1.ArrayList

集合和数组的优势对比：

1. 长度可变
2. 添加数据的时候不需要考虑索引，默认将数据添加到末尾

1.1 ArrayList类概述

- 什么是集合

  提供一种存储空间可变的存储模型，存储的数据容量可以发生改变

- ArrayList集合的特点

  长度可以变化，只能存储引用数据类型。

- 泛型的使用

  用于约束集合中存储元素的数据类型

1.2 ArrayList类常用方法

1.2.1 构造方法

方法名	说明
public ArrayList()	创建一个空的集合对象

1.2.2 成员方法

方法名	说明
public boolean add(要添加的元素)	将指定的元素追加到此集合的末尾
public boolean remove(要删除的元素)	删除指定元素,返回值表示是否删除成功
public E remove(int index)	删除指定索引处的元素，返回被删除的元素
public E set(int index,E element)	修改指定索引处的元素，返回被修改的元素
public E get(int index)	返回指定索引处的元素
public int size()	返回集合中的元素的个数

1.2.3 示例代码

```
public class ArrayListDemo02 {
    public static void main(String[] args) {
```

```
        ArrayList<String> array = new ArrayList<String>();

        //添加元素
        array.add("hello");
        array.add("world");
        array.add("java");

        //public boolean remove(Object o)：删除指定的元素，返回删除是否成功
        //        System.out.println(array.remove("world"));
        //        System.out.println(array.remove("javaee"));

        //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(array.remove(1));

        //IndexOutOfBoundsException
        //        System.out.println(array.remove(3));

        //public E set(int index,E element)：修改指定索引处的元素，返回被修改的元素
        //        System.out.println(array.set(1,"javaee"));

        //IndexOutOfBoundsException
        //        System.out.println(array.set(3,"javaee"));

        //public E get(int index)：返回指定索引处的元素
        //        System.out.println(array.get(0));
        //        System.out.println(array.get(1));
        //        System.out.println(array.get(2));
        //System.out.println(array.get(3)); //？？？？？？ 自己测试

        //public int size()：返回集合中的元素的个数
        System.out.println(array.size());

        //输出集合
        System.out.println("array:" + array);
    }
}
```

1.3 ArrayList存储字符串并遍历

1.3.1 案例需求

创建一个存储字符串的集合，存储3个字符串元素，使用程序实现在控制台遍历该集合

1.3.2 代码实现

```
public class ArrayListDemo3 {
    public static void main(String[] args) {
        //1.创建集合对象
        ArrayList<String> list = new ArrayList<>();


        //2.添加元素
```

```
        list.add("bbb");
        list.add("ccc");
        list.add("ddd");

        //3.遍历
        //快捷键: list.fori 正向遍历
        //list.forr 倒着遍历
        System.out.print("[");
        for (int i = 0; i < list.size(); i++) {
            //i 依次表示集合里面的每一个索引

            if(i == list.size() - 1){
                //最大索引
                System.out.print(list.get(i));
            }else{
                //非最大索引
                System.out.print(list.get(i) + ", ");
            }
        }
        System.out.print("]");
    }
}
```

1.4 ArrayList存储学生对象并遍历

1.4.1 案例需求

创建一个存储学生对象的集合，存储3个学生对象，使用程序实现在控制台遍历该集合

1.4.2 代码实现

```
public class ArrayListDemo4 {
    public static void main(String[] args) {
        //1.创建集合对象，用来存储数据
        ArrayList<Student> list = new ArrayList<>();

        //2.创建学生对象
        Student s1 = new Student("zhangsan",16);
        Student s2 = new Student("lisi",15);
        Student s3 = new Student("wangwu",18);

        //3.把学生对象添加到集合中
        list.add(s1);
        list.add(s2);
        list.add(s3);

        //4.遍历
        for (int i = 0; i < list.size(); i++) {
            //i 依次表示集合中的每一个索引

            Student stu = list.get(i);
```

```
    }


  }
}
```

1.5 查找用户的索引

需求：

1，main方法中定义一个集合，存入三个用户对象。

用户属性为：id，username，password

2，要求：定义一个方法，根据id查找对应的学生信息。

如果存在，返回索引

如果不存在，返回-1

代码示例：

```
public class ArrayListDemo6 {
  public static void main(String[] args) {
    /*需求：
    1，main方法中定义一个集合，存入三个用户对象。
    用户属性为：id，username，password
    2，要求：定义一个方法，根据id查找对应的学生信息。
    如果存在，返回索引
    如果不存在，返回-1*/


    //1.创建集合对象
    ArrayList<User> list = new ArrayList<>();

    //2.创建用户对象
    User u1 = new User("heima001", "zhangsan", "123456");
    User u2 = new User("heima002", "lisi", "1234");
    User u3 = new User("heima003", "wangwu", "1234qwer");

    //3.把用户对象添加到集合当中
    list.add(u1);
    list.add(u2);
    list.add(u3);

    //4.调用方法，通过id获取对应的索引
    int index = getIndex(list, "heima001");

    System.out.println(index);


  }
```

```
    //1.我要干嘛？  根据id查找对应的学生信息
    //2.我干这件事情需要什么才能完成？  集合 id
    //3.方法的调用处是否需要继续使用方法的结果？
    //要用必须返回，不要用可以返回也可以不返回
    //明确说明需要有返回值 int
    public static int getIndex(ArrayList<User> list, String id) {
        //遍历集合得到每一个元素
        for (int i = 0; i < list.size(); i++) {
            User u = list.get(i);
            String uid = u.getId();
            if(uid.equals(id)){
                return i;
            }
        }
        //因为只有当集合里面所有的元素都比较完了，才能断定id是不存在的。
        return -1;
    }
}
```

1.6 判断用户的是否存在

```
public class ArrayListDemo5 {
    public static void main(String[] args) {
        /* 需求：
        1，main方法中定义一个集合，存入三个用户对象。
        用户属性为：id，username，password
        2，要求：定义一个方法，根据id查找对应的学生信息。
        如果存在，返回true
        如果不存在，返回false*/

        //1.定义集合
        ArrayList<User> list = new ArrayList<>();

        //2.创建对象
        User u1 = new User("heima001","zhangsan","123456");
        User u2 = new User("heima002","lisi","12345678");
        User u3 = new User("heima003","wangwu","1234qwer");

        //3.把用户对象添加到集合当中
        list.add(u1);
        list.add(u2);
        list.add(u3);

        //4.调用方法，查询id是否存在
        boolean result = contains(list, "heima001");
        System.out.println(result);

    }
```

```
    //1.我要干嘛？ 我要根据id查询学生是否存在
    //2.我干这件事情，需要什么才能完成？ 集合 id
    //3.方法的调用处是否需要使用方法的结果？
    //如果要用，必须返回，如果不用，可以返回也可以不返回
    //但是本题明确说明需要返回
    public static boolean contains(ArrayList<User> list, String id){
        //循环遍历集合，得到集合里面的每一个元素
        //再进行判断

        for (int i = 0; i < list.size(); i++) {
            //i 索引  list.get(i); 元素
            User u = list.get(i);
            //判断id是否存在，我是拿着谁跟谁比较
            //需要把用户对象里面的id拿出来再进行比较。
            String uid = u.getId();
            if(id.equals(uid)){
                return true;//return 关键字：作用就是结束方法。
            }
        }
        //只有当集合里面所有的元素全部比较完毕才能认为是不存在的。
        return false;
    }

}
```

2.学生管理系统

2.1学生管理系统实现步骤

- 案例需求

  针对目前我们的所学内容，完成一个综合案例：学生管理系统。该系统主要功能如下：

  添加学生：通过键盘录入学生信息，添加到集合中

  删除学生：通过键盘录入要删除学生的学号，将该学生对象从集合中删除

  修改学生：通过键盘录入要修改学生的学号，将该学生对象其他信息进行修改

  查看学生：将集合中的学生对象信息进行展示

  退出系统：结束程序

- 实现步骤

  1. 定义学生类，包含以下成员变量

     private String sid // 学生id

     private String name // 学生姓名

     private String age // 学生年龄

     private String address // 学生所在地

  2. 学生管理系统主界面的搭建步骤

循环完成功能结束后再次回到主界面

3. 学生管理系统的添加学生功能实现步骤

   3.1 定义一个方法，接收ArrayList集合 3.2 方法内完成添加学生的功能 ①键盘录入学生信息 ②根据录入的信息创建学生对象 ③将学生对象添加到集合中 ④提示添加成功信息 3.3 在添加学生的选项里调用添加学生的方法

4. 学生管理系统的查看学生功能实现步骤

   4.1 定义一个方法，接收ArrayList集合 4.2 方法内遍历集合，将学生信息进行输出 4.3 在查看所有学生选项里调用查看学生方法

5. 学生管理系统的删除学生功能实现步骤

   5.1 定义一个方法，接收ArrayList集合 5.2 方法中接收要删除学生的学号 5.3 遍历集合，获取每个学生对象 5.4 使用学生对象的学号和录入的要删除的学号进行比较,如果相同，则将当前学生对象从集合中删除 5.5 在删除学生选项里调用删除学生的方法

6. 学生管理系统的修改学生功能实现步骤

   6.1 定义一个方法，接收ArrayList集合 6.2 方法中接收要修改学生的学号 6.3 通过键盘录入学生对象所需的信息，并创建对象 6.4 遍历集合，获取每一个学生对象。并和录入的修改学生学号进行比较.如果相同，则使用新学生对象替换当前学生对象 6.5 在修改学生选项里调用修改学生的方法

7. 退出系统

   使用System.exit(0);退出JVM

2.2学生类的定义

```
package com.itheima.studentsystem;

public class Student {
    private String id;
    private String name;
    private int age;
    private String address;

    //下面是空参，有参，get和set方法
}
```

2.3测试类的定义

```
public class StudentSystem {
    public static void main(String[] args) {
        ArrayList<Student> list = new ArrayList<>();
        loop:
        while (true) {
            System.out.println("----------------欢迎来到黑马学生管理系统----------------");
            System.out.println("1:添加学生");
            System.out.println("2:删除学生");

            System.out.println("3:修改学生");
```

```
            System.out.println("5:退出");
            System.out.println("请输入您的选择：");
            Scanner sc = new Scanner(System.in);
            String choose = sc.next();
            switch (choose) {
                case "1" -> addStudent(list);
                case "2" -> deleteStudent(list);
                case "3" -> updateStudent(list);
                case "4" -> queryStudent(list);
                case "5" -> {
                    System.out.println("退出");
                    //break loop;
                    System.exit(0);//停止虚拟机运行
                }
                default -> System.out.println("没有这个选项");
            }
        }
    }

    //添加学生
    public static void addStudent(ArrayList<Student> list) {
        //利用空参构造先创建学生对象
        Student s = new Student();

        Scanner sc = new Scanner(System.in);
        String id = null;
        while (true) {
            System.out.println("请输入学生的id");
            id = sc.next();
            boolean flag = contains(list, id);
            if(flag){
                //表示id已经存在，需要重新录入
                System.out.println("id已经存在，请重新录入");
            }else{
                //表示id不存在，表示可以使用
                s.setId(id);
                break;
            }
        }

        System.out.println("请输入学生的姓名");
        String name = sc.next();
        s.setName(name);

        System.out.println("请输入学生的年龄");
        int age = sc.nextInt();
        s.setAge(age);

        System.out.println("请输入学生的家庭住址");
        String address = sc.next();
        s.setAddress(address);
```

```
    //把学生对象添加到集合当中
    list.add(s);

    //提示一下用户
    System.out.println("学生信息添加成功");
}

//删除学生
public static void deleteStudent(ArrayList<Student> list) {
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入要删除的id");
    String id = sc.next();
    //查询id在集合中的索引
    int index = getIndex(list, id);
    //对index进行判断
    //如果-1，就表示不存在，结束方法，回到初始菜单
    if(index >= 0){
        //如果大于等于0的，表示存在，直接删除
        list.remove(index);
        System.out.println("id为：" + id + "的学生删除成功");
    }else{
        System.out.println("id不存在，删除失败");
    }
}

//修改学生
public static void updateStudent(ArrayList<Student> list) {
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入要修改学生的id");
    String id = sc.next();

    int index = getIndex(list, id);

    if(index == -1){
        System.out.println("要修改的id" + id + "不存在，请重新输入");
        return;
    }

    //当代码执行到这里，表示什么？表示当前id是存在的。
    //获取要修改的学生对象
    Student stu = list.get(index);

    //输入其他的信息并修改
    System.out.println("请输入要修改的学生姓名");
    String newName = sc.next();
    stu.setName(newName);

    System.out.println("请输入要修改的学生年龄");
    int newAge = sc.nextInt();
    stu.setAge(newAge);


    System.out.println("请输入要修改的学生家庭住址");
```

```
    stu.setAddress(newAddress);

    System.out.println("学生信息修改成功");


}


//查询学生
public static void queryStudent(ArrayList<Student> list) {
    if (list.size() == 0) {
        System.out.println("当前无学生信息，请添加后再查询");
        //结束方法
        return;
    }

    //打印表头信息
    System.out.println("id\t\t姓名\t年龄\t家庭住址");
    //当代码执行到这里，表示集合中是有数据的
    for (int i = 0; i < list.size(); i++) {
        Student stu = list.get(i);
        System.out.println(stu.getId() + "\t" + stu.getName() + "\t" + stu.getAge() + "\t" + stu.getAddress());
    }
}


//判断id在集合中是否存在
public static boolean contains(ArrayList<Student> list, String id) {
    //循环遍历集合得到里面的每一个学生对象
    /*for (int i = 0; i < list.size(); i++) {
        //拿到学生对象后，获取id并进行判断
        Student stu = list.get(i);
        String sid = stu.getId();
        if(sid.equals(id)){
            //存在，true
            return true;
        }
    }
    // 不存在false
    return false;*/
   return getIndex(list,id) >= 0;
}

//通过id获取索引的方法
public static int getIndex(ArrayList<Student> list, String id){
    //遍历集合
    for (int i = 0; i < list.size(); i++) {
        //得到每一个学生对象
        Student stu = list.get(i);
        //得到每一个学生对象的id
        String sid = stu.getId();

        //拿着集合中的学生id跟要查询的id进行比较
```

```
            //如果一样，那么就返回索引
            return i;
        }
    }
    //当循环结束之后还没有找到，就表示不存在，返回-1.
    return -1;
  }
}
```