

Credit
3130
Assignment
E2 Account

tests personal and business accounts

How has your program changed from planning to coding to now? Please explain?

Tester

```
7 tests personal and business accounts
8 */
9
10 public class AccountClient
11 {
12     public static void main(String[] args)
13     {
14         // scanner for input
15         Scanner input = new Scanner(System.in);
16         // menu choice
17         char choice;
18
19         // hard coded accounts
20         PersonalAcct personal = new PersonalAcct(150.0, "john", "smith");
21         BusinessAcct business = new BusinessAcct(600.0, "mary", "brown");
22
23         // main loop
24         do
25         {
26             // print menu
27             System.out.println("\n--- main menu ---");
28             System.out.println("1 personal account");
29             System.out.println("2 business account");
30             System.out.println("q quit");
31             System.out.print("select option: ");
32             String line = input.nextLine();
33             if (line.isEmpty())
34             {
35                 System.out.println("invalid input try again");
36                 choice = 'x';
37             }
38             else
39             {
40                 choice = Character.toLowerCase(line.charAt(0));
41             }
42
43             // select personal
44             if (choice == '1')
45             {
46                 handleAccount(personal, input);
47             }
48             // select business
49             else if (choice == '2')
50             {
51                 handleAccount(business, input);
52             }
53             // invalid menu
54             else if (choice != 'q')
55             {
56                 System.out.println("invalid selection");
57             }
58
59         } while (choice != 'q');
```

```

58     } while (choice != 'q');
59
60
61     // end program
62     System.out.println("program terminated");
63     input.close();
64 }
65
66 // account actions
67 public static void handleAccount(Account acct, Scanner input)
68 {
69     // action loop
70     boolean done = false;
71
72     while (!done)
73     {
74         // print action menu
75         System.out.println("\n(e) deposit");
76         System.out.println("(w) withdraw");
77         System.out.println("(v) view balance");
78         System.out.println("(b) back");
79         System.out.print("choose action: ");
80         String line = input.nextLine();
81         if (line.isEmpty())
82         {
83             System.out.println("invalid input");
84         }
85         else
86         {
87             char action = Character.toLowerCase(line.charAt(0));
88
89             // deposit money
90             if (action == 'e')
91             {
92                 System.out.print("enter deposit amount: ");
93                 try
94                 {
95                     double amt = Double.parseDouble(input.nextLine());
96                     acct.deposit(amt);
97                 }
98                 catch (NumberFormatException e)
99                 {
100                     System.out.println("invalid number");
101                 }
102             }
103             // withdraw money
104             else if (action == 'w')
105             {
106                 System.out.print("enter withdrawal amount: ");
107                 try
108                 {
109                     double amt = Double.parseDouble(input.nextLine());
110                     acct.withdrawal(amt);

```

```

83     System.out.println("invalid input");
84 }
85 else
86 {
87     char action = Character.toLowerCase(line.charAt(0));
88
89     // deposit money
90     if (action == 'e')
91     {
92         System.out.print("enter deposit amount: ");
93         try
94         {
95             double amt = Double.parseDouble(input.nextLine());
96             acct.deposit(amt);
97         }
98         catch (NumberFormatException e)
99         {
100             System.out.println("invalid number");
101         }
102     }
103     // withdraw money
104     else if (action == 'w')
105     {
106         System.out.print("enter withdrawal amount: ");
107         try
108         {
109             double amt = Double.parseDouble(input.nextLine());
110             acct.withdrawal(amt);
111         }
112         catch (NumberFormatException e)
113         {
114             System.out.println("invalid number");
115         }
116     }
117     // view balance
118     else if (action == 'v')
119     {
120         System.out.println(acct.toString());
121     }
122     // go back
123     else if (action == 'b')
124     {
125         done = true;
126     }
127     else
128     {
129         System.out.println("invalid action");
130     }
131 }
132 }
133 }
134 }

```

Persona

```

1 package mastery;
2
3 /*
4  * personal account class
5  */
6
7 public class PersonalAcct extends Account
8 {
9     // minimum balance constant
10    private static final double MIN_BALANCE = 100.0;
11    // penalty constant
12    private static final double PENALTY = 2.0;
13
14    // constructor
15    public PersonalAcct(double bal, String fName, String lName)
16    {
17        super(bal, fName, lName);
18    }
19
20    // override withdrawal
21    public void withdrawal(double amt)
22    {
23        // do normal withdrawal
24        super.withdrawal(amt);
25
26        // check minimum balance
27        if (getBalance() < MIN_BALANCE)
28        {
29            super.withdrawal(PENALTY);
30        }
31    }
32 }
33

```

Bussienius

```

PersonalAcct.java BusinessAcct.java AccountClient.java Account.java Customer.java
1 package mastery;
2
3 /*
4  * business account class
5  */
6
7 public class BusinessAcct extends Account
8 {
9     // minimum balance constant
10    private static final double MIN_BALANCE = 500.0;
11    // penalty constant
12    private static final double PENALTY = 10.0;
13
14    // constructor
15    public BusinessAcct(double bal, String fName, String lName)
16    {
17        super(bal, fName, lName);
18    }
19
20    // override withdrawal
21    public void withdrawal(double amt)
22    {
23        // do normal withdrawal
24        super.withdrawal(amt);
25
26        // check minimum balance
27        if (getBalance() < MIN_BALANCE)
28        {
29            super.withdrawal(PENALTY);
30        }
31    }
32 }
33

```

Account

```

13
14
15
16@ /**
17  * constructor
18  * pre: none
19  * post: An account has been created. Balance and
20  * customer data has been initialized with parameters.
21  */
22@ public Account(double bal, String fName, String lName)//include street, city, province or state, postal code or zip code
23 {
24     balance = bal;
25     cust = new Customer(fName, lName);//this constructor should reflect the new additions above, street, city, province, postal code
26     acctID = fName.substring(0,1) + lName;
27 }
28
29
30@ /**
31  * constructor
32  * pre: none
33  * post: An empty account has been created with the specified account ID.
34  */
35@ public Account(String ID) {
36     balance = 0;
37     cust = new Customer("", "");
38     acctID = ID;
39 }
40
41
42@ /**
43  * Returns the account ID.
44  * pre: none
45  * post: The account ID has been returned.
46  */
47@ public String getID() {
48     return(acctID);
49 }
50
51
52@ /**
53  * Returns the current balance.
54  * pre: none
55  * post: The account balance has been returned.
56  */
57@ public double getBalance() {
58     return(balance);
59 }
60
61
62@ /**
63  * A deposit is made to the account.
64  * pre: none
65  * post: The balance has been increased by the amount of the deposit.

```

```

13 private String acctID;
14
15
16
17 /**
18  * constructor
19  * pre: none
20  * post: An account has been created. Balance and
21  * customer data has been initialized with parameters.
22  */
23 public Account(double bal, String fName, String lName) //include street, city, province or state, postal code or zip code
24 {
25     balance = bal;
26     cust = new Customer(fName, lName); //this constructor should reflect the new additions above, street, city, province, postal code
27     acctID = fName.substring(0,1) + lName;
28 }
29
30
31 /**
32  * constructor
33  * pre: none
34  * post: An empty account has been created with the specified account ID.
35  */
36 public Account(String ID) {
37     balance = 0;
38     cust = new Customer("", "");
39     acctID = ID;
40 }
41
42
43 /**
44  * Returns the account ID.
45  * pre: none
46  * post: The account ID has been returned.
47  */
48 public String getID() {
49     return(acctID);
50 }
51
52
53 /**
54  * Returns the current balance.
55  * pre: none
56  * post: The account balance has been returned.
57  */
58 public double getBalance() {
59     return(balance);
60 }
61
62
63 /**
64  * A deposit is made to the account.
65  * pre: none

```

Customer one

```

4
3- /**
4  * Customer class.
5  */
6
7  public class Customer {
8      private String firstName, lastName;
9
10     //create String variables street, city, province, postal code
11
12
13- /**
14  * constructor
15  * pre: none
16  * post: A Customer object has been created.
17  * Customer data has been initialized with parameters.
18  */
19- public Customer(String fName, String lName) //modify constructor to include street, city, province, postal code
20 {
21     firstName = fName;
22     lastName = lName;
23
24     //reflect the changes in the parameter
25 }
26
27
28 //create changeCity method that asks the user their city and records city in a variable above
29
30 //create changeStreet method that asks the user their street and records street in a variable above
31
32 //create changeProvince method that asks the user their province and records province in a variable above
33
34 //create changePostalCode method that asks the user their postal code and records postal code in a variable above
35
36
37
38- /**
39  * Returns a String that represents the Customer object.
40  * pre: none
41  * post: A string representing the Customer object has
42  * been returned.
43  */
44- public String toString() {
45     String custString;
46
47     //update this string so that it contains the street, city, province, and postal code
48     custString = firstName + " " + lastName + "\n";
49     return(custString);
50 }
51

```