

Reference Manual

Generated by Doxygen 1.8.13

Contents

1	Test List	2
2	Class Index	4
2.1	Class List	4
3	File Index	5
3.1	File List	5
4	Class Documentation	6
4.1	BankAccount Class Reference	6
4.1.1	Detailed Description	7
4.1.2	Member Function Documentation	7
4.1.3	Member Data Documentation	11
4.2	Portfolio Class Reference	12
4.2.1	Detailed Description	13
4.2.2	Member Function Documentation	13
4.2.3	Member Data Documentation	20
5	File Documentation	21
5.1	bankAccount.h File Reference	21
5.2	portfolio.cpp File Reference	21
5.2.1	Function Documentation	22
5.3	portfolio.h File Reference	31
	Index	33

1 Test List

Member `TEST_CASE` ("deposit")

`deposit()` member function to check if depositing 100 into the checking will change the checking account to be 100. Calls [BankAccount](#) member function `deposit` A portfolio with one saving and one checking bank account.

```
portfolio.cpp:68: passed: ss.str() == "The amount in the checking account is
100\n" "The amount in the saving account is 0\n" for: "The amount in the
checking account is 100
The amount in the saving account is 0
"
==
"The amount in the checking account is 100
The amount in the saving account is 0
"
Passed 1 test case with 1 assertion.
```

Member `TEST_CASE` ("output")

`withdraw()` `deposit()` `transfer()` `print_balance()` member function to check checking if the output matches when all function are called Calls [BankAccount](#) all of member functions A portfolio with one saving and one checking bank account.

```
portfolio.cpp:175: passed: ss.str() == "The amount in the checking account is
300\n" "The amount in the saving account is 290\n" for: "The amount in the
checking account is 300
The amount in the saving account is 290
"
==
"The amount in the checking account is 300
The amount in the saving account is 290
"
Passed 1 test case with 1 assertion.
```

Member `TEST_CASE` ("print_balance")

`print_balance()` member function to check if multiple deposits, transfers, and withdrawals are made the function will print the total of both accounts Calls [BankAccount](#) member function `get_balance` A portfolio with one saving and one checking bank account.

```
portfolio.cpp:149: passed: ss.str() == "The amount in the checking account is
200\n" "The amount in the saving account is 200\n" for: "The amount in the
checking account is 200
The amount in the saving account is 200
```

```
"
==
"The amount in the checking account is 200
The amount in the saving account is 200
"
Passed 1 test case with 1 assertion.
```

Member **TEST_CASE** ("transfer")

transfer() member function to check if depositing 200 into the saving after adding 100 and transferring 100 from saving to checking will change the checking account to be 200 and saving to 100; Calls [BankAccount](#) member function withdraw and deposit A portfolio with one saving and one checking bank account.

```
portfolio.cpp:120: passed: ss.str() == "The amount in the checking account is
200\n" "The amount in the saving account is 100\n" for: "The amount in the
checking account is 200
The amount in the saving account is 100
"
==
"The amount in the checking account is 200
The amount in the saving account is 100
"
Passed 1 test case with 1 assertion.
```

Member **TEST_CASE** ("withdraw")

withdraw() member function to check if withdrawing 100 into the checking after adding 200 will change the checking account to be 100. Calls [BankAccount](#) member function withdraw A portfolio with one saving and one checking bank account.

```
portfolio.cpp:93: passed: ss.str() == "The amount in the checking account is
0\n" "The amount in the saving account is 100\n" for: "The amount in the
checking account is 0
The amount in the saving account is 100
"
==
"The amount in the checking account is 0
The amount in the saving account is 100
"
Passed 1 test case with 1 assertion.
```

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BankAccount	6
Portfolio	12

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

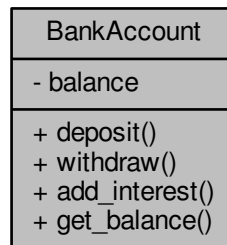
bankAccount.h	21
portfolio.cpp	21
portfolio.h	31

4 Class Documentation

4.1 BankAccount Class Reference

```
#include <bankAccount.h>
```

Collaboration diagram for BankAccount:



Public Member Functions

- void [deposit](#) (double amount)
- void [withdraw](#) (double amount)
- void [add_interest](#) (double rate)
- double [get_balance](#) () const

Private Attributes

- double `balance`

4.1.1 Detailed Description

A bank account whose balance can be changed by deposits and withdrawals.

Definition at line 4 of file `bankAccount.h`.

4.1.2 Member Function Documentation

4.1.2.1 `add_interest()`

```
void BankAccount::add_interest (  
    double rate )
```

Adds interest to this account.

Parameters

<i>rate</i>	the interest rate in percent
-------------	------------------------------

Definition at line 53 of file `bankAccount.h`.

```
54 {  
55     double amount = balance * rate / 100;
```



```
56     deposit (amount) ;  
57 }
```

Here is the call graph for this function:



4.1.2.2 deposit()

```
void BankAccount::deposit (  
    double amount )
```

Makes a deposit into this account.

Parameters

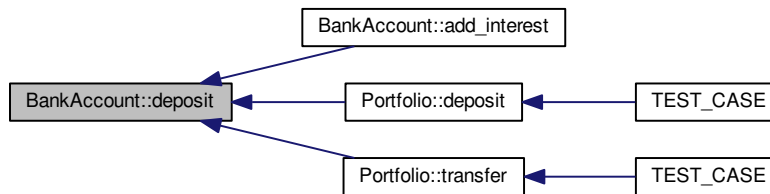
<i>amount</i>	the amount of the deposit
---------------	---------------------------

Definition at line 35 of file bankAccount.h.

```
36 {
```

```
37     balance = balance + amount;  
38 }
```

Here is the caller graph for this function:



4.1.2.3 get_balance()

```
double BankAccount::get_balance ( ) const
```

Gets the current balance of this bank account.

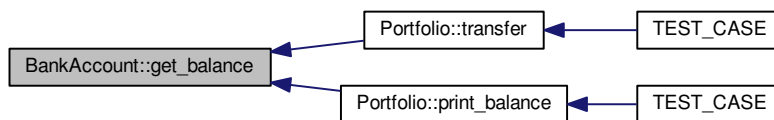
Returns

the current balance

Definition at line 59 of file `bankAccount.h`.

```
60 {  
61     return balance;  
62 }
```

Here is the caller graph for this function:



4.1.2.4 withdraw()

```
void BankAccount::withdraw (  
    double amount )
```

Makes a withdrawal from this account, or charges a penalty if sufficient funds are not available.

Parameters

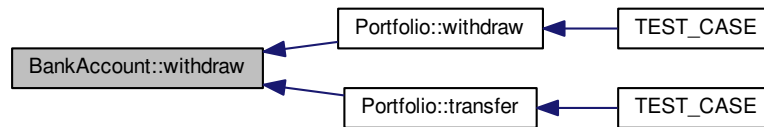
<i>amount</i>	the amount of the withdrawal
---------------	------------------------------

Definition at line 40 of file `bankAccount.h`.

```
41 {
```

```
42     const double PENALTY = 10;  
43     if (amount > balance)  
44     {  
45         balance = balance - PENALTY;  
46     }  
47     else  
48     {  
49         balance = balance - amount;  
50     }  
51 }
```

Here is the caller graph for this function:



4.1.3 Member Data Documentation

4.1.3.1 balance

```
double BankAccount::balance [private]
```

Definition at line 32 of file `bankAccount.h`.

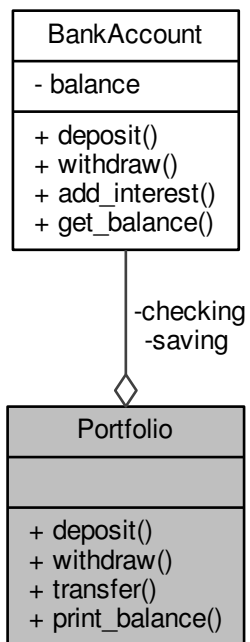
The documentation for this class was generated from the following file:

- [bankAccount.h](#)

4.2 Portfolio Class Reference

```
#include <portfolio.h>
```

Collaboration diagram for Portfolio:



Public Member Functions

- void [deposit](#) (double amount, std::string account)
- void [withdraw](#) (double amount, std::string account)
- void [transfer](#) (double amount, std::string account)
- void [print_balance](#) () const

Private Attributes

- [BankAccount](#) checking
- [BankAccount](#) saving

4.2.1 Detailed Description

A portfolio with one saving and one checking bank account. Money can be transfer or deposited or withdraw in each account

Definition at line 5 of file portfolio.h.

4.2.2 Member Function Documentation

4.2.2.1 deposit()

```
void Portfolio::deposit (
    double amount,
    std::string account )
```

Makes a deposit from the saving or checking account. Calls [BankAccount](#) member function deposit

Parameters

<i>amount</i>	the amount of the deposit
---------------	---------------------------

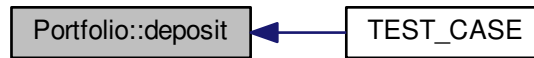
Definition at line 6 of file portfolio.cpp.

```
7 {  
8     if (account == "S")  
9         saving.deposit(amount);  
10    if (account == "C")  
11        checking.deposit(amount);  
12 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.2 print_balance()

```
void Portfolio::print_balance ( ) const
```

Prints the current balances of both bank accounts. Calls [BankAccount](#) member function `get_balance`

Definition at line 36 of file `portfolio.cpp`.

```
37 {  
38     double balance = checking.get_balance();  
39     std::cout << "The amount in the checking account is "  
40     << balance << std::endl;  
41     balance = saving.get_balance();  
42     std::cout << "The amount in the saving account is "  
43     << balance << std::endl;  
44 }
```


Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.3 transfer()

```
void Portfolio::transfer (
    double amount,
    std::string account )
```

Transfers an amount from saving or checking to the opposing account Calls [BankAccount](#) member function deposit and withdraw

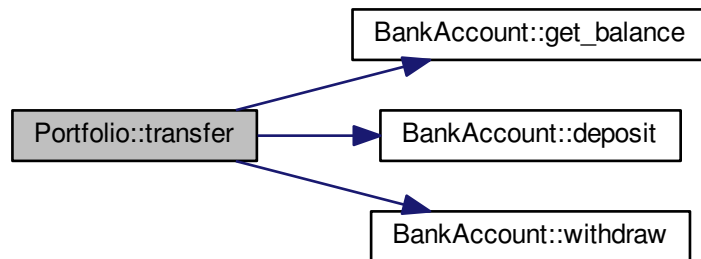
Parameters

<i>amount</i>	the amount of the transfer
---------------	----------------------------

Definition at line 20 of file portfolio.cpp.

```
21 {
22     if (account == "S")
23     {
24         if (amount <= saving.get_balance())
25             checking.deposit(amount);
26         saving.withdraw(amount);
27     }
28     if (account == "C")
29     {
30         if (amount <= checking.get_balance())
31             saving.deposit(amount);
32         checking.withdraw(amount);
33     }
34 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.4 withdraw()

```
void Portfolio::withdraw (
    double amount,
    std::string account )
```

Makes a withdraw from the saving or checking account. Calls [BankAccount](#) member function withdraw

Parameters

<i>amount</i>	the amount of the withdraw
---------------	----------------------------

Definition at line 13 of file portfolio.cpp.

```
14 {
15     if (account == "S")
16         saving.withdraw(amount);
17     if (account == "C")
18         checking.withdraw(amount);
19 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3 Member Data Documentation

4.2.3.1 checking

`BankAccount` `Portfolio::checking` [private]

Definition at line 36 of file `portfolio.h`.

4.2.3.2 saving

`BankAccount` `Portfolio::saving` [private]

Definition at line 37 of file `portfolio.h`.

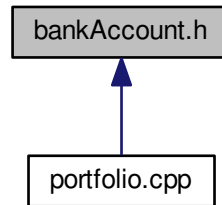
The documentation for this class was generated from the following files:

- [portfolio.h](#)
- [portfolio.cpp](#)

5 File Documentation

5.1 `bankAccount.h` File Reference

This graph shows which files directly or indirectly include this file:



Classes

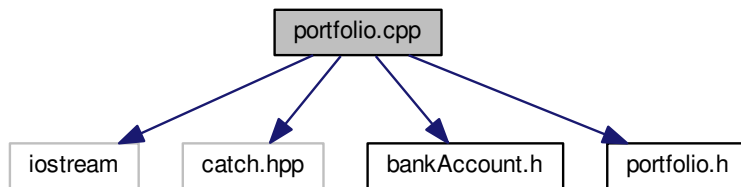
- class [BankAccount](#)

5.2 `portfolio.cpp` File Reference

```
#include <iostream>
#include "catch.hpp"
#include "bankAccount.h"
```

```
#include "portfolio.h"
```

Include dependency graph for portfolio.cpp:



Functions

- `TEST_CASE` ("deposit")
- `TEST_CASE` ("withdraw")
- `TEST_CASE` ("transfer")
- `TEST_CASE` ("print_balance")
- `TEST_CASE` ("output")

5.2.1 Function Documentation

5.2.1.1 TEST_CASE() [1/5]

```
TEST_CASE (
    "deposit" )
```

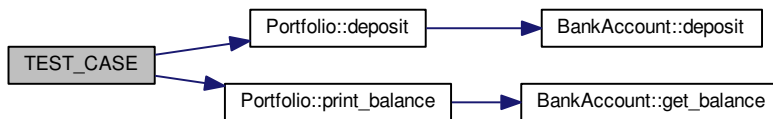
Test deposit() member function to check if depositing 100 into the checking will change the checking account to be 100. Calls [BankAccount](#) member function deposit A portfolio with one saving and one checking bank account.

```
portfolio.cpp:68: passed: ss.str() == "The amount in the checking account is
100\n" "The amount in the saving account is 0\n" for: "The amount in the
checking account is 100
The amount in the saving account is 0
"
==
"The amount in the checking account is 100
The amount in the saving account is 0
"
Passed 1 test case with 1 assertion.
```

Definition at line 54 of file portfolio.cpp.

```
55 {
56     Portfolio account1;
57     account1.deposit(100, "C");
58     std::streambuf *b = std::cout.rdbuf();
59     std::stringstream ss;
60     std::streambuf *sb = ss.rdbuf();
61     std::cout.rdbuf(sb);
62     // Now all output will be redirected into ss
63     account1.print_balance();
64     // set output back to the terminal
65     std::cout.rdbuf(b);
66     CHECK(ss.str() ==
67         "The amount in the checking account is 100\n"
68         "The amount in the saving account is 0\n");
69 }
```


Here is the call graph for this function:



5.2.1.2 TEST_CASE() [2/5]

```
TEST_CASE (
    "withdraw" )
```

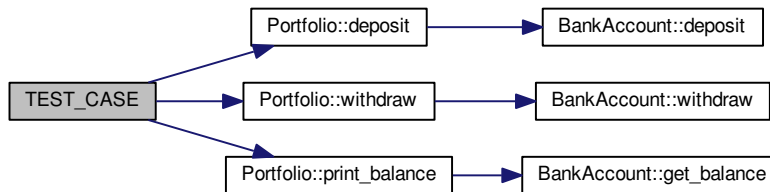
Test `withdraw()` member function to check if withdrawing 100 into the checking after adding 200 will change the checking account to be 100. Calls [BankAccount](#) member function `withdraw` A portfolio with one saving and one checking bank account.

```
portfolio.cpp:93: passed: ss.str() == "The amount in the checking account is
0\n" "The amount in the saving account is 100\n" for: "The amount in the
checking account is 0
The amount in the saving account is 100
"
==
"The amount in the checking account is 0
The amount in the saving account is 100
"
Passed 1 test case with 1 assertion.
```

Definition at line 78 of file `portfolio.cpp`.

```
79 {
80     Portfolio account1;
81     account1.deposit(200, "S");
82     account1.withdraw(100, "S");
83     std::streambuf *b = std::cout.rdbuf();
84     std::stringstream ss;
85     std::streambuf *sb = ss.rdbuf();
86     std::cout.rdbuf(sb);
87     // Now all output will be redirected into ss
88     account1.print_balance();
89     // set output back to the terminal
90     std::cout.rdbuf(b);
91     CHECK(ss.str() ==
92         "The amount in the checking account is 0\n"
93         "The amount in the saving account is 100\n");
94 }
```

Here is the call graph for this function:



5.2.1.3 TEST_CASE() [3/5]

```
TEST_CASE (
    "transfer" )
```

Test transfer() member function to check if depositing 200 into the saving after adding 100 and transferring 100 from saving to checking will change the checking account to be 200 and saving to 100; Calls [BankAccount](#) member function withdraw and deposit A portfolio with one saving and one checking bank account.

```
portfolio.cpp:120: passed: ss.str() == "The amount in the checking account is
200\n" "The amount in the saving account is 100\n" for: "The amount in the
checking account is 200
The amount in the saving account is 100
"
==
"The amount in the checking account is 200
The amount in the saving account is 100
"
Passed 1 test case with 1 assertion.
```

Definition at line 104 of file portfolio.cpp.

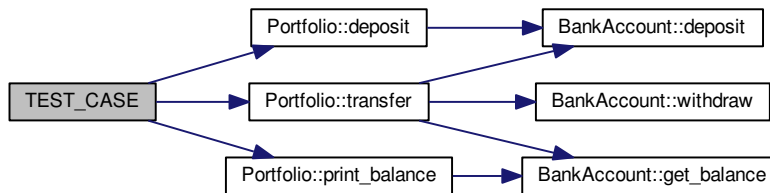
```
105 {
106     Portfolio account1;
107     account1.deposit(200,"S");
108     account1.deposit(100,"C");
109     account1.transfer(100,"S");
110     std::streambuf *b = std::cout.rdbuf();
111     std::stringstream ss;
112     std::streambuf *sb = ss.rdbuf();
113     std::cout.rdbuf(sb);
114     // Now all output will be redirected into ss
115     account1.print_balance();
116     // set output back to the terminal
```

```

117     std::cout.rdbuf(b) ;
118     CHECK(ss.str() ==
119     "The amount in the checking account is 200\n"
120     "The amount in the saving account is 100\n");
121 }

```

Here is the call graph for this function:



5.2.1.4 TEST_CASE() [4/5]

```

TEST_CASE (
    "print_balance" )

```

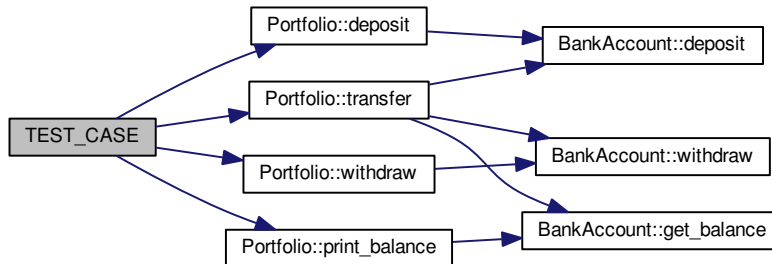
Test `print_balance()` member function to check if multiple deposits, transfers, and withdrawals are made the function will print the total of both accounts Calls [BankAccount](#) member function `get_balance` A portfolio with one saving and one checking bank account.

```
portfolio.cpp:149: passed: ss.str() == "The amount in the checking account is
200\n" "The amount in the saving account is 200\n" for: "The amount in the
checking account is 200
The amount in the saving account is 200
"
==
"The amount in the checking account is 200
The amount in the saving account is 200
"
Passed 1 test case with 1 assertion.
```

Definition at line 130 of file portfolio.cpp.

```
131 {
132     Portfolio account1;
133     account1.deposit(500,"S");
134     account1.withdraw(100,"S");
135     account1.withdraw(100,"S");
136     account1.deposit(100,"C");
137     account1.transfer(200,"S");
138     account1.transfer(100,"C");
139     std::streambuf *b = std::cout.rdbuf();
140     std::stringstream ss;
141     std::streambuf *sb = ss.rdbuf();
142     std::cout.rdbuf(sb);
143     // Now all output will be redirected into ss
144     account1.print_balance();
145     // set output back to the terminal
146     std::cout.rdbuf(b);
147     CHECK(ss.str() ==
148         "The amount in the checking account is 200\n"
149         "The amount in the saving account is 200\n");
150 }
```

Here is the call graph for this function:



5.2.1.5 TEST_CASE() [5/5]

```
TEST_CASE (
    "output" )
```

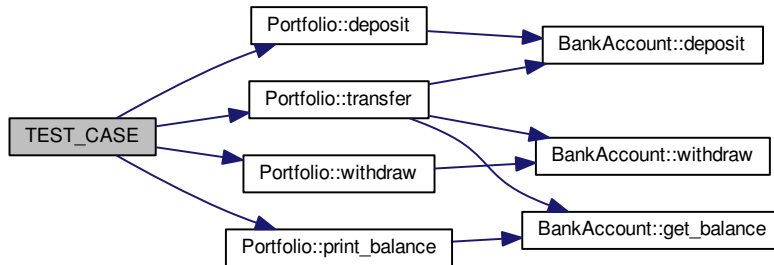
Test `withdraw()` `deposit()` `transfer()` `print_balance()` member function to check checking if the output matches when all function are called Calls [BankAccount](#) all of member functions A portfolio with one saving and one checking bank account.

```
portfolio.cpp:175: passed: ss.str() == "The amount in the checking account is
300\n" "The amount in the saving account is 290\n" for: "The amount in the
checking account is 300
The amount in the saving account is 290
"
==
"The amount in the checking account is 300
The amount in the saving account is 290
"
Passed 1 test case with 1 assertion.
```

Definition at line 158 of file portfolio.cpp.

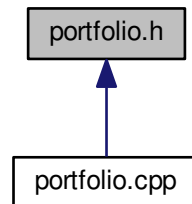
```
159 {
160     Portfolio account1;
161     account1.deposit(500,"C");
162     account1.deposit(300,"S");
163     account1.withdraw(200,"C");
164     account1.transfer(500,"S");
165     std::streambuf *b = std::cout.rdbuf();
166     std::stringstream ss;
167     std::streambuf *sb = ss.rdbuf();
168     std::cout.rdbuf(sb);
169     // Now all output will be redirected into ss
170     account1.print_balance();
171     // set output back to the terminal
172     std::cout.rdbuf(b);
173     CHECK(ss.str() ==
174         "The amount in the checking account is 300\n"
175         "The amount in the saving account is 290\n");
176 }
```

Here is the call graph for this function:



5.3 portfolio.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class [Portfolio](#)

Index

add_interest
 BankAccount, 7

balance
 BankAccount, 11

BankAccount, 6
 add_interest, 7
 balance, 11
 deposit, 8
 get_balance, 9
 withdraw, 10

bankAccount.h, 21

checking
 Portfolio, 20

deposit
 BankAccount, 8
 Portfolio, 13

get_balance
 BankAccount, 9

Portfolio, 12
 checking, 20
 deposit, 13
 print_balance, 15
 saving, 20
 transfer, 16
 withdraw, 18

portfolio.cpp, 21
 TEST_CASE, 22, 24, 25, 27, 29
portfolio.h, 31

print_balance
 Portfolio, 15

saving
 Portfolio, 20

TEST_CASE
 portfolio.cpp, 22, 24, 25, 27, 29

transfer
 Portfolio, 16

withdraw
 BankAccount, 10
 Portfolio, 18