数组既可以存储基本数据类型(存值)

又可以存储引用数据类型(存内存地址)

//案例 1　有 3 个学生 把这 3 个学生的信息存储到数组中，并遍历数组，获取得到每一个学生的信息

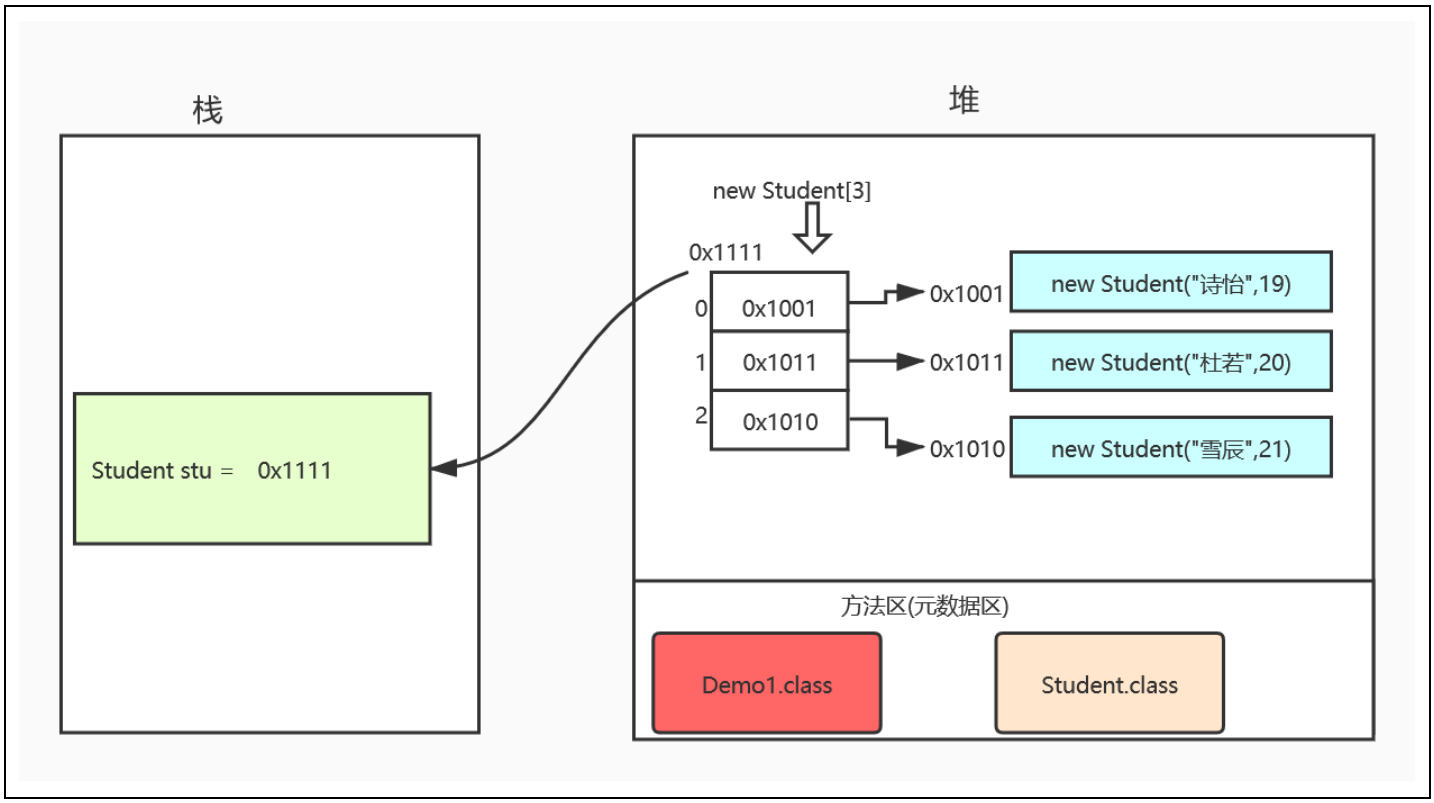
```
public class Demo1 {

    public static void main(String[] args) {



        Student[] stus = new Student[3];

        stus[0] = new Student("诗怡",19);

        stus[1] = new Student("杜若",20);

        stus[2] = new Student("雪辰",21);

    }

}

class Student{

    private String name;

    private int age;



    public Student() {

    }

    public Student(String name, int age) {

        this.name = name;

        this.age = age;

    }
```

```
    @Override
    public String toString() {
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

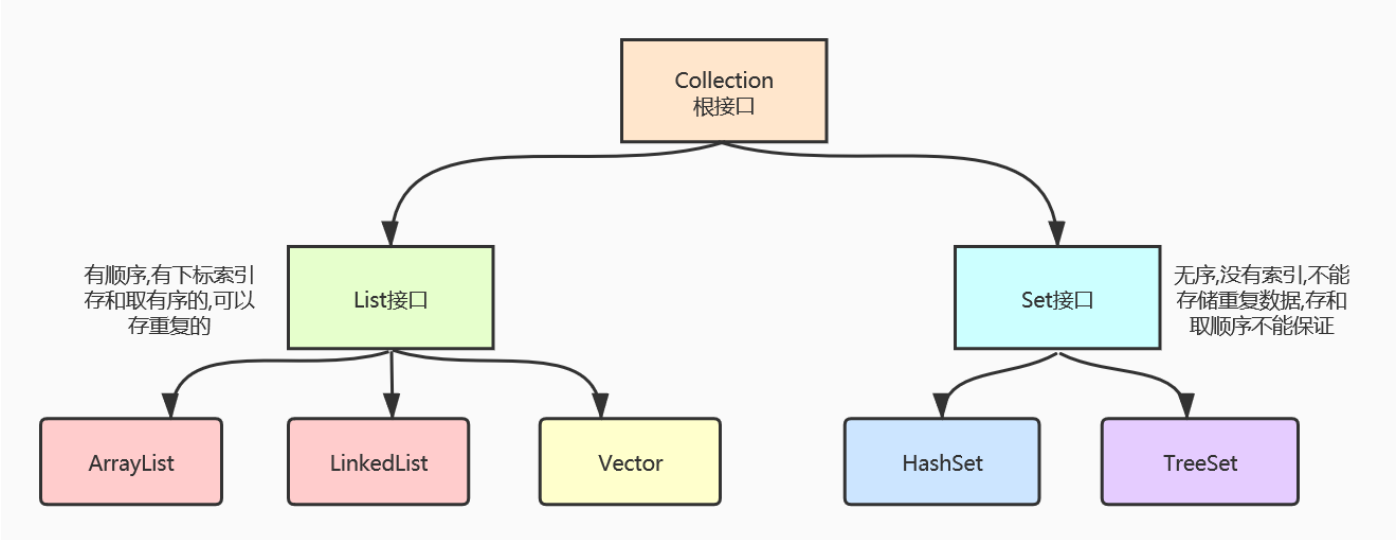
```
public class Student {
    private String name;
    private int age;

    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 Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
```

```
    public void SayHi(){

        System.out.println("我是"+name+",今年"+age);
    }

public static void main(String[] args) {
    Student[] stus=new Student[3];
    stus[0]=new Student("小明",19);
    stus[1]=new Student("小红",20);
    stus[2]=new Student("小芳",16);
    /*System.out.println(stus[0].toString());
    stus[1].SayHi();*/
    for (int i=0;i<stus.length;i++){

        /*stus[i].SayHi();*/
        System.out.println(stus[i]);
    }
}
```



●集合框架(集合的由来及集合继承体系图)



A:集合的由来

数组长度是固定,当添加的元素超过了数组的长度时需要对数组重新定义,太麻烦,java 内部给我们提供了集合类,能存储任意对象,长度是可以改变的,随着元素的增加而增加,随着元素的减少而减少

B:数组和集合的区别

区别 1 :

数组既可以存储基本数据类型,又可以存储引用数据类型,基本数据类型存储的是值,引用数据类型存储的是地址值

集合只能存储引用数据类型(对象)

注意点：集合中也可以存储基本数据类型,但是在存储的时候会自动装箱变成对象 int new Integer(); 自动封装

区别 2:

数组长度是固定的,不能自动增长

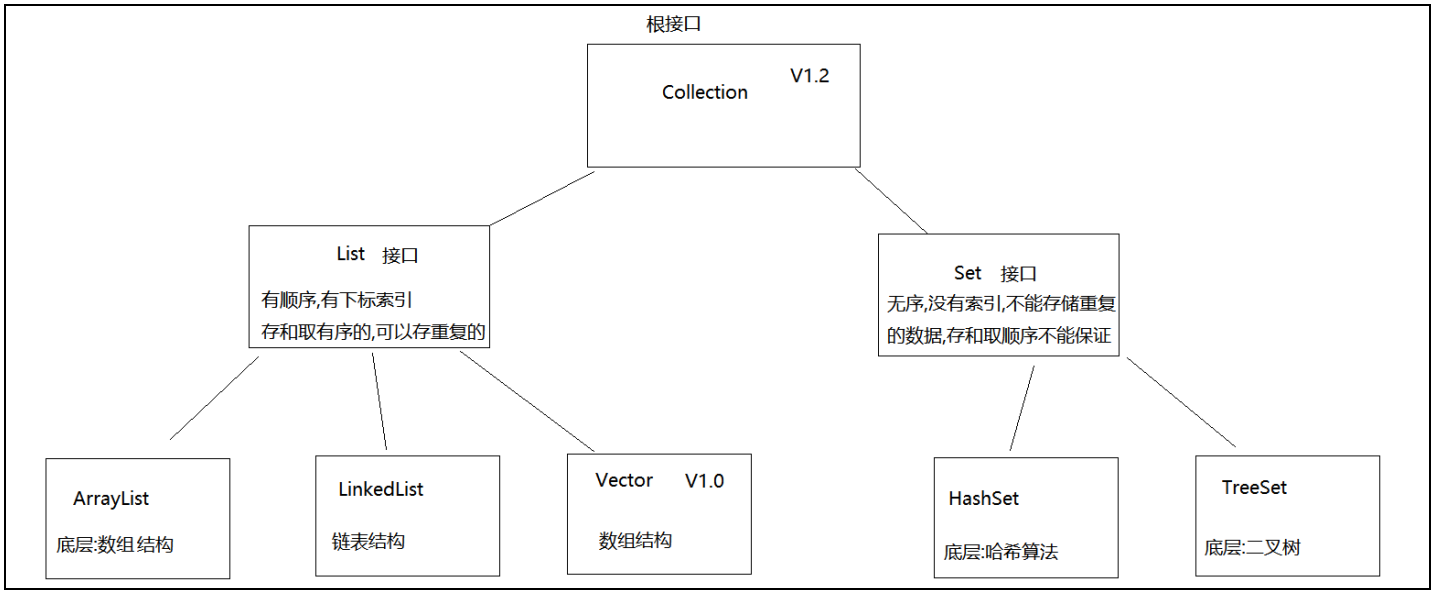
集合的长度的是可变的,可以根据元素的增加而增长

C:数组和集合什么时候用

1,如果元素个数是固定的推荐用数组

2,如果元素个数不是固定的推荐用集合

D:集合继承体系图



数组存和取都是有序的 有索引 比如 11 22 33 打印出来就是 11 22 33

●集合框架(Collection 集合的基本功能测试)

A:案例演示

基本功能演:1:

boolean add(E e) //E 就是表示 Object　1.5 之前没有 E　可以添加任何对象

```
package com.sxt.fuxi;

import java.util.ArrayList;
import java.util.Collection;

public class Demo7 {
    public static void main(String[] args) {
        Collection c = new ArrayList();//多态;接口引用指向间接子类对象
        c.add(3);//3 在这里自动包装成 Integer 类型
        c.add(34.5f);
        c.add(65.6);
        System.out.println(c.size());//显示集合中元素的个数
        c.add(true);
        c.add(new Student("小明",19));
        System.out.println(c);   //集合重写了 Object 中的 toString 方法
        System.out.println(c.size());
    }
}
class Student{
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
```

```
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

```
Collection c=new ArrayList();//父类指向子类
//boolean add(E e) //E 就是表示Object
boolean b1= c.add("abc");
boolean b2=c.add(true); //自动装箱  封装成对象
boolean b3=c.add(100);
boolean b4=c.add(new Student("张三",23));
boolean b5=c.add("abc");
System.out.println(b1);
System.out.println(b2);
System.out.println(b3);
System.out.println(b4);
System.out.println(b5);
System.out.println(c); //集合重写了Object 中的toString 方法
```

Add()方法如果是 List 集合一直都返回 True 因为 List 集合中 可以存储重复

如果是 set 集合 存储重复元素的时候就会返回 false

ArrayList 的父类的父类 重写了 toString() 所以在打印对象的引用的时候输出的结果 object 类中 tostring()的结果

基本功能演示 2：

boolean remove(Object o)

void clear()

boolean contains(Object o)　　:StringBuffer 中学过一样的方法

boolean isEmpty()

int size()

```
package com.sxt.fuxi;


import java.util.ArrayList;

import java.util.Collection;


public class Demo7 {

    public static void main(String[] args) {

        Collection c = new ArrayList();//多态;接口引用指向间接子类对象

        //add 往集合中添加元素

        boolean bo1 = c.add(3);//   3 在这里自动包装成 Integer 类型

        c.add(3);

        //基本数据类型在集合中都会自动包装成引用类型

        System.out.println(bo1);

        c.add(34.5f);

        boolean bo2 = c.add(65.6);

        System.out.println(bo2);

        System.out.println(c.size());//显示集合中元素的个数

        c.add(true);

        c.add(new Student("小明",19));

        System.out.println(c);    //集合重写了 Object 中的 toString 方法

        System.out.println(c.size());
```

```
        System.out.println("------------------");
//          boolean bo3 = c.remove(3);
//          System.out.println(bo3);
        c.remove(true);       //移除元素
        boolean bo4   =c.remove(new Student("小明",19)); //对象不能这么移除,因为集合中存储的是对象的
地址
        System.out.println(bo4);
        System.out.println(c);
        System.out.println("--------------------");
//          c.clear();      //清空集合中的所有元素
//          System.out.println(c);
//          System.out.println(c.size());
        boolean bo5 = c.contains(3);//判断是否包含了指定的元素
        System.out.println(bo5);
        System.out.println(c);
        System.out.println("----------------");
        boolean bo6 = c.isEmpty();//判断集合是否是空的
        System.out.println(bo6);


    }
}
class Student{
    private String name;
    private int age;
```

```
    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

●集合框架(集合的遍历之集合转数组遍历)[ ]

A:集合的遍历

其实就是依次获取集合中的每一个元素。

B:案例演示

把集合转成数组，可以实现集合的遍历

toArray()

```
package com.sxt.fuxi;


import java.util.ArrayList;
import java.util.Collection;


public class Demo8 {
    public static void main(String[] args) {
        Collection c = new ArrayList();
        c.add(4);
        c.add('a');
        c.add(45.6);
        c.add(true);
        c.add(false);
        c.add(new Student("张三",20));
        Object[] obj = c.toArray();
        for (Object ob:obj) {
            System.out.println(ob);
        }
    }
}
```

特殊 独立拿到姓名和年龄：

```
Object[] arr = c.toArray();
//遍历 是为了看所有元素  对元素进行操作
for (int i=0;i<arr.length;i++){

    //System.out.println(obj[i]);
    //独立拿到名字和年龄
    //向下转型
    Student s=(Student)arr[i];

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

System.out.println(c);//显示结果
```

●集合框架(Collection 集合的带 All 功能测试)

A:案例演示

带 All 的功能演示

boolean addAll(Collection c)

boolean removeAll(Collection c)

boolean containsAll(Collection c)

boolean retainAll(Collection c) //取两个集合中的交集

```
package com.sxt.fuxi;


import java.awt.desktop.AboutEvent;

import java.util.ArrayList;

import java.util.Collection;


public class Demo8 {

    public static void main(String[] args) {

        /*

         boolean addAll(Collection c)

       boolean removeAll(Collection c)

       boolean containsAll(Collection c)

        boolean retainAll(Collection c) //取两个集合中的交集

          */

         Collection c1 = new ArrayList();

         c1.add(3);

         c1.add(4);

         c1.add(5);

         c1.add(6);

         Collection c2 = new ArrayList();

         boolean bo = c2.addAll(c1);//把集合 c1 中的所有元素全部添加到 C2 集合中

         System.out.println(bo);

         System.out.println(c2);
```

```
        Collection c3 = new ArrayList();
        c3.add(3);
        c3.add(4);
//          c3.add(7);
//          boolean bo2 = c1.removeAll(c3);//从 c1 集合中移除与 c3 集合有交集的元素
//          System.out.println(bo2);
//          System.out.println(c1);
没有交集  就为 false 无法移除  还是会保存原来的值

        boolean bo3 = c1.containsAll(c3);//判断 c1 中是否包含 c3 中全部的元素
        System.out.println(bo3);
        System.out.println("------------------");
        System.out.println(c1);

        //c1 与 c3 取交集,c1 中不是交集的元素被移除,保留是交集的元素
        boolean bo4 = c1.retainAll(c3);
        System.out.println(bo4);
        System.out.println(c1);

    }
}
```

```
Collection c1 = new ArrayList();
c1.add(3);
```

```
c1.add(4);
c1.add(5);
c1.add(6);
Collection c3 = new ArrayList();
c3.add(3);
c3.add(4);
c3.add(7);

boolean bo4 = c1.retainAll(c3);
System.out.println(bo4);
System.out.println(c1);//3 4  共同拥有 交集
```

●集合框架(集合的遍历之迭代器遍历)

A:迭代器概述

集合是用来存储元素,存储的元素需要查看,那么就需要迭代(遍历)

```
public interface Iterator<E> {
        boolean hasNext();   //判断是否有下一个元素

        E next();         //获取下一个元素


    default void remove() {      //移除下一个元素
        throw new UnsupportedOperationException("remove");
    }


    default void forEachRemaining(Consumer<? super E> action) {
        Objects.requireNonNull(action);
        while (hasNext())
            action.accept(next());
```

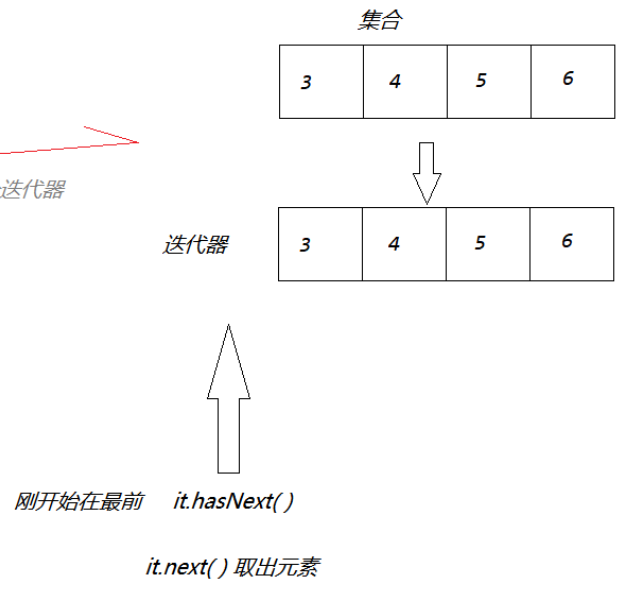
```
    }
}
```

B:案例演示：迭代器的使用

```
package com.sxt.fuxi;



import java.awt.desktop.AboutEvent;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;



public class Demo8 {

    public static void main(String[] args) {



        Collection c1 = new ArrayList();

        c1.add(3);

        c1.add(4);

        c1.add(5);

        c1.add(6);



        Iterator it = c1.iterator();   //通过对象返回一个迭代器

//          if(it.hasNext()){
```

```
//              System.out.println(it.next());
//          }
//          if(it.hasNext()){
//              System.out.println(it.next());
//          }
        while(it.hasNext()){
            Object obj = it.next();
            System.out.println(obj);
        }
    }
}
```

```
public class Demo8 {
    public static void main(String[] args) {
        Collection c1 = new ArrayList();
        c1.add(3);
        c1.add(4);
        c1.add(5);
        c1.add(6);
        Iterator it = c1.iterator();  //通过对象返回一个迭代器
        while(it.hasNext()){
            Object obj = it.next();
            System.out.println(obj);
        }
    }
}
```



添加学生对象

```
Collection c1 = new ArrayList();
c1.add(new Student("张三 1",22));
```

```
c1.add(new Student("张三 2",23));
c1.add(new Student("张三 3",24));
c1.add(new Student("张三 4",25));
Iterator it = c1.iterator();  //通过对象返回一个迭代器
while(it.hasNext())
    System.out.println(it.next());
```

单独获取学生姓名和年龄　向下转型

自定义迭代

```
Student obj=(Student)it.next();
System.out.println(obj.getAge()+"---"obj.getName());
```

```
package com.sxt.fuxi;


import java.awt.desktop.AboutEvent;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;


public class Demo8 {

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

```
        Collection c1 = new ArrayList();
        c1.add(3);
        c1.add(4);
        c1.add(5);
        c1.add(6);

        Iterator it = c1.iterator();  //通过对象返回一个迭代器
        System.out.println(it.hasNext());  //用来判断是否有下一个元素,如果达到集合的末尾,就不再用 next 取元素了
        System.out.println(it.next());  //next 不仅是取出元素,还把底层的指针在往右移动
        System.out.println(it.next());
        System.out.println(it.next());
        System.out.println(it.next());
        System.out.println(it.next());

    }
}
```

```java
package com.sxt.fuxi;


import java.util.ArrayList;

import java.util.Collection;

public class Demo8 {

    public static void main(String[] args) {


        Collection c1 = new ArrayList();

        c1.add(3);

        c1.add(4);

        c1.add(5);

        c1.add(6);

        ArrayList a = (ArrayList)c1;

       for(int i=0;i<c1.size();i++){

            Object obj = a.get(i);

            System.out.println(obj);

       }

    }

}
```

●集合框架(Collection 存储自定义对象并遍历)

A:案例演示

Collection 存储自定义对象并用迭代器遍历

```
package com.sxt.fuxi;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;


public class Demo8 {

    public static void main(String[] args) {


        Collection c1 = new ArrayList();

        c1.add(new Student("张三",19));

        c1.add(new Student("李四",29));

        c1.add(new Student("王五",22));


        Iterator it = c1.iterator();

        while(it.hasNext()){

            Object obj = it.next();   //返回的 Student 对象被提升为 Object 了

            Student stu = (Student)obj;

            System.out.println(stu.getName()+"--"+stu.getAge());
```

```
            }
        }
    }
```

●集合框架(迭代器的原理及源码解析)(了解)

A:迭代器原理

迭代器原理:迭代器是对集合进行遍历,而每一个集合内部的存储结构都是不同的,所以每一个集合存和取都是不一样,那么就需要在每一个类中定义 hasNext()和 next()方法,这样做是可以的,但是会让整个集合体系过于臃肿,迭代器是将这样的方法向上抽取出接口,然后在每个类的内部,定义自己迭代方式,这样做的好处有二,第一规定了整个集合体系的遍历方式都是 hasNext()和 next()方法,第二,代码有底层内部实现,使用者不用管怎么实现的,会用即可

B:迭代器源码解析

1,在 idea 中 ctrl + shift + n 找到 ArrayList 类

2,Ctrl+F12 查找 iterator()方法

3,查看返回值类型是 new Itr(),说明 Itr 这个类实现 Iterator 接口

4,查找 Itr 这个内部类,发现重写了 Iterator 中的所有抽象方法

```
private class Itr implements Iterator<E> {
    int cursor;       // index of next element to return
    int lastRet = -1; // index of last element returned; -1 if no such
    int expectedModCount = modCount;

    // prevent creating a synthetic constructor
```

```
    Itr() {}


    public boolean hasNext() {
        return cursor != size;     返回 false 程序停止
    }


  //是 ArrayList 中 iteractor 对接口 Iteractor 中 next 的重写实现


    public E next() {
        checkForComodification();
        int i = cursor;
        if (i >= size)
            throw new NoSuchElementException();
        Object[] elementData = ArrayList.this.elementData;
        if (i >= elementData.length)
            throw new ConcurrentModificationException();
        cursor = i + 1;        获取并把指针向后加 1
        return (E) elementData[lastRet = i];
    }


    public void remove() {
        if (lastRet < 0)
            throw new IllegalStateException();
        checkForComodification();
```

```
            try {
                ArrayList.this.remove(lastRet);
                cursor = lastRet;
                lastRet = -1;
                expectedModCount = modCount;
            } catch (IndexOutOfBoundsException ex) {
                throw new ConcurrentModificationException();
            }
        }
}
```

●集合框架(List 集合的特有功能概述和测试)

A:List 集合的特有功能概述

void add(int index,E element)

E remove(int index)

E get(int index)

E set(int index,E element)

```
package com.sxt.fuxi;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;

import java.util.List;
```

```java
public class Demo8 {

    public static void main(String[] args) {


        List list= new ArrayList(); //手动导包快捷键:alt+enter

        list.add(3);

        list.add(4);

        list.add(3);

        System.out.println(list);

        list.add(1,9);   //把 9 插入指定的索引上

        System.out.println(list.size());

        System.out.println(list);


    }

}
```

```java
List list= new ArrayList(); //手动导包快捷键:alt+enter
list.add(3);
list.add(4);
list.add(3);
System.out.println(list);
list.add(1,9);   //把9插入指定的索引上
System.out.println(list.size());
System.out.println(list);
```


```java
package com.sxt.fuxi;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;

import java.util.List;
```

```
public class Demo8 {

    public static void main(String[] args) {


        List list= new ArrayList(); //手动导包快捷键:alt+enter

         list.add(3);

         list.add(4);

         list.add(3);

         System.out.println(list);

         list.add(1,9);   //把 9 插入指定的索引上

         System.out.println(list.size());

         System.out.println(list);

         list.remove(1);//移除指定索引位置上的元素

         System.out.println(list);

         System.out.println("-----------------------");


通过索引获取元素

System.out.println(list.get(2));   //获取索引为 2 位置上的元素

         list.set(0,10);//修改指定索引位置上的数

         System.out.println(list);


    }

}
```

注意点:

```
List list=new ArrayList();
//Integer 类型
list.add(111);
list.add(222);
list.add(333);
list.add(444);

list.remove(111);//将 111 当成索引  并不会装箱操作
报异常
System.out.println(list);
```

●集合框架(List 集合存储遍历)

```
List list=new ArrayList();
 //Integer 类型
 list.add(111);
 list.add(222);
 list.add(333);
 list.add(444);

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

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

集合框架(List 集合存储学生对象并遍历)

A:案例演示

通过 size()和 get()方法结合使用遍历。

```
package com.sxt.fuxi;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;

import java.util.List;


public class Demo8 {

    public static void main(String[] args) {


        List list= new ArrayList();

        list.add(new Student("小明",19));

        list.add(new Student("小红",21));

        list.add(new Student("张韶涵",20));

        list.add(new Student("刘德华",22));
```

```
        for (int i = 0; i <list.size() ; i++) {
                Object obj = list.get(i);
            Student stu = (Student)obj;
            System.out.println(stu.getName()+"---"+stu.getAge());
        }
    }
}
```

●集合框架(并发修改异常产生的原因及解决方案)

A:案例演示

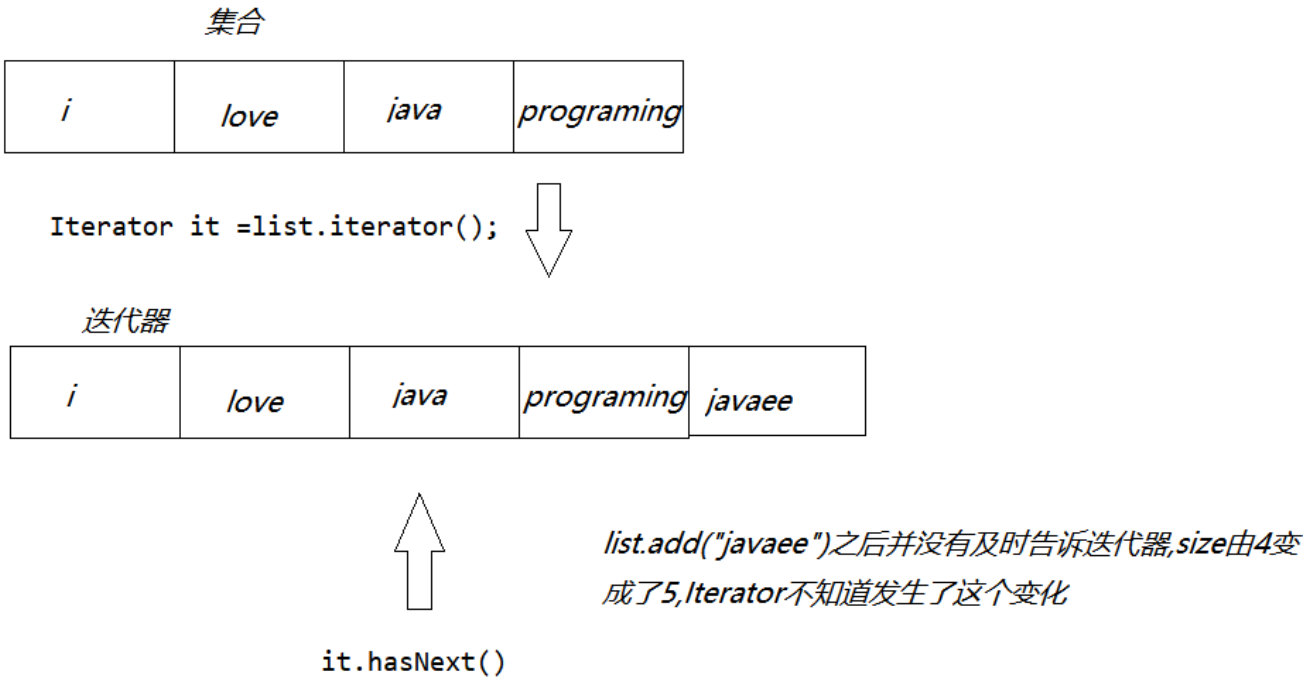
需求：我有一个集合，请问，我想判断里面有没有"world"这个元素，如果有，我就添加一个"javaee"元素，请写代码实现。

```
package com.sxt.fuxi;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;

public class Demo8 {
    public static void main(String[] args) {
//          我有一个集合，请问，我想判断里面有没有"java"这个元素，如果有，我就添加一个"javaee"元素，请写代码
```

实现。

```
        List list= new ArrayList();
        list.add("i");
        list.add("love");
        list.add("java");
        list.add("programing");

        Iterator it =list.iterator();   获取迭代器
        while (it.hasNext()){
            Object obj= it.next();
            String s = (String)obj;   想取 String  因为 it.next()返回了 Object  所以需要强制转换
            if(s.equals("java")){
                list.add("javaee");
            }
        }
    }
}
```

报异常:Exception in thread "main" java.util.ConcurrentModificationException 修改异常



解决:

```
package com.sxt.fuxi;

import java.util.*;



public class Demo8 {

    public static void main(String[] args) {

//          我有一个集合，请问，我想判断里面有没有"java"这个元素，如果有，我就添加一个"javaee"元素，请写代码实现。

        List list= new ArrayList();

        list.add("i");

        list.add("love");

        list.add("java");

        list.add("programing");
```

```
        ListIterator lit = list.listIterator();//这是 ArrayList 集合中扩展功能的迭代器
        while(lit.hasNext()){
            Object obj = lit.next();
            String s = (String)obj;
            if(s.equals("java")){
                lit.add("javaee");   //通过迭代器往 list 集合中添加元素
            }
        }
        System.out.println(list);
    }
}
```

迭代器遍历集合，修改集合

并发修改异常 ConcurrentModificationException 出现，list 刚开始告诉 Iterator 它的 size 是 6 个，在 Iterator 迭代的过程中加进一个，变成 7 个，此时迭代器不知道增加了，这时候就报异常

C:解决方案

a:迭代器迭代元素，迭代器修改元素(List 类中有个 ListIterator 的特有功能 add)

b:集合遍历元素，集合修改元素

●集合框架(ListIterator)(了解)

boolean hasNext()是否有下一个

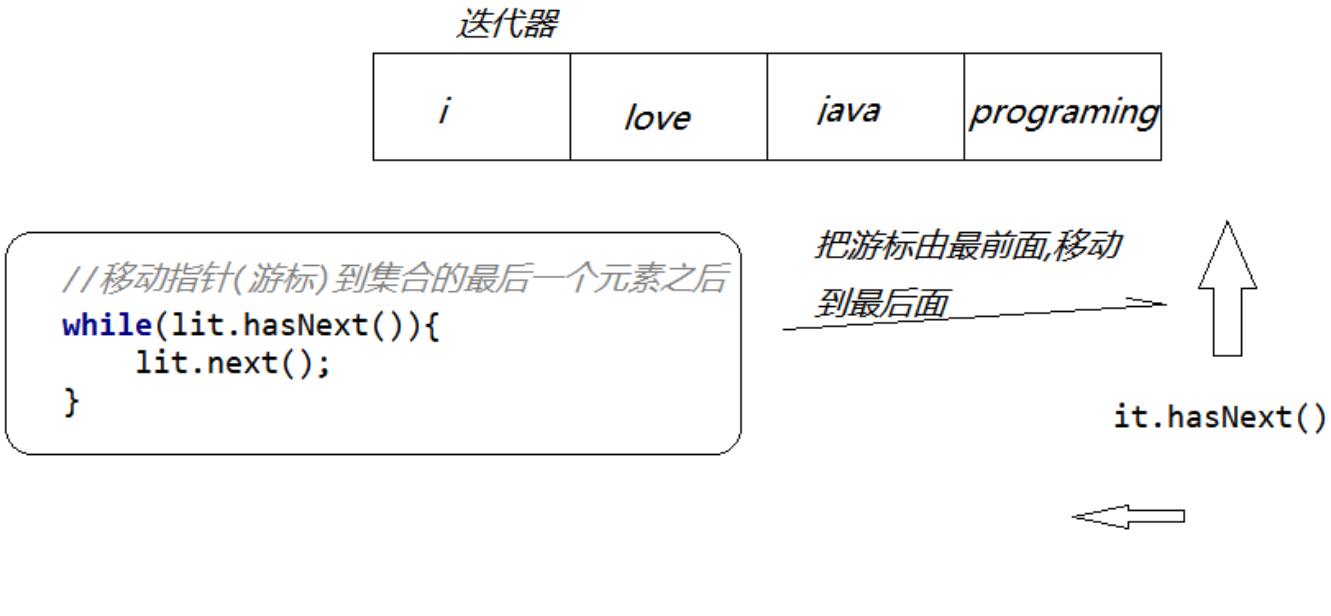
boolean hasPrevious()是否有前一个

Object next()返回下一个元素

Object previous();返回上一个元素

```
package com.sxt.fuxi;

import java.util.*;



public class Demo8 {

    public static void main(String[] args) {

//          我有一个集合，请问，我想判断里面有没有"java"这个元素，如果有，我就添加一个"javaee"元素，请写代码
实现。

        List list= new ArrayList();

        list.add("i");

        list.add("love");

        list.add("java");

        list.add("programing");

        ListIterator lit = list.listIterator();//这是 ArrayList 集合中扩展功能的迭代器

        //顺序迭代

    /*      while(lit.hasNext()){

            Object obj = lit.next();

            System.out.println(obj);

        }

        */

        //移动指针(游标)到集合的最后一个元素之后
```

```
        while(lit.hasNext()){
            lit.next();
        }

        //逆序迭代
        while(lit.hasPrevious()){   //判断是否有上一个元素
            Object obj = lit.previous();//获取上一个元素
            System.out.println(obj);
        }
    }
}
```



●集合框架(Vector 的特有功能)(了解)

A:Vector 类概述

B:Vector 类特有功能

public void addElement(E obj)　等同于 add(E obj)

public E elementAt(int index)　等同于 List 里的 get(int index)

public Enumeration elements()　//枚举

C:案例演示

Vector 的迭代

```
package com.sxt.fuxi;

import java.util.*;



public class Demo8 {

    public static void main(String[] args) {

        Vector v= new Vector();

        v.add(3);

        v.add(4);

        v.add(5);

        v.add(6);

        v.addElement(7);

        v.addElement(8);

        v.addElement(9);

        System.out.println(v);

//          System.out.println(v.elementAt(1));//获取指定索引位置上的元素
```

```
        /*
        Iterator it = v.iterator();
        while(it.hasNext()){
            Object obj = it.next();
            System.out.println(obj);
        }*/

        //迭代方式二:了解即可, Vector 特有的
        Enumeration en = v.elements();
        while(en.hasMoreElements()){
            Object obj = en.nextElement();
            System.out.println(obj);
        }
    }
}
```

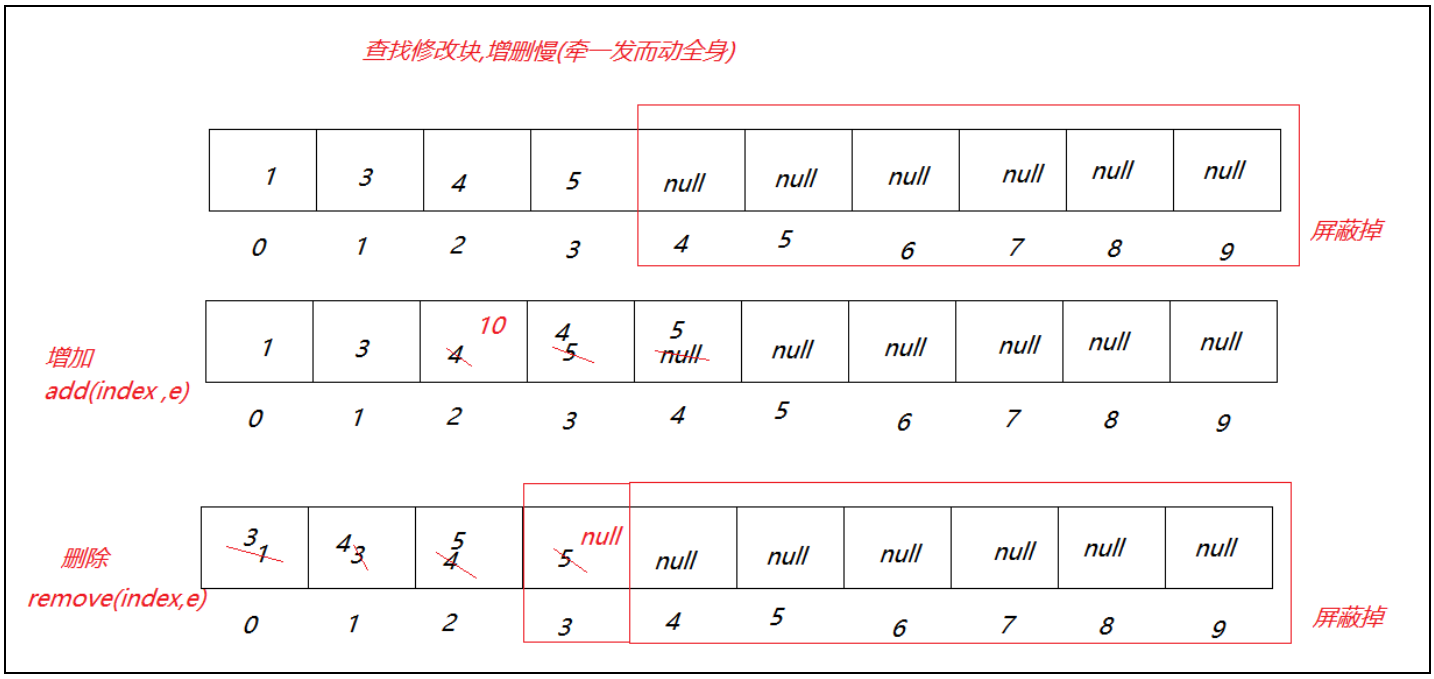
●集合框架(数据结构之数组和链表)

A:数组

查询快修改也快

增删慢

画图表示:

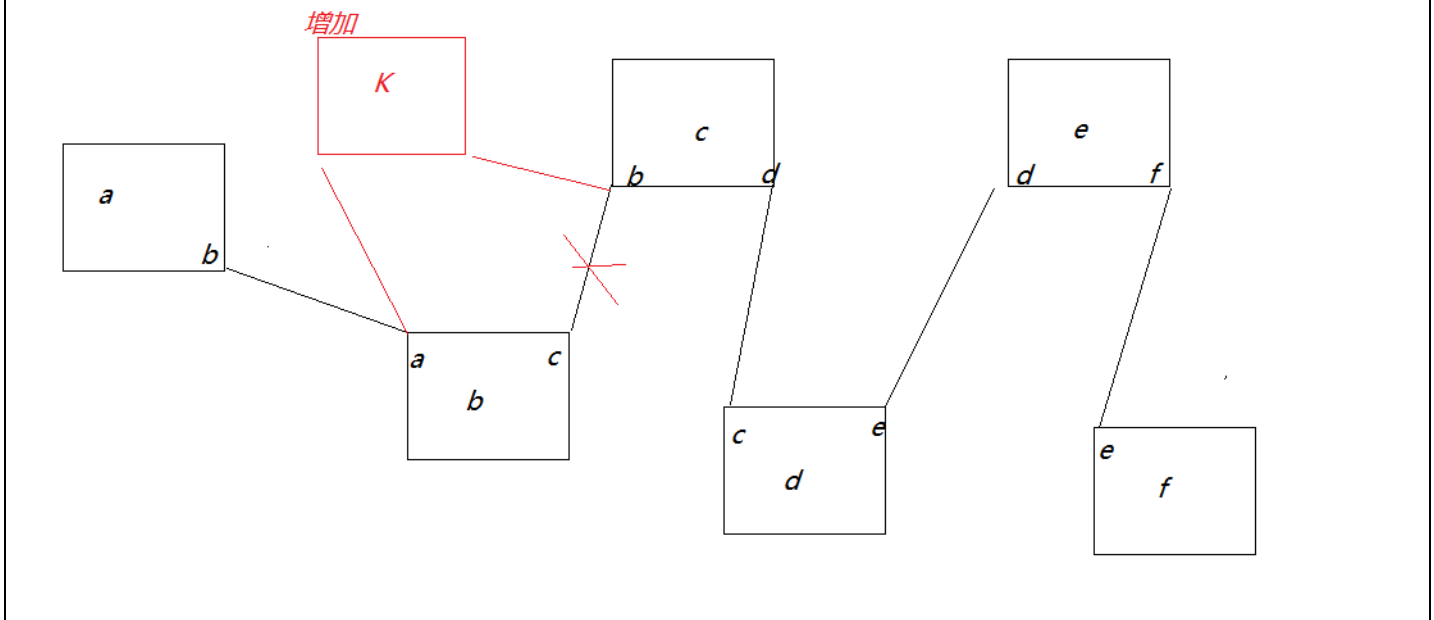


B:链表 LinkedList

查询慢,修改也慢

增删快

画图分析:



增删块,查改慢