【案例5-3】 模拟用户注册

### 【案例介绍】

**1.任务描述**

互联网为我们提供了巨大的便利，如微信带给我们的视频、淘宝带给我们便利的购物等，但这些APP都需要有一个账户才可以登录，而账户需要注册可以获取。

本例要求编写一个程序，模拟用户注册。用户输入用户名、密码、确认密码、生日（格式为yyyy--mm—dd为正确）、手机号（手机号长度为11位，并且以13、15、17、18为开头的手机号为正确）、邮箱（包含符号“@”为正确）信息之后，判断信息正确后，验证用户是否重复，重复则给出相应提示，如果不重复则注册成功。案例要求使用HashSet集合实现。

**2.运行结果**

任务运行结果如图6-1所示：

手机屏幕截图

描述已自动生成

图6-1 运行结果

### 【案例任务】

* 学会分析“模拟微信用户注册”任务的实现思路。
* 根据思路独立完成“模拟微信用户注册”任务的源代码编写、编译及运行。
* 掌握HashSet集合常用方法的使用及存储数据的流程。

### 【案例思路】

（1） 为了便于存储用户的信息。需要创建一个用户类，在类中重写其中的HashCode()方法，令其返回用户的额哈希值，再重写equals()方法，来比较对象的用户属性是否相等。

（2） 创建一个用户注册类来模拟用户注册信息，该类中可以用HashSet集合创建一个数据列表，然后向列表中添加两条初始用户信息。

（3） 从控制台获取用户填写信息，通过Scanner类的nextline()方法实现，获取后，需要将获取的数据进行校验。

（4） 单独创建一个校验类，在该类中实现校验用户输入信息的方法。校验结束后，如果效验结果错误就直接返回错误信息，这里可以分别声明一个校验结果的变量和一个校验状态的变量。校验结果变量用于存储提示信息，校验状态变量用于存储结果的判断标识。

（5） 当用户输入的信息不满足规定的格式时，需要修改变量的状态并且存储错误信息。

（6） 判断校验状态，如果所有信息都通过校验，则将用户信息创建为用户对象，通过将对象添加到用户列表返回结果来判断用户是否重复，并记录下返回结果信息。

### 【案例实现】

（1）创建用户类，并重写其HashCode()和equals()方法，其代码具体如文件6-3-1所示。

文件6-1 User.java

1. package com.itheima.shiyan6\_3;
2. import java.util.Date;
3. //用户信息
4. public class User {
5. private String userName; // 用户名
6. private String password; // 密码
7. private Date birthday; // 生日
8. private String telNumber; // 手机号码
9. private String email; // 邮箱
10. public User() {
11. }
12. public User(String userName, String password, Date birthday,
13. String telNumber, String email) {
14. this.userName = userName;
15. this.password = password;
16. this.birthday = birthday;
17. this.telNumber = telNumber;
18. this.email = email;
19. }
20. // 重写hashCode与equals方法
21. @Override
22. public int hashCode() {// 重写hashCode方法，以用户名作为是否重复的依据
23. return userName.hashCode();
24. }
25. @Override
26. public boolean equals(Object obj) {
27. if (this == obj) {// 判断是否是同一个对象
28. return true;// 如果是同一个对象，直接返回true
29. }
30. if (obj == null) {// 判断这个对象是否为空
31. return false;// 如果对象是空的，直接返回false
32. }
33. if (getClass() != obj.getClass()) {// 判断这个对象是否是User类型
34. return false;// 如果不是，直接返回false
35. }
36. User other = (User) obj;// 将对象强转为User类型
37. if (userName == null) {// 判断集合中用户名是否为空
38. if (other.userName != null) {// 判断对象中的用户名是否为空
39. // 如果集合中用户名为空并且对象中用户名不为空，则返回false
40. return false;
41. }
42. // 判断用户名是否相同
43. } else if (!userName.equals(other.userName)) {
44. return false;// 如果不同，返回false
45. }
46. return true;
47. }
48. }

在文件6-1中创建了一个用户类，在代码22~24行代码中重写了HashCode()方法，使其返回userName属性的哈希值，并且在代码26~47行重写了equals()方法用于比较对象userName的属性是否相等，并返回结果。

（2）创建用户注册类，模拟注册信息，其代码如文件6-2所示。

文件6-2 UserRegister.java

1. package com.itheima.shiyan6\_3;
2. import java.util.Date;
3. import java.util.HashSet;
4. import java.util.Scanner;
5. public class UserRegister {
6. public static HashSet<User> USER\_DATA = new HashSet<User>(); // 用户数据
7. public static void main(String[] args) {
8. initData();// 初始化人员信息
9. Scanner scan = new Scanner(System.in);
10. System.out.print("请输入用户名：");
11. String userName = scan.nextLine();// 获取用户名
12. System.out.print("请输入密码：");
13. String password = scan.nextLine();// 获取密码
14. System.out.print("请重复密码：");
15. String repassword = scan.nextLine();// 获取重复密码
16. System.out.print("出生日期：");
17. String birthday = scan.nextLine();// 获取出生日期
18. System.out.print("手机号码：");
19. String telNumber = scan.nextLine();// 获取手机号码
20. System.out.print("电子邮箱：");
21. String email = scan.nextLine();// 获取电子邮箱
22. // 校验用户信息,返回登录状态信息
23. CheckInfo checkInfo = new CheckInfo(USER\_DATA);
24. String result = checkInfo.checkAction(userName, password,
25. repassword, birthday, telNumber, email);
26. System.out.println("注册结果:" + result);
27. }
28. // 初始化数据，创建两个已存在的用户信息
29. private static void initData() {
30. User user = new User("迪丽热巴", "zz,123", new Date(),
31. "18810319240", "zhangzheng@itcast.cn");
32. User user2 = new User("吴宣仪", "zq,123", new Date(),
33. "18618121193", "zhouqi@itcast.cn");
34. USER\_DATA.add(user);
35. USER\_DATA.add(user2);
36. }
37. }

在文件6-2中，程序首先会执行第29~36行代码的initData()方法，创建“迪丽热巴”和“吴宣仪”两位用户，并将其放入到用户数据列表中，然后执行9~25行代码，获取输入的信息并将其传入CheckInfo类的checkAction()方法对输入信息进行校验。最后将结果输出。

（3）创建校验信息类，其代码如文件6-3所示。

文件6-3 CheckInfo.java

1. package com.itheima.shiyan6\_3;
2. import java.text.DateFormat;
3. import java.text.ParseException;
4. import java.text.SimpleDateFormat;
5. import java.util.Date;
6. import java.util.HashSet;
7. public class CheckInfo {
8. public static HashSet<User> USER\_DATA = new HashSet<User>(); // 用户数据
9. public CheckInfo(HashSet<User> USER\_DATA) {
10. this.USER\_DATA = USER\_DATA;
11. }
12. // 校验用户信息,返回登录状态信息
13. public String checkAction(String userName, String password,
14. String rePassword,String birthday, String phone, String email) {
15. StringBuilder result = new StringBuilder();
16. // 1代表成功 2代表失败
17. int state = 1;
18. // 密码判断
19. if (!password.equals(rePassword)) {// 判断密码和重复密码是否相同
20. result.append("两次输入密码不一致!\r\n");
21. state = 2;
22. }
23. // 生日判断
24. if (birthday.length() != 10) {// 字符串长度不为10，则认为格式错误
25. result.append("生日格式不正确!\r\n");
26. state = 2;
27. } else {
28. for (int i = 0; i < birthday.length(); i++) {
29. Character thisChar = birthday.charAt(i);
30. if (i == 4 || i == 7) {
31. if (!(thisChar == '-')) {// 验证第4位和第7位是否是
32. 符号“-”
33. result.append("生日格式不正确!\r\n");
34. state = 2;
35. }
36. } else {// 验证除了第4位和第7位的字符是否是0~9的数字
37. if (!(Character.isDigit(thisChar))) {
38. result.append("生日格式不正确!\r\n");
39. state = 2;
40. }
41. }
42. }
43. }
44. // 手机号判断
45. // 判断手机号长度不等于11位则认为此手机号无效
46. if (phone.length() != 11) {
47. result.append("手机号码不正确!\r\n");
48. state = 2;
49. // 默认有效手机号为13、15、17和18开头的手机号
50. } else if (!(phone.startsWith("13") || phone.startsWith("15")
51. || phone.startsWith("17") || phone.startsWith("18"))){
52. result.append("手机号码不正确!\r\n");
53. state = 2;
54. }
55. // 邮箱判断
56. // 判断邮箱地址，默认不带@符号的邮箱为无效邮箱
57. if (!email.contains("@")) {
58. result.append("邮箱不正确!\r\n");
59. state = 2;
60. }
61. // 如果以上信息校验无误，则将新用户加入到集合
62. if (state == 1) {
63. // 格式化日期返回Date对象
64. //定义日期格式
65. DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
66. Date dateBirthday = null;
67. try {
68. // 将生日格式化成日期格式
69. dateBirthday = format.parse(birthday);
70. } catch (ParseException e) {
71. e.printStackTrace();
72. }
73. User newUser = new User(userName, rePassword,
74. dateBirthday, phone, email);
75. // 将用户添加到列表中，同时可根据HashSet判断出用户名有没有重复
76. if (!USER\_DATA.add(newUser)) {
77. result.append("用户重复!");
78. state = 2;
79. }
80. if (state == 1) {
81. result.append("注册成功!");
82. }
83. }
84. return result.toString();
85. }
86. }

在文件6-3中，当主程序调用该类中的checkAction()方法后，第19~60行代码会校验用户填写的信息，所有校验通过后，将信息创建成User对象，通过第74~75行的代码，判断用户名的哈希值是否相等。相等则添加失败，反之则成功，返回结果。