【案例8-3】 模拟银行存取钱功能

### 【案例介绍】

**1.任务描述**

在银行办理业务时，通常银行会开多个窗口，客户排队等候，窗口办理完业务，会呼叫下一个用户办理业务。本案例要求编写一个程序模拟银行存取钱业务办理。假如有两个用户在存取钱，两个用户分别操作各自的账户，并在控制台打印存取钱的数量以及账户的余额。

**2.运行结果**

运行结果如图8-1所示。

电脑屏幕的照片

描述已自动生成

图8-1 运行结果

### 【案例目标】

* 学会分析”模拟银行存取钱功能”任务实现的逻辑思路。
* 能够独立完成”模拟银行存取钱功能”程序的源代码编写、编译以及运行。
* 通过存取款线程理解多线程安全问题的发生原因，并掌握如何解决多线程安全问题。

### 【案例思路】

1. 通过任务描述和运行结果可以看出，该任务需要使用多线程的相关知识来是实现。由于两个用户操作各自的账户，因此我们需要创建两个线程完成每个用户的操作。这里我们使用实现Runnable接口的方法来创建线程。
2. 既然是储户去银行存款，那么可以得出该任务会涉及到三个类，分别是银行类、储户类和测试类。
3. 定义一个实体类作为账务的集合，包括用户名、登录名、登录密码、钱包、取钱时间和存钱时间等字段。
4. 在银行类中需要定义一个账户的实体类、一个存钱的方法、一个取钱的方法、查询余额的方法和获取当前用户的方法。获取等前用户方法需要使用synchronized线程锁判断是是哪一位用户，在存钱和取钱的方法中先调用获取用户方法判断操作者，再进行存取钱操作，需要注意的是在进行取钱操作是，需要判断余额是否大于需要取的钱数。
5. 在测试类中使用for循环调用线程模拟用户存取钱操作。

### 【案例代码】

1. 创建用户类

定义一个用户的类，根据用户实现多人同时存取钱功能，如文件8-1所示。

文件8-1 User.java

1. package chapter0403;
2. import java.util.Date;
3. public class User {
4. private String u\_name;//用户名
5. private String u\_login\_name;//登录名 卡的id
6. private String u\_login\_pwd;//登录密码
7. private String u\_wallet;//钱包
8. private Date draw\_money\_time;//取钱时间
9. private Date save\_money\_time;//存钱时间
10. public User(){}
11. public User(String u\_name, String u\_login\_name, String u\_login\_pwd,
12. String u\_wallet) {
13. this.u\_name = u\_name;
14. this.u\_login\_name = u\_login\_name;
15. this.u\_login\_pwd = u\_login\_pwd;
16. this.u\_wallet = u\_wallet;
17. }
18. public User(String u\_name, String u\_login\_name, String u\_login\_pwd,
19. String u\_wallet, Date draw\_money\_time, Date save\_money\_time) {
20. this.u\_name = u\_name;
21. this.u\_login\_name = u\_login\_name;
22. this.u\_login\_pwd = u\_login\_pwd;
23. this.u\_wallet = u\_wallet;
24. this.draw\_money\_time = draw\_money\_time;
25. this.save\_money\_time = save\_money\_time;
26. }
27. public String getU\_name() {
28. return u\_name;
29. }
30. public void setU\_name(String u\_name) {
31. this.u\_name = u\_name;
32. }
33. public String getU\_login\_name() {
34. return u\_login\_name;
35. }
36. public void setU\_login\_name(String u\_login\_name) {
37. this.u\_login\_name = u\_login\_name;
38. }
39. public String getU\_login\_pwd() {
40. return u\_login\_pwd;
41. }
42. public void setU\_login\_pwd(String u\_login\_pwd) {
43. this.u\_login\_pwd = u\_login\_pwd;
44. }
45. public String getU\_wallet() {
46. return u\_wallet;
47. }
48. public void setU\_wallet(String u\_wallet) {
49. this.u\_wallet = u\_wallet;
50. }
51. public Date getDraw\_money\_time() {
52. return draw\_money\_time;
53. }
54. public void setDraw\_money\_time(Date draw\_money\_time) {
55. this.draw\_money\_time = draw\_money\_time;
56. }
57. public Date getSave\_money\_time() {
58. return save\_money\_time;
59. }
60. public void setSave\_money\_time(Date save\_money\_time) {
61. this.save\_money\_time = save\_money\_time;
62. }
63. }
64. 创建银行业务类

定义一个业务类，实现用户的存取钱功能，如文件8-2所示。

文件8-2 Bank.java

1. package chapter0403;
2. import java.math.BigDecimal;
3. import java.text.SimpleDateFormat;
4. import java.util.ArrayList;
5. import java.util.Date;
6. import java.util.List;
7. public class Bank {
8. private List<User> userList=new ArrayList<>();
9. public Bank(List<User> userList) {
10. this.userList = userList;
11. }
12. public List<User> getUserList() {
13. return userList;
14. }
15. public void setUserList(List<User> userList) {
16. this.userList = userList;
17. }
18. //存钱
19. public Boolean saveMoney(String card,String pwd,String moneyNum){
20. User u=getUserByCard(card);
21. synchronized (Bank.class) {
22. if (u.getU\_login\_name().equals(card) &&
23. u.getU\_login\_pwd().equals(pwd)) {
24. BigDecimal oldData=new BigDecimal(u.getU\_wallet());
25. BigDecimal money=new BigDecimal(moneyNum);
26. u.setU\_wallet(oldData.add(money).toString());
27. u.setSave\_money\_time(new Date());
28. System.out.println(Thread.currentThread().getName()+"存钱
29. ---->"+u.getU\_name()+"在"+new SimpleDateFormat("yyyy-MM-dd
30. HH:mm:ss").format(u.getSave\_money\_time())+"存["+moneyNum+"]钱，余额：
31. "+u.getU\_wallet());
32. return true;
33. }
34. }
35. System.out.println(getUserByCard(card).getU\_name()+"存钱失败");
36. return false;
37. }
38. //取钱
39. public Boolean getMoney(String card,String pwd,String moneyNum){
40. User u=getUserByCard(card);
41. synchronized (Bank.class) {
42. if (u!=null && u.getU\_login\_name().equals(card) &&
43. u.getU\_login\_pwd().equals(pwd)) {
44. BigDecimal oldData=new BigDecimal(u.getU\_wallet());
45. BigDecimal money=new BigDecimal(moneyNum);
46. if(oldData.compareTo(money)>=0){
47. u.setU\_wallet(oldData.subtract(money).toString());
48. u.setDraw\_money\_time(new Date());
49. System.out.println(Thread.currentThread().getName()+"取钱
50. ---->"+u.getU\_name()+"在"+new SimpleDateFormat("yyyy-MM-dd
51. HH:mm:ss").format(u.getDraw\_money\_time())+"取["+moneyNum+"]钱
52. ,余额："+u.getU\_wallet());
53. return true;
54. }else {
55. System.out.println(getUserByCard(card).getU\_name()+"
56. 要取["+moneyNum+"]钱,但余额不足");
57. return false;
58. }
59. }
60. }
61. System.out.println(card+"取钱失败");
62. return false;
63. }
64. //查询余额
65. public String balanceEnquiry(String card,String pwd){
66. for(User u :this.userList){
67. if(u.getU\_login\_name().equals(card)&&
68. u.getU\_login\_pwd().equals(pwd)){
69. System.out.println(Thread.currentThread().getName()+":"
70. +u.getU\_name()+"余额："+u.getU\_wallet());
71. return u.getU\_wallet();
72. }
73. }
74. System.out.println(Thread.currentThread().getName()+":"+card+"
75. 操作失败");
76. return null;
77. }
78. //获取当前用户
79. public synchronized User getUserByCard(String card){
80. for(User u :this.userList){
81. if(u.getU\_login\_name().equals(card)){
82. return u;
83. }
84. }
85. return null;
86. }
87. public void delayTime(Integer nim){
88. try {
89. Thread.sleep(nim);
90. } catch (InterruptedException e) {
91. e.printStackTrace();
92. }
93. }
94. }
95. 创建测试类

定义测试类，再类中创建客户对象，并创建和开启线程执行存取钱操作，如文件8-3所示。

1. package chapter0403;
2. import java.util.ArrayList;
3. import java.util.List;
4. public class BankText {
5. public static void main(String[] args) throws Exception {
6. User u = new User("张三", "132466", "123", "100");
7. User uu = new User("李四", "4600882", "123", "0");
8. List<User> list = new ArrayList<>();
9. list.add(u);
10. list.add(uu);
11. Bank atm = new Bank(list);//初始化数据 模拟
12. Thread t = new Thread() {
13. public void run() {
14. for (int i = 0; i < 10; i++) {
15. atm.saveMoney("132466", "123", "12");
16. atm.delayTime(250);
17. atm.getMoney("4600882", "123", "14");
18. atm.delayTime(250);
19. }
20. }
21. };
22. Thread tt = new Thread() {
23. public void run() {
24. for (int i = 0; i < 10; i++) {
25. atm.getMoney("132466", "123", "2");
26. atm.delayTime(250);
27. atm.saveMoney("4600882", "123", "12");
28. atm.delayTime(250);
29. }
30. }
31. };
32. t.start();
33. tt.start();
34. }
35. }