

**Worksheet: Simple BankAccount Class**

Given the test code and output, write class BankAccount.

All fields must be made `private`.

Other than the constructor, there is to be no way to set a balance. The balance is to be modified using the `deposit` and `withdraw` methods. These methods are to input the number of dollars to increase or decrease, respectively, the account balance. To avoid errors in the storage of fractional values, the account balance is to be stored using type `int`, and input values are to be modified by multiplying any `double` dollar input value by 100, and truncating to an integer number of cents (1/100<sup>th</sup> of a dollar).

The `withdraw` method should only deduct the amount if there are sufficient funds in the bank account, and return `true` if the account was successfully deducted, and `false` if there are insufficient funds.

Test Code

```
1 public class TestBankAccount {
2     public static void main(String[] args) {
3         BankAccount yourAccount =
4             new BankAccount("Stewart Dent", 1);
5         System.out.println(yourAccount);
6         yourAccount.deposit(99.9999);
7         System.out.println(
8             yourAccount.getName() + "\n" +
9             yourAccount.getAccountNumber() + "\n" +
10            yourAccount.getBalance() + "\n");
11         yourAccount.setName("Stu Dent");
12         yourAccount.withdraw(90.01);
13         System.out.println(yourAccount);
14         BankAccount myAccount =
15             new BankAccount("Chris", 2, 100000.0);
16         System.out.println(myAccount);
17         myAccount.withdraw(0.9999);
18         System.out.println(myAccount);
19     }
20 }
```

Output

```
1 Account name:   Stewart Dent
2 Account number: 1
3 Account balance: 0.0
4 Stewart Dent
5 1
6 99.99
7 Account name:   Stu Dent
8 Account number: 1
9 Account balance: 9.98
10 Account name:   Chris
11 Account number: 2
12 Account balance: 100000.0
13 Account name:   Chris
14 Account number: 2
15 Account balance: 99999.01
```

**Hints:**

- The constructor parameters and the instructions above should inform you what fields are required for the class.
- The class will need two separate constructors. (Overload the constructor).
- Notice that we can call `println` directly on a `BankAccount` object and it will print some useful output. This means that the class requires a `toString` method.
- As all fields are to be `private`, accessing fields using the dot operator ( `.` ) to access fields is disallowed. For example, we cannot access a name field using `yourAccount.name`. Therefore, the class will require *modifier methods* (such as `setName`) to set any values that need to be set and *accessor methods* (such as `getBalance`) to get values that the user of the class will need access to.
- There is to be no way to change an account number of an account that has been created.