

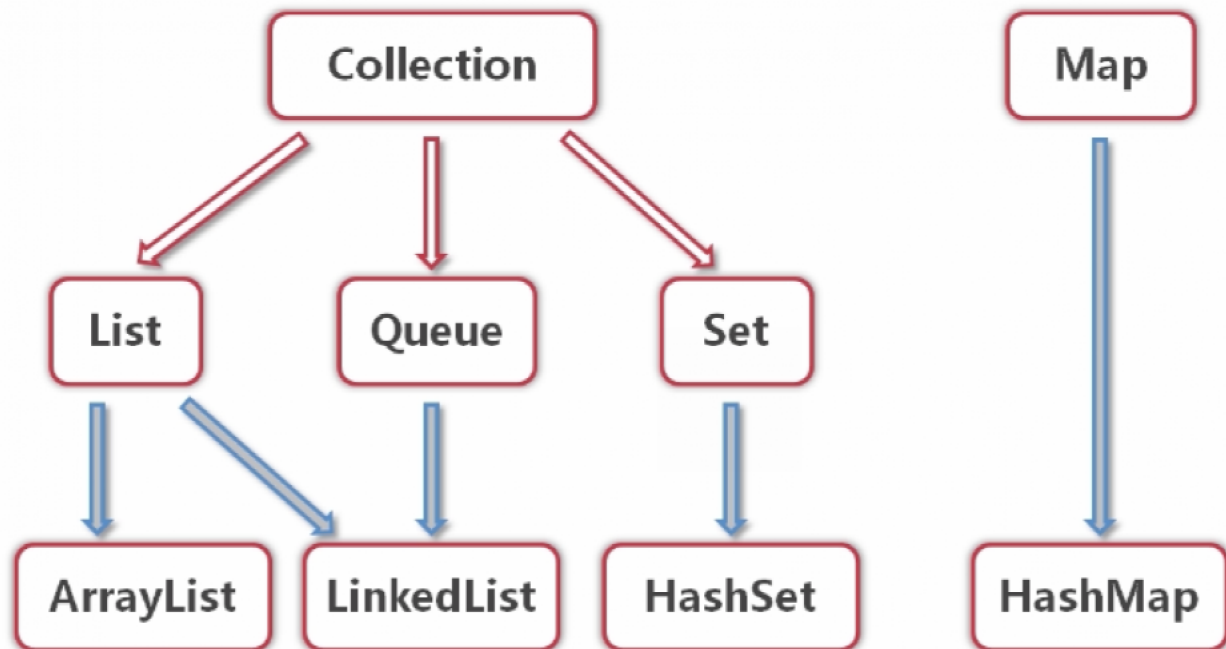
集合

- **疑问：为什么使用集合，而不用数组？**

## 应用场景

- 无法预测存储数据的数量
- 同时存储具有一对一关系的数据
- 需要进行数据的增删改查
- 数据重复问题

# 集合框架的体系结构



## List ( 列表 )

- List是元素有序并且可以重复的集合，称为序列
- List可以精确的控制每个元素的插入位置，或删除某个位置的元素
- List的两个主要实现类是ArrayList和LinkedList

# ArrayList

- **ArrayList底层是由数组实现的**
- **动态增长，以满足应用程序的需求**
- **在列表尾部插入或删除非常有效**
- **更适合查找和更新元素**
- **ArrayList中的元素可以为null**

- **案例：用ArrayList存储编程语言的名称，并输出。**
- **名称包括" Java"、" C"、" C++"、" Go" 和" Swift"**

# 案例：公告管理

- **需求**

- 公告的添加和显示
- 在指定位置处插入公告
- 删除公告
- 修改公告



# 案例：公告管理

- **公告类属性**
  - 编号 id
  - 标题 title
  - 创建人 creator
  - 创建时间 createTime
- **公告类方法**
  - 构造方法
  - 获取和设置属性值的方法

# Set

- **Set是元素无序并且不可以重复的集合，被称为集。**

# HashSet

- **HashSet是Set的一个重要实现类，称为哈希集**
- **HashSet中的元素无序并且不可以重复**
- **HashSet中只允许一个null元素**
- **具有良好的存取和查找性能**

## 案例

- 用HashSet存储多个表示颜色的英文单词，并输出。
- 单词包括:  
“blue”、“red”、“black”、“yellow”和“white”

# 案例：宠物猫信息管理

- **需求**
  - **添加和显示宠物猫信息**
  - **查找某只宠物猫的信息并输出**
  - **修改宠物猫的信息**
  - **删除宠物猫信息**

# 案例：宠物猫信息管理

- 属性

- 名字 name
- 年龄 month
- 品种 species

# 案例：宠物猫信息管理

- **方法**
  - **构造方法**
  - **获取和设置属性值的方法**
  - **其他方法**

# Iterator ( 迭代器 )

- **Iterator接口可以以统一的方式对各种集合元素进行遍历**
- **hasNext()方法检测集合中是否还有下一个元素**
- **next()方法返回集合中的下一个元素**



# Map

- Map中的数据是以键值对（key-value）的形式存储的
- key-value以Entry类型的对象实例存在
- 可以通过key值快速地查找value
- 一个映射不能包含重复的键
- 每个键最多只能映射到一个值

# HashMap

- 基于哈希表的Map接口的实现
- 允许使用null值和null键
- key值不允许重复
- HashMap中的Entry对象是无序排列的

# 案例1

- **完成一个类似字典的功能。**
  - **将单词以及单词的注释存储到HashMap中**
  - **显示HashMap中的内容**
  - **查找某个单词的注释并显示**

## 案例2：商品信息管理

- 使用HashMap对商品信息进行管理
  - 其中key为商品编号，value为商品对象
- 对HashMap中的商品信息进行增、删、改、查操作

# 分析商品信息类

- **属性**

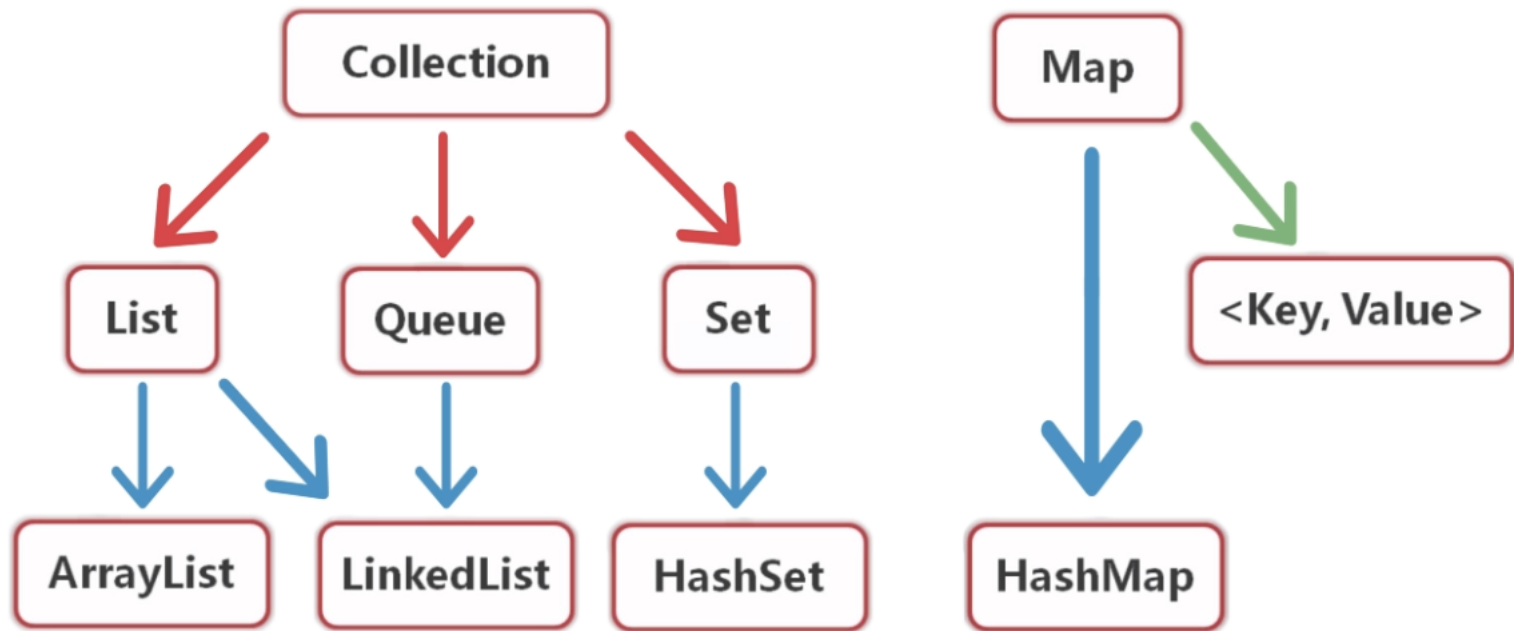
- **商品编号 : id**
- **商品名称 : name**
- **商品价格 : price**

- **方法**

- **构造方法**
- **获取和设置属性值的方法**
- **其他方法**

# List、Map、Set总结

# 总结



# ArrayList

- 底层由数组实现
- 元素有序且可以重复
- 可以动态增长，以满足应用程序的需求
- 元素值可以为null



# HashSet

- 元素无序并且不可以重复
- 只允许一个null元素

# HashMap

- 键不能重复
- 允许使用null值和null键
- HashMap中的Entry对象是无序排列的

# Iterator (迭代器)

- Iterator接口以统一的方式对各种集合元素进行遍历

```
Iterator<String> it=set.iterator();  
while(it.hasNext()){  
    System.out.print(it.next()+" ");  
}
```

# hashCode()

```
public int hashCode() {  
    final int prime = 31;  
    int result = 1;  
    result = prime * result + age;  
    result = prime * result + ((name == null) ? 0 : name.hashCode());  
    result = prime * result + ((species == null) ? 0 : species.hashCode());  
    return result;  
}
```

# equals()

```
public boolean equals(Object obj) {  
    if (this == obj)  
        return true;  
    if(obj.getClass()==Cat.class){  
        Cat cat=(Cat)obj;  
        return cat.getName().equals(name)&&(cat.getAge()==age)  
            &&cat.getSpecies().equals(species);  
    }  
  
    return false;  
}
```