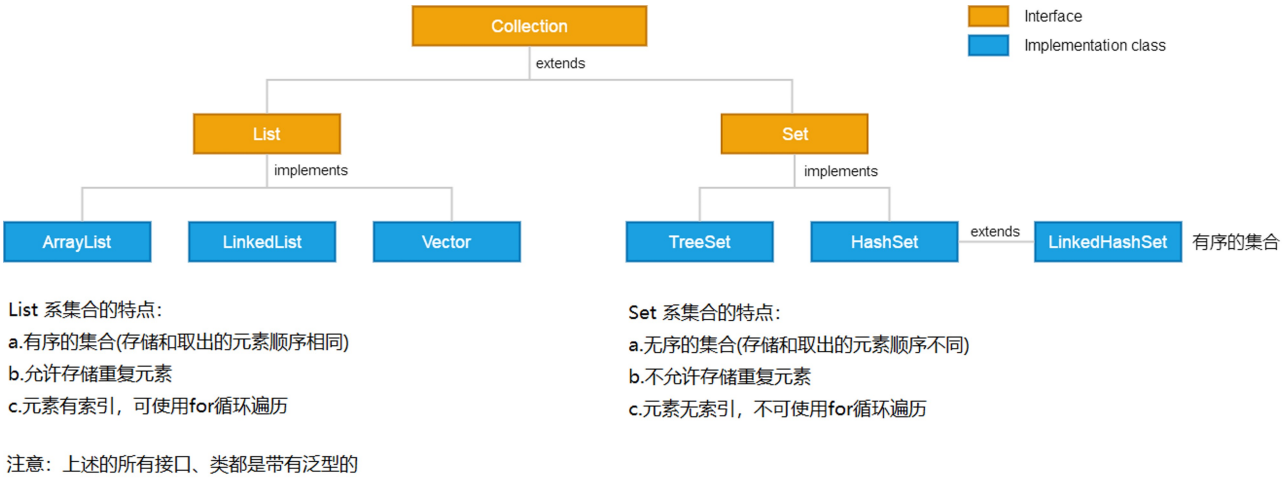
day02_Collection、泛型

一.Collection 集合

集合是java中提供的一种容器，可以用来存储多个数据。集合和数组都是存储数据的容器，它们的区别如下：
(1).数组的长度是固定的，而集合的长度是可变的。
(2).数组中存储的是同一类型的元素，可以存储基本类型数据、引用类型数据(对象数组)。集合存储的都是对象，而且对象的类型可以不一致。

1.集合框架介绍

集合按照其存储结构可以分为两大类，分别是单列集合 java.util.Collection 和双列集合 java.util.Map。Collection 是单列集合类的根接口，Collection 接口中定义着单列集合框架中最共性的内容，一般将实现了 Collection 接口的对象称为 Collection 集合。下图就描述了整个 Collection 集合的继承体系。



由于接口不能创建对象，所以只能通过其实现类创建集合对象来使用集合。

2.Collection 集合常用方法

二.Iterator 迭代器

1.Iterator 接口

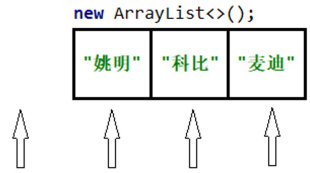
在程序开发中，经常需要遍历集合中的所有元素。针对这种需求，JDK专门提供了一个接口 java.util.Iterator，Iterator 接口也是Java集合中的一员，但它与 Collection、Map 接口有所不同，Collection、Map 接口主要用于存储元素，而 Iterator 接口主要是以"迭代"的方式遍历集合中的元素，因此 Iterator 接口的实现类对象也被称为集合的迭代器。

迭代：首先判断集合中有没有元素，如果有就把这个元素取出来，然后再继续判断，如果还有就再取出来，直到把集合中的元素取完为止。这种取出方式专业术语称为迭代。(Collection 集合元素的通用获取方式)

2.迭代器的实现原理

Iterator 迭代器对象在遍历集合时，内部采用指针的方式来跟踪集合中的元素。

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

    Collection<String> coll = new ArrayList<>();

    coll.add("姚明");
    coll.add("科比");
    coll.add("麦迪");

    Iterator<String> it = coll.iterator();  获取迭代器的实现类对象，指针位于第一个元素之前

    while (it.hasNext()) {
        String e = it.next();  调用next方法后，迭代器的索引会向后移动一位，指向下一个元素并将该元素返回
        System.out.println(e);
    }

}
```



3. 增强for循环

三.泛型

泛型就是一种未知的数据类型，当我们不知道使用什么数据类型，或者需要兼容多种数据类型时，就可以使用泛型。一般是在类、方法、接口中预支地使用这种未知的类型，泛型可以接收任何数据类型，并且会随着我们指定的数据类型而变化。(泛型只能是引用数据类型)

ArrayList 集合在定义时，不知道集合中会存储什么类型的数据，所以类型使用泛型 E

```java
public class ArrayList<E> {
    public boolean add(E e) {}
    public E get(int index) {}
}
```

new一个ArrayList 集合类对象时，可以指定数据类型，并传递给泛型 E
(这个对象就是根据 "已经发生变化的ArrayList类" new出来的)

```java
public class ArrayList<String> {
    public boolean add(String e) {}
    public String get(int index) {}
}
```

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

```java
public class ArrayList<Student> {
    public boolean add(Student e) {}
    public Student get(int index) {}
}
```

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

```java
public class ArrayList<Object> {
    public boolean add(Object e) {}
    public Object get(int index) {}
}
```

未指定数据类型时，泛型默认为Object类型

```java
ArrayList list = new ArrayList();
```

1.泛型的用法：在类、接口、方法中使用泛型

2.使用泛型的好处

3.泛型通配符

四.Collection 集合综合案例——斗地主案例