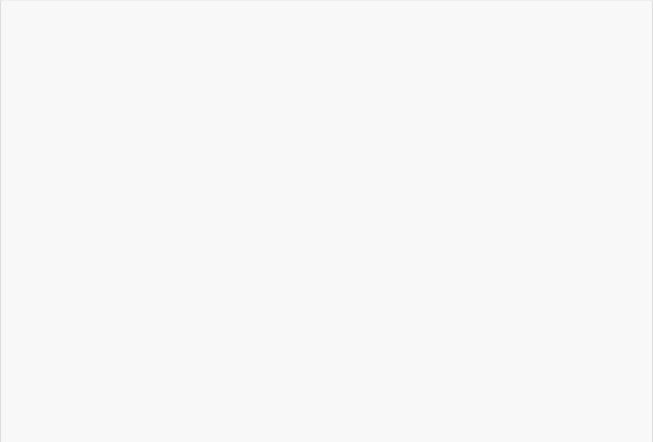
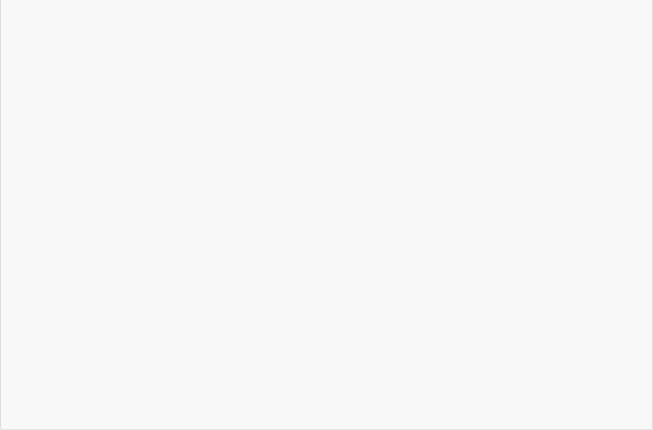
    // 创建集合对象

    ArrayList <String> list = new ArrayList<String>();

    // 添加字符串到集合中

    list.add("张三丰");

    list.add("宋远桥");



    list.add( 张无忌 );

    list.add("殷梨亭");

    // 调用方法

    printArrayList (list);

  }

  public static void printArrayList(ArrayList <String> list) {

    // 拼接左括号

    System.out.print("{");

    // 遍历集合

    for (int i = 0;i < list.size(); i++) {

      // 获取元素

      String s = list.get(i);

      // 拼接@符号

      if (i != list.size() ‐ 1) {

        System.out.print(s + "@");

      } else {

        // 拼接右括号

        System.out.print(s + "}");

      }

    }

  }

}

**获取集合方法**

定义获取所有偶数元素集合的方法(ArrayList 类型作为返回值)

 public class Test04ArrayList {

  public static void main(String[] args) {

    // 创建Random 对象

    Random random = new Random();

    // 创建ArrayList 对象

    ArrayList <Integer > list = new ArrayList<>();

    // 添加随机数到集合

    for (int i = 0;i < 20;i++) {

      int r = random.nextInt(1000) + 1;

      list.add(r);

    }

    // 调用偶数集合的方法

    ArrayList <Integer > arrayList = getArrayList(list);

    System.out.println (arrayList );

  }

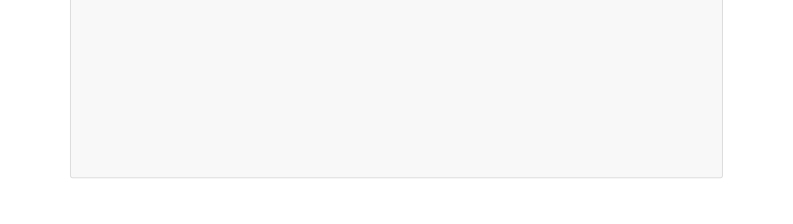
  public static ArrayList<Integer> getArrayList(ArrayList <Integer > list) {     // 创建小集合,来保存偶数

    ArrayList <Integer > smallList = new ArrayList<>();

    // 遍历list

    for (int i = 0;i < list.size(); i++) {

      // 获取元素



      Integer num = list.get(i);       // 判断为偶数,添加到小集合中       if (num % 2 == 0){         smallList.add(num);       }

    }

    // 返回小集合

    return smallList;

  }

}