1.集合是什么

java中提供的一种容器，可以存储多个数据

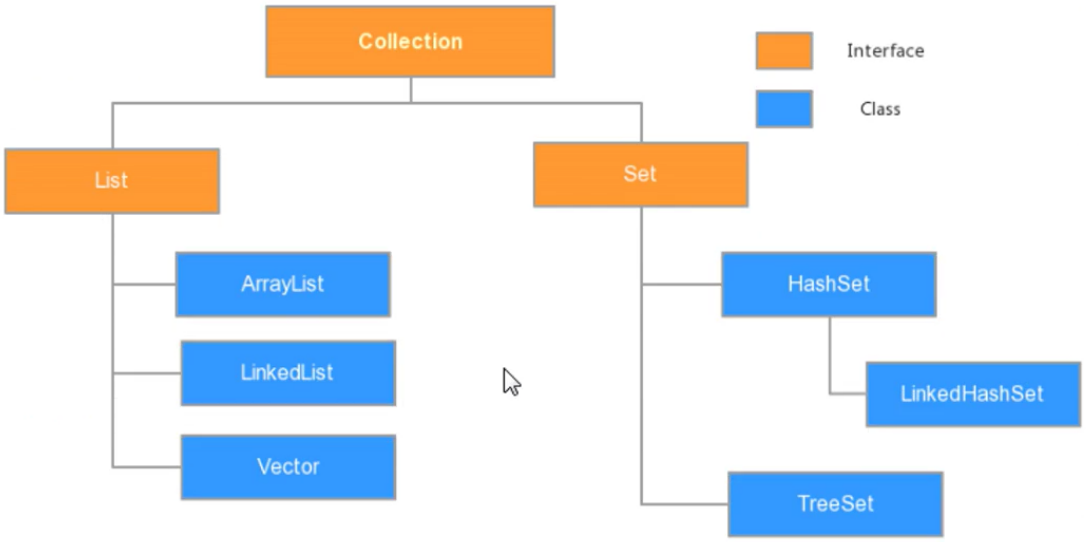
	长度	存储内容
集合	不固定	只能存储对象
数组	固定	基本类型/对象

2.集合框架

会使用集合存储数据集

会遍历集合，把数据取出来

掌握每种集合的特性



Collection接口：定义的是所有单列集合中共性的方法（没有带索引的方法）

List接口：

Set接口：

Vector集合:

ArrayList集合：

LinkedList集合：

HashSet集合：无序：底层哈希表 + 红黑树实现

LinkedHashSet集合：有序：哈希表 + 链表实现

TreeSet集合：无序（存取数据顺序可能不一致）：底层二叉树实现。一般用于排序

	区别
List	有序（存取元素顺序一直） 允许存储重复数据 有索引，可以使用普通For循环遍历
Set	不允许存储重复元素 没有索引，不能使用普通For循环遍历

Collection中的方法

```
boolean add(E e);
boolean remove(E e);
boolean clear(); //清空集合所有的元素，但是不删除集合
boolean contains(E e); //判断集合中是否包含某个元素
boolean isEmpty();
```

```
int size(); // 获取集合的长度
Object toArray(); //将集合转成一个数组，返回一个Object类型的数组
```

3.Iterator接口：迭代器，对集合进行遍历

Collection集合元素通用方法，有元素取出来，直到去完

Collection有一个方法iterator（），返回此元素上进行迭代的迭代器

```
boolean hasNext(); // 判断集合中还有没有下一个元素，有返回true 没有返回false
E next(); //返回集合的下一个元素
```

```
步骤：
1.使用集合中的iterator（）方法获取迭代器的实现类对象，用Iterator接口接受
2.使用Iterator接口中的方法hasNext判断还有没有下一个元素
3.使用Iterator接口中的方法next取出集合中的下一个元素
```

```
// 迭代器
Iterator<String> iterator = collection.iterator();
while(iterator.hasNext()){
 System.out.println(iterator.next());
}
```

实现原理：

获取迭代器的实现类对象，指向-1索引

4.增强for循环（jdk1.5）

底层使用的也是迭代器，使用for循环的格式，简化了迭代器的书写

```
格式：
for（集合/数组的数据类型 变量名 ： 集合名/数组名）{
 sout（变量名）；
}
```

5.泛型

泛型：是一种未知的数据类型，当我们不知道使用什么数据类型的时候，可以使用泛型

泛型也可以看成一个变量，用来接受数据类型

类型什么时候确定：创建集合对象的时候，就会确定泛型的数据类型，会把数据类型作为参数类型传递，复制给泛型E

	好处	弊端
使用泛型	避免了类型转换的麻烦 把运行期的异常，提升到编译器	泛型是什么类型，就只能存

不使用泛型	默认类型Object，可以存储任何类型数据	不安全，会引发异

定义和使用泛型

定义含有泛型的方法

```
格式：
修饰符 <泛型> 返回值类型 方法名（参数列表（使用泛型））{

}
public <T> void method(T t){
 Sytem.out.println(m);
}
```

含有泛型的接口

```
定义泛型接口
1.实现类指定泛型类型
2.实现类不指定泛型类型，接口使用什么泛型，实现类就使用什么泛型
```

泛型通配符

任意的数据类型

使用方式，不能创建对象使用，只能作为方法的参数使用

```
定义一个方法，能遍历所有类型的Arraylist集合
这时候我们不知道Arraylist集合使用什么类型，可以使用泛型通配符？来接受数据类型
```

泛型的上限限定

```
? extends E // 代表使用的泛型只能是E类型的子类/本身
```

泛型的下限限定

```
? super E // 代表使用的泛型只能是E类型的父类/本身
```