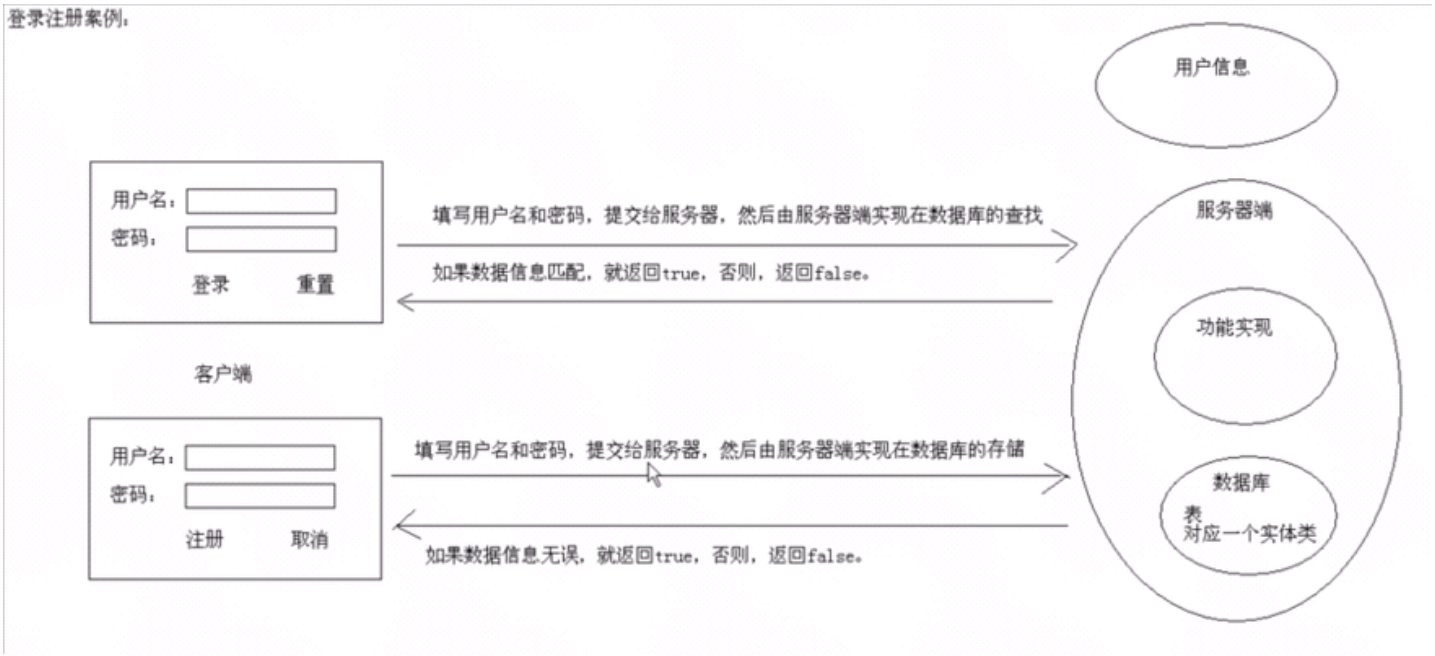
登陆注册图解

2018年7月1日 14:58



用户登陆注册案例

2018年7月16日 15:49

类：用户类、测试类

成员：

用户类：如果用户类的内容较多，维护起来麻烦，把用户类又划分为两类

用户基本描述类

用户操作类

测试类：main

在测试类中创建用户类对象，并使用其功能

分包：按功能分包

用户基本描述类包：cn.itcast.pojo

用户操作接口：cn.itcast.dao

用户操作类包：cn.itcast.dao.impl

用户测试类：cn.itcast.test

代码

2018年7月16日 22:11

```
package cn.itcast.dao;

import cn.itcast.pojo.User;

/**
 * 这是针对用户进行操作的接口
 *
 * @author TJtulong
 * @version V1.0
 */
public interface UserDao {
	/**
	 * 用户登陆功能
	 *
	 * @param username
	 * @param password
	 * @return
	 */
	public abstract boolean isLogin(String username, String password);

	/**
	 * 用户注册功能
	 *
	 * @param user
	 */
	public abstract void regist(User user);
}
```

```
package cn.itcast.dao.impl;

import java.util.ArrayList;

import cn.itcast.dao.UserDao;
import cn.itcast.pojo.User;

/**
 * 这是用户操作的具体实现类(集合版)
 *
 * @author TJtulong
 * @version V1.0
 *
 */
public class UserDaoImpl implements UserDao {
	//为了让多个成员变量公用一个变量,用static修饰
	private static ArrayList<User> array = new ArrayList<User>();

	@Override
	public boolean isLogin(String username, String password) {
		// 遍历集合,获取每一个用户,判断用户名和密码是否匹配
		boolean flag = false;
		for (User u : array) {
			if (u.getName().equals(username) &&
			u.getPassword().equals(password)) {
				flag = true;
				break;
			}
		}
		return flag;
	}

	@Override
	public void regist(User user) {
		// 把用户信息存入集合
		array.add(user);
	}

}
```

代码

2018年7月16日 22:12

```
package cn.itcast.pojo;

/**
 * 这是用户基本描述类
 *
 * @author TJtulong
 * @version V1.0
 */
public class User {
	// 用户名
	private String Name;
	// 密码
	private String Password;

	public User() {
		super();
		// TODO Auto-generated constructor stub
	}

	public String getName() {
		return Name;
	}

	public void setName(String name) {
		Name = name;
	}

	public String getPassword() {
		return Password;
	}

	public void setPassword(String password) {
		Password = password;
	}

}
```

```
package cn.itcast.game;

import java.net.StandardSocketOptions;
import java.util.Scanner;

/**
 * 这是猜数字小游戏
 *
 * @author TJtulong
 * @version V1.0
 */
public class GuessGame {
	private GuessGame() {
	}

	public static void start() {
		int number = (int) (Math.random() * 100) + 1;
		while (true) {
			Scanner sc = new Scanner(System.in);
			System.out.println("请输入1-100的数据：");
			int guessnumber = sc.nextInt();

			if (guessnumber > number) {
				System.out.println("你猜得大了");
			} else if (guessnumber < number) {
				System.out.println("你猜得小了");
			} else {
				System.out.println("你猜中了");
				break;
			}
		}
	}
}
```

```
package cn.itcast.test;

import java.util.Scanner;

import cn.itcast.dao.UserDao;
import cn.itcast.dao.impl.UserDaoImpl;
import cn.itcast.game.GuessGame;
import cn.itcast.pojo.User;

/**
 * 用户测试类
 *
 * @author TJtulong
 * @version V1.0
 *
 */
public class UserTest {
	public static void main(String[] args) {
		while (true) {
			System.out.println("----------欢迎光临----------");
			System.out.println("1 登陆");
			System.out.println("2 注册");
			System.out.println("3 退出");
			System.out.
```

```
println
("请输
入你的
选
择：");

Scann
er sc =
new
Scann
er(Syst
em.in);
String
choice
String
=
sc.next
Line();
// 调用
注册功
能
UserD
ao ud
= new
UserD
aoImp
l();
switch
(choic
eStrin
g) {
case
"1":
    Syste
    m.out.
    println
    ("-----
    --------
    --欢迎
    登
    录-----
    --------
    --");
    Syste
    m.out.
    println
    ("请输
    入用户
    名");
    String
    userna
    me =
    sc.next
    Line();
    Syste
    m.out.
    println
    ("请输
    入密
    码");
    String
    passw
    ord =
    sc.next
```

```
Line();

boole
an flag
=
ud.isL
ogin(u
serna
me,
passw
ord);
if
(flag) {
    Syste
    m.out.
    println
    ("登录
    成功");
    Guess
    Game.
    start();
    Syste
    m.exit(
    0);
} else {
    Syste
    m.out.
    println
    ("用户
    名或密
    码有误,
    登陆失
    败");
}

    break;
case
"2":
    Syste
    m.out.
    println
    ("-----
    -------
    --注册
    界
    面-----
    -------
    --");
    Syste
    m.out.
    println
    ("请输
    入用户
    名");
    String
    newUs
    ernam
    e =
    sc.next
    Line();
    Syste
    m.out.
    println
    ("请输
```

```
                    入密码");
                    String newPassword = sc.nextLine();

                    User user = new User();
                    user.setName(newUsername);
                    user.setPassword(newPassword);
                    ud.regist(user);
                    break;
                case "3":
                default:
                    System.out.println("欢迎使用");
                    break;
                }
            }
        }
    }
```

Set类

2018年7月16日 22:15

不包含重复元素的collection

```
package cn.itcast_01;

import java.util.HashSet;
import java.util.Set;

/*
 * Set:无序（存储顺序与取出顺序不唯一），唯一
 * HastSet:它不保证set的迭代顺序；特别是它不保证该顺序恒久不变
 *                          有自己的存储顺序
 */
public class SetDemo {
        public static void main(String[] args) {
                Set<String> set = new HashSet<String>();

                set.add("hello");
                set.add("world");
                set.add("java");
                set.add("world");// 相同的不进去
                set.add("java");// 相同的不进去

                for (String s : set) {
                        System.out.println(s);
                }
        }
}
```

Add()源码

2018年7月16日 22:54

通过查看add方法源码，这个方法的底层依赖两个方法：hashCode()和equals()

首先比较哈希值：

如果相同：比较地址值或者走equals()

如果不同：就直接添加到集合中

哈希表：是一个元素为链表的数组。**综合了数组和链表的好处**

HashSet自定义对象

2018年7月17日 0:15

```
import java.util.HashSet;

/*
 * 需求：储存自定义对象并遍历
 * 需求：如果对象的成员变量值都相同，则为同一个元素
 */
public class SetDemo2 {
	public static void main(String[] args) {
		HashSet<Student> hs = new HashSet<Student>();

		Student s1=new Student("王若潇",22);
		Student s2=new Student("梅西",31);
		Student s3=new Student("莫德里奇",33);
		Student s4=new Student("莫德里奇",33);
		Student s5=new Student("莫德里奇",35);

		hs.add(s1);
		hs.add(s2);
		hs.add(s3);
		hs.add(s4);
		hs.add(s5);

		for (Student s:hs) {
			System.out.println(s.getName()+"----"+s.getAge());
		}
	}
}
```

重写hashcode

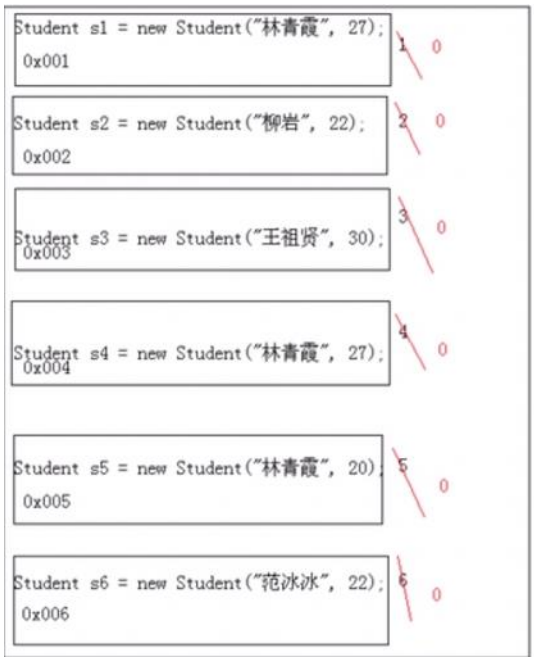
重写equals

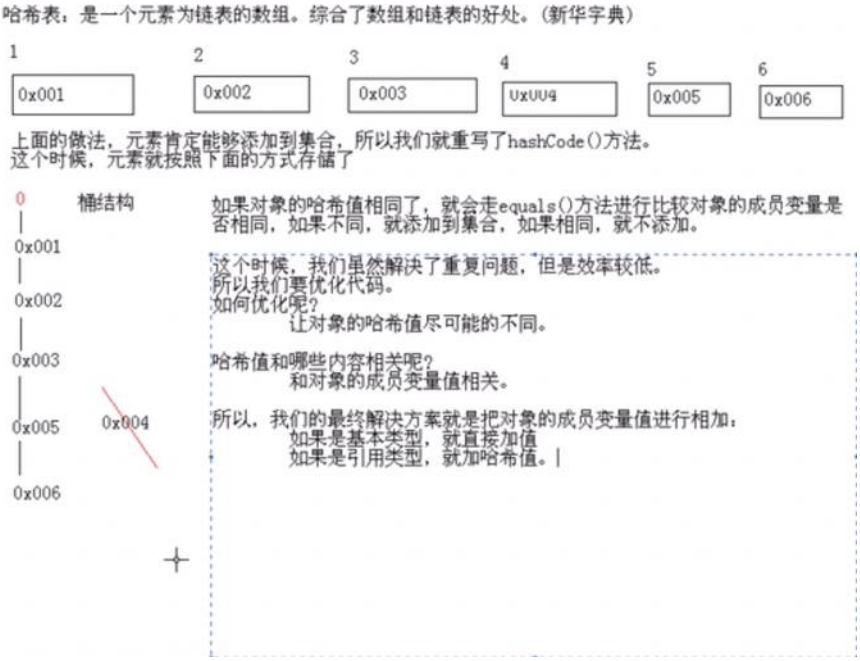
```
@Override
public int hashCode() {
	// return 0;
	// 因为成员变量值影响了哈希值，因此把成员变量值相加即可
	// return this.Name.hashCode()+this.age;
	// 为了尽可能地区分，把他们乘以整数
	return this.Name.hashCode() + this.age * 15;
}

@Override
public boolean equals(Object obj) {
	if (this == obj)
		return true;
	if (obj == null)
		return false;
	if (getClass() != obj.getClass())
		return false;
	Student other = (Student) obj;
	if (Name == null) {
		if (other.Name != null)
			return false;
	} else if (!Name.equals(other.Name))
		return false;
	if (age != other.age)
		return false;
	return true;
}

}
```

自动生成即可

图解

2018年7月17日 14:14





LinkedHashSet

2018年7月17日 14:13

哈希表+链表

```
import java.util.LinkedHashSet;

/*
 * LinkedHashList:底层数据结构由哈希表和链表组成
 * 哈希表保证元素的一致性
 * 链表保证元素有序（储存和取出一致）
 */
public class LinkedHashListDemo {
	public static void main(String[] args) {
		LinkedHashSet<String> hs = new LinkedHashSet<String>();

		hs.add("hello");
		hs.add("world");
		hs.add("java");

		for (String s : hs) {
			System.out.println(s);// 输出结果有序
		}
	}
}
```

TreeSet

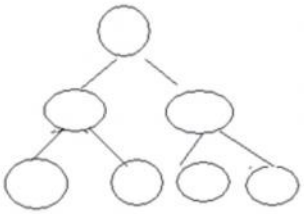
2018年7月17日 14:21

```
import java.util.TreeSet;

/*
 * TreeSet:能够对元素按照某种规则进行排序
 * 排序有两种方式：
 *                  1.自然排序（元素具备比较器）
 *                  2.比较器排序（集合具备比较器）
 * TreeSet集合的特点：排序和唯一
 *
 */
public class TreeSetDemo {
        public static void main(String[] args) {
                TreeSet<Integer> ts=new TreeSet<Integer>();

                ts.add(20);
                ts.add(18);
                ts.add(23);
                ts.add(17);
                ts.add(20);
                ts.add(19);

                for(Integer i:ts) {
                        System.out.println(i);//由小到大输出（自然排序）
                }

        }
}
```

图解二叉树

2018年7月17日 15:18

TreeSet：底层是二叉树结构。(红黑树是一种自平衡的二叉树)



```
TreeSet<Integer> ts = new TreeSet<Integer>();

// 创建元素并添加
// 20,18,23,22,17,24,19,18,24
ts.add(20);
ts.add(18);
ts.add(23);
ts.add(22);
ts.add(17);
ts.add(24);
ts.add(19);
ts.add(18);
ts.add(24);

// 遍历
for (Integer i : ts) {
        System.out.println(i);
}
```

元素是如何存储进去的呢?

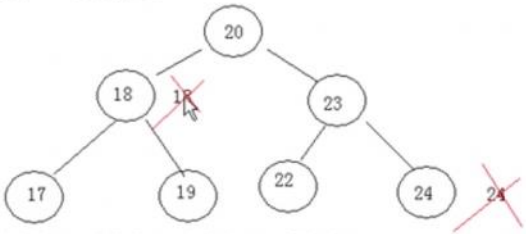
第一个元素存储的时候，直接作为根节点存储。

从第二个元素开始，每个元素从根节点开始比较

大　　就作为右孩子

小　　就作为左孩子

相等　　就不搭理它



元素是如何取出来的呢?(前序遍历，中序遍历，后序遍历)

从根节点开始，按照左，中，右的原则依次取出元素即可。

TreeSet存储自定义对象

2018年7月17日 15:19

```
package cn.itcast_04;

/*
 * 如果一个类的元素想要能够自然排序，就要实现自然排序接口
 */
public class Student implements Comparable<Student> {
	private String name;
	private int age;

	public Student() {
		super();
		// TODO Auto-generated constructor stub
	}

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

	@Override
	public String toString() {
		return "Student [Name=" + name + ", age=" + age + "]";
	}

	@Override
	public int compareTo(Student s) {
		// return 0;
		// 按照年龄排序
		int num = this.age - s.age;
		// 次要条件
		int num2 = num == 0 ? this.name.compareTo(s.name) : num;
		return num2;
	}
}
```

```
package cn.itcast_04;

import java.util.TreeSet;

/*
 * TreeSet储存自定义对象并排序
 * 1.按照什么顺序排序：按照年龄从小到大
 * 2.什么算同一个元素：姓名年龄相同
 */
public class TreeSetDemo2 {
	public static void main(String[] args) {
		TreeSet<Student> ts = new TreeSet<Student>();

		Student s1 = new Student("wangruoxiao", 22);
		Student s2 = new Student("wuqilong", 40);
		Student s3 = new Student("linqingxia", 27);
		Student s4 = new Student("zhurui", 30);
		Student s5 = new Student("linqingxia", 27);
		Student s6 = new Student("wangruoxia", 22);

		ts.add(s1);
		ts.add(s2);
		ts.add(s3);
		ts.add(s4);
		ts.add(s5);
		ts.add(s6);

		for (Student s : ts) {
			System.out.println(s.getName() + "---" + s.getAge());
		}
	}
}
```

比较器排序

2018年7月17日 15:42

```
import java.util.Comparator;
import java.util.TreeSet;

/*
 * TreeSet储存自定义对象并排序
 * 1.按照什么顺序排序：按照姓名长度
 * 2.什么算同一个元素：姓名年龄相同
 */
public class TreeSetDemo2 {
	public static void main(String[] args) {
		//TreeSet<Student> ts = new TreeSet<Student>();默认为自然排序

		//比较器排序
		TreeSet<Student> ts = new TreeSet<Student>(new MyComparator());

		Student s1 = new Student("wangruoxiao", 22);
		Student s2 = new Student("wuqilong", 40);
		Student s3 = new Student("linqingxia", 27);
		Student s4 = new Student("zhurui", 30);
		Student s5 = new Student("linqingxia", 27);
		Student s6 = new Student("wangruoxia", 22);

		ts.add(s1);
		ts.add(s2);
		ts.add(s3);
		ts.add(s4);
		ts.add(s5);
		ts.add(s6);

		for (Student s : ts) {
			System.out.println(s.getName() + "---" + s.getAge());
		}
	}
}
```

```
import java.util.Comparator;

public class MyComparator implements
Comparator<Student> {

	@Override
	public int compare(Student s1, Student s2) {
		//return 0;

		int num=s1.getName().length()-
		s2.getName().length();
		int num2=num==0?
		s1.getName().compareTo(s2.getName()):num;
		int num3=num2==0?s1.getAge()-
		s2.getAge():num2;

		return num3;
	}

}
```

比较器排序改进

2018年7月17日 16:07

```
import java.util.Comparator;
import java.util.TreeSet;

/*
 * TreeSet储存自定义对象并排序
 * 1.按照什么顺序排序：按照姓名长度
 * 2.什么算同一个元素：姓名年龄相同
 */
public class TreeSetDemo2 {
	public static void main(String[] args) {
		// TreeSet<Student> ts = new TreeSet<Student>();默认为自然排序

		// 比较器排序
		// TreeSet<Student> ts = new TreeSet<Student>(new MyComparator());

		// 用匿名内部类实现
		TreeSet<Student> ts = new TreeSet<Student>(new Comparator<Student>
		() {
			public int compare(Student s1, Student s2) {
				int num = s1.getName().length() - s2.getName().length();
				int num2 = num == 0 ?
				s1.getName().compareTo(s2.getName()) : num;
				int num3 = num2 == 0 ? s1.getAge() - s2.getAge() : num2;

				return num3;
			}
		});

		Student s1 = new Student("wangruoxiao", 22);
		Student s2 = new Student("wuqilong", 40);
		Student s3 = new Student("linqingxia", 27);
		Student s4 = new Student("zhurui", 30);
		Student s5 = new Student("linqingxia", 27);
		Student s6 = new Student("wangruoxia", 22);

		ts.add(s1);
```

```
            ts.add(s2);
            ts.add(s3);
            ts.add(s4);
            ts.add(s5);
            ts.add(s6);

            for (Student s : ts) {
                System.out.println(s.getName() + "---" + s.getAge());
            }
        }
}
```

练习

2018年7月17日 16:16

```
import java.util.HashSet;
import java.util.Random;

/*
 * 获取10个1-10的随机数，要求随机数不能重复
 */
public class HashSetDemo {
	public static void main(String[] args) {
		Random r= new Random();
		HashSet<Integer> ts = new HashSet<Integer>();
		while(ts.size()<10) {
			int num = r.nextInt(20)+1;
			ts.add(num);
		}

		for(Integer i:ts) {
			System.out.println(i);
		}
```