HashSet

特点：

HashSet中的元素都是唯一的,不保证无序，不保证有序，

因为存储数据依靠HashCode，不能添加到指定位置，若存在，则添加失败.

非线性表

底层原理：

HashSet 底层用的是HashMap 当添加一个对象时，会将对象设为Key，然后Value定义的是一个无作用的对象(Object).

HashSet中常用方法：

boolean add(E e)

若不存在则添加(true)，若存在则失败(false).

void clear()

移除此集合中所有元素.

Object clone()

返回此 HashSet实例的浅层副本：元素本身不被克隆。

克隆的只是Set、Map对象，并未克隆其中的Set存储的元素

所以当一方Set对存储元素做出修改，则双方都会改变.

boolean contains(Object o)

如果此集合包含指定的元素，则返回true

Iterator<E> iterator()

返回此集合中元素的迭代器， 元素没有特定的顺序返回

boolean remove(Object o)

删除此集合包含的指定元素.

1.8新增：

spliterator()

将本集合分割成若干个子迭代器

```
public static void main(String[] args) {
   HashSet hashSet = new HashSet();
   for (int i = 10; i >0 ; i--) {
      hashSet.add(i);
   }
   for (int i = 1; i <1000 ; i++) {
      hashSet.add(i);
   }
   Spliterator spliterator =  hashSet.spliterator().trySplit();
```

```
    System.out.println("======================");
    spliterator.trySplit().forEachRemaining(T->{
        System.out.println(T);
    });
    System.out.println("======================");
    spliterator.trySplit().forEachRemaining(T->{
        System.out.println(T);
    });
    System.out.println("======================");
    spliterator.forEachRemaining(T->{
        System.out.println(T);
    });
}
```

必须先将集合分割一次，然后再利用获取到的对象继续对该spliterator 分割（trySplit）

迭代方式三种：

调用迭代器iterator

依靠增强for

```
for(String value: HashSet){
    System.out.println(value);
}
```

调用forEach

```
hashSet.forEach(T->{
    System.out.println(T);
})
```