COEN 244

Tutorial 7

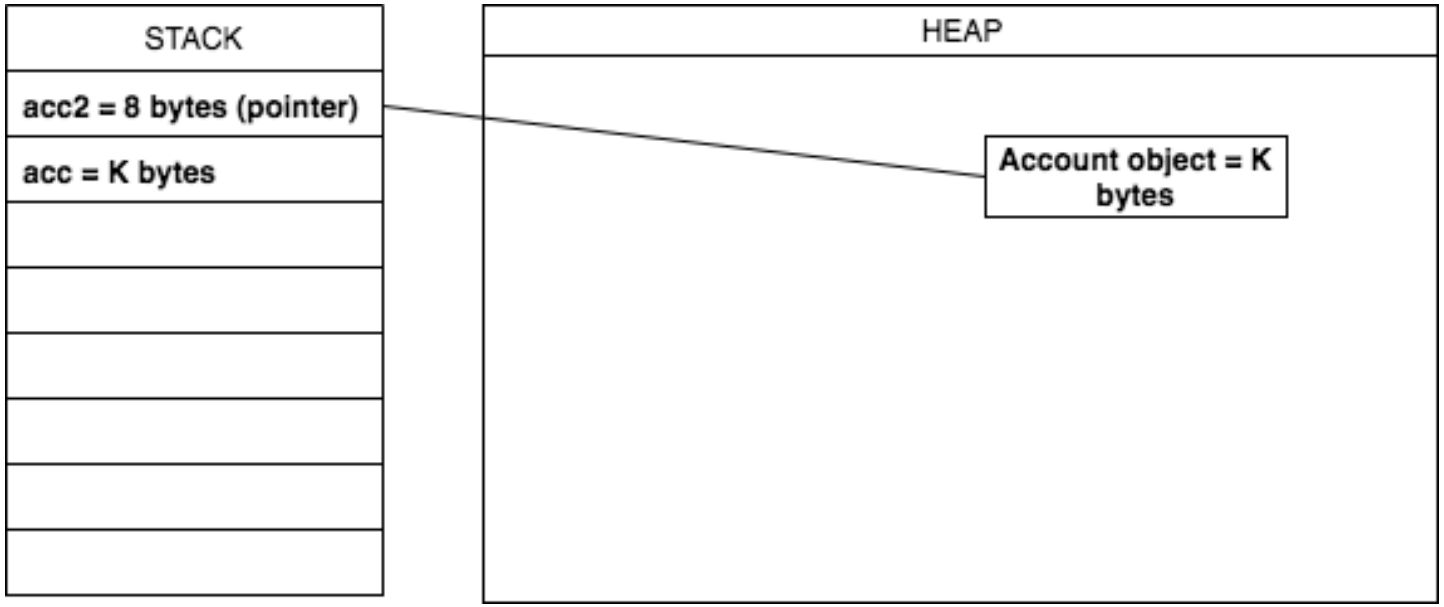
Mohammad Altahat

Account Class Example

```
#pragma once

#include <iostream>
using namespace std;

class Account
{
	private:
		int accountNum; // account number
		double balance; // the balance
	public:

	Account()
	{
		cout << "account default constructor called" << endl;
		accountNum = 0;
		balance = 0;
	} // default constructor

	Account(int an, double b)
	{
		cout << "account regular constructor called" << endl;
		accountNum = an;
		balance = b;
	} // regular constructor

	Account(const Account& anotherAccount)
	{
		cout << "account copy constructor called" << endl;
		accountNum = anotherAccount.accountNum;
		balance = anotherAccount.balance;
	} // copy constructor

	~Account()
	{


	} // destructor. Does nothing
};
```

Account Class Example

```
int main()
{
        Account acc;
        cout << "after Account acc" << endl;


        Account* acc2;
        cout << "after Account* acc2" << endl;


        acc2 = new Account(20, 2.5);
        cout << "after acc2= new Account(20,2.5)" << endl;


        return 0;
}
```

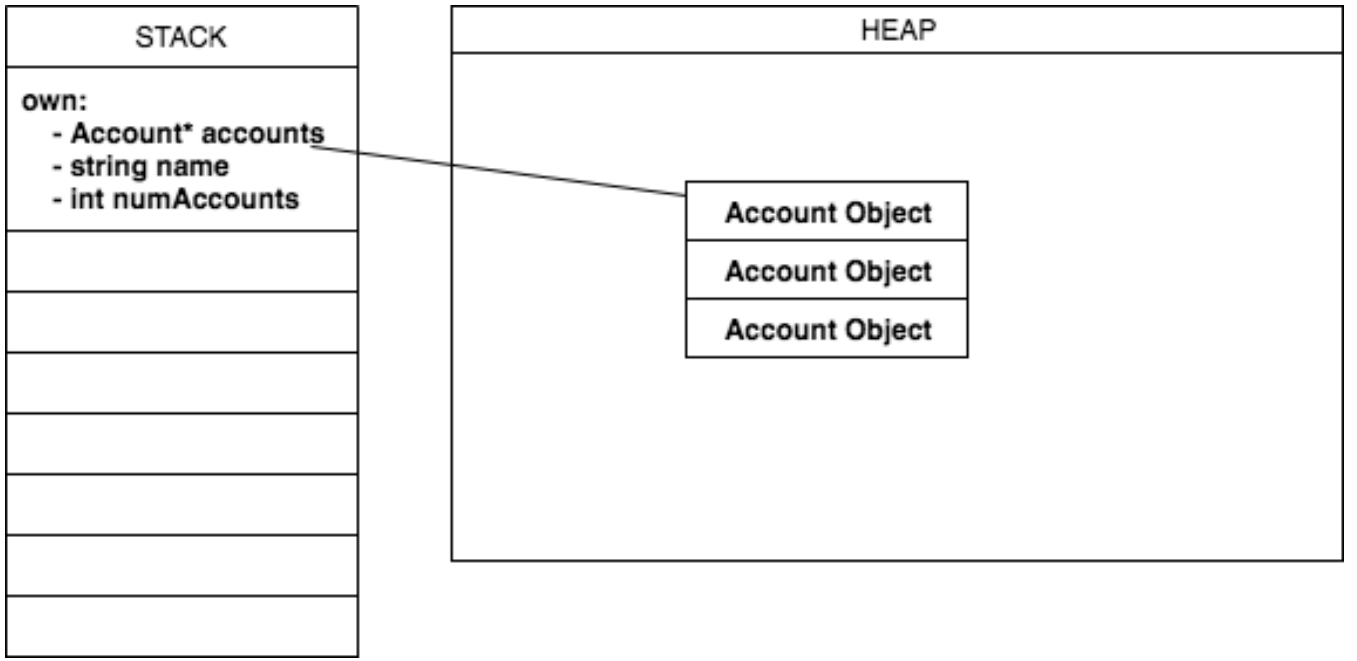


Owner Class Example – Array Implementation

```
#pragma once

#include <string>
#include "Account.h"
#include <array>

using namespace std;

class Owner
{
	private:
		string name; // name of the person
		int numAccounts; // represents the current number of accounts held by this owner
		Account * accounts ; // list of accounts of this person, assume that an owner can have a max of 3 accounts
		//array<Account*,3> accounts; //array of Account pointers

	public:
	Owner()
	{
		name = "";
		numAccounts = 0;
		accounts = new Account[3];

		//for(int i=0; i<accounts.size(); i++)
		// accounts[i] = nullptr; // default constructor

	} // default constructor

	~Owner()
	{
		delete[] accounts; // delete all attributes that are pointers
	}


};
```

Owner Class Example – Array Implementation

```
int main()
{
        Owner own;
        return 0;
}
```



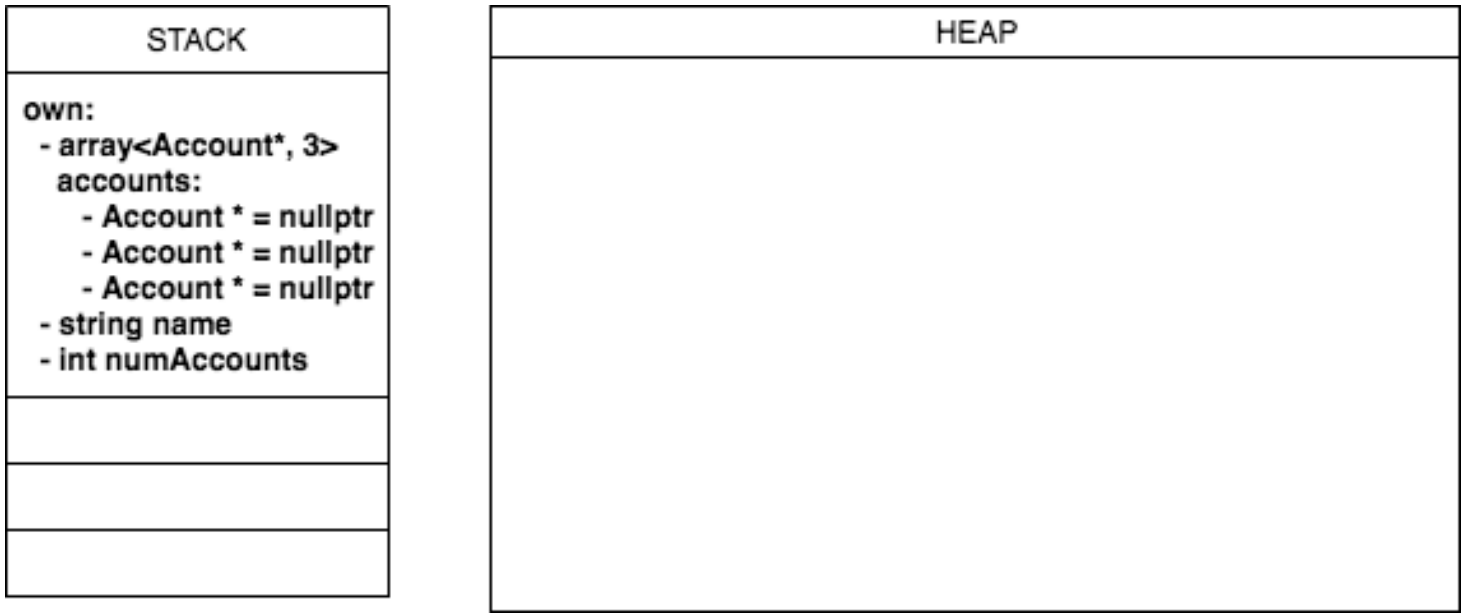
Owner Class Example – Array of pointers Implementation

```
array<Account*,3> accounts; //array of Account pointers
```

```
Owner()
{
	name = "";
	numAccounts = 0;

	for(int i=0; i < accounts.size(); i++)
		accounts[i] = nullptr;

} // default constructor
```

Owner Class Example – Array of pointers Implementation

```
int main()
{
        Owner own;
        return 0;
}
```

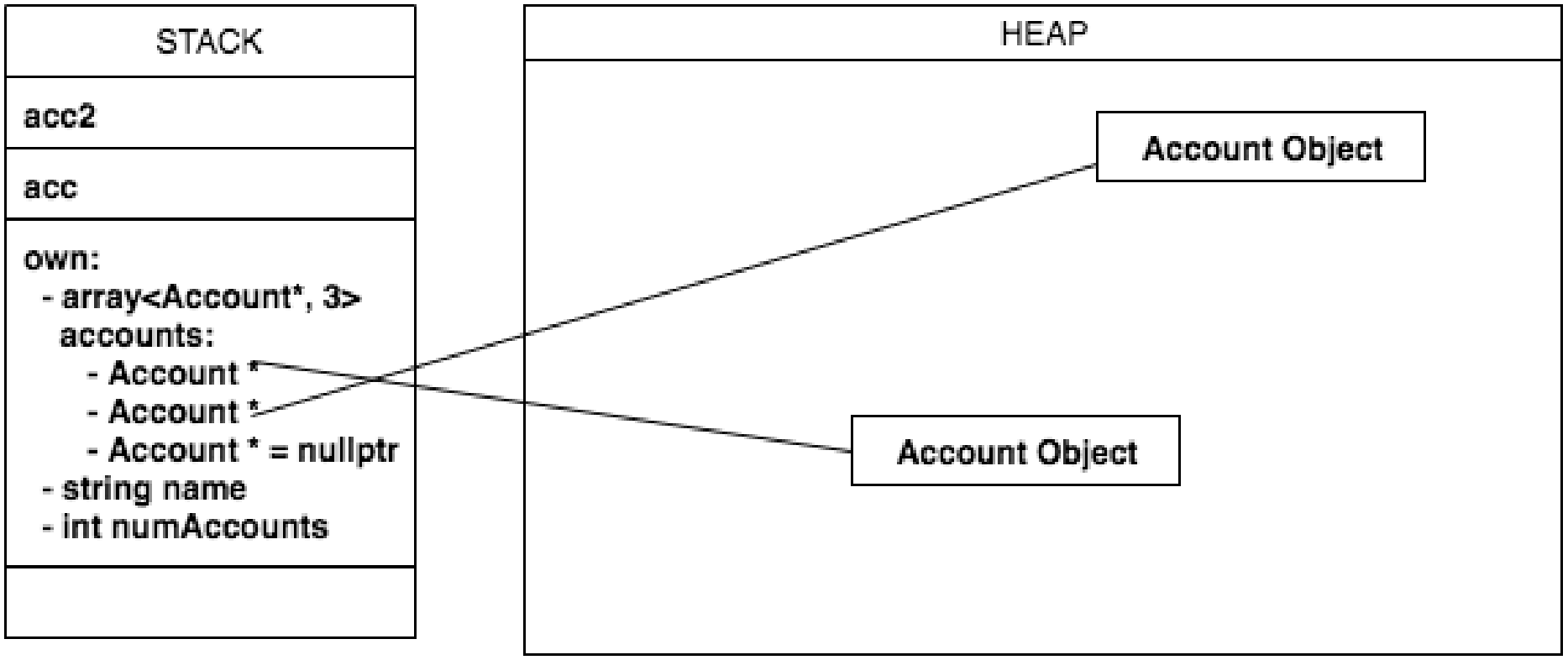


Owner Class Example – Array of pointers
Implementation – Add Accounts to owner

```
void addAccount(const Account& a)
{
	if (numAccounts < 10) {
	    accounts[numAccounts] = new Account(a);
	    numAccounts++;
	}
	else {
	   cout << "Account cannot be added number of accounts exceeded " << endl;
	}
}
```

```
int main()
{
	Owner own;
	Account acc(20,2.5);
	Account acc2;
	own.addAccount(acc);
	own.addAccount(acc2);
	return 0;
}
```

Owner Class Example – Array of pointers
Implementation – Add Accounts to owner



Copy Constructor

- In C++, a Copy Constructor may be called in following cases:
  1. When an object of the class is returned by value.
  2. When an object of the class is passed (to a function) by value as an argument.
  3. When an object is constructed based on another object of the same class.
  4. When the compiler generates a temporary object.*

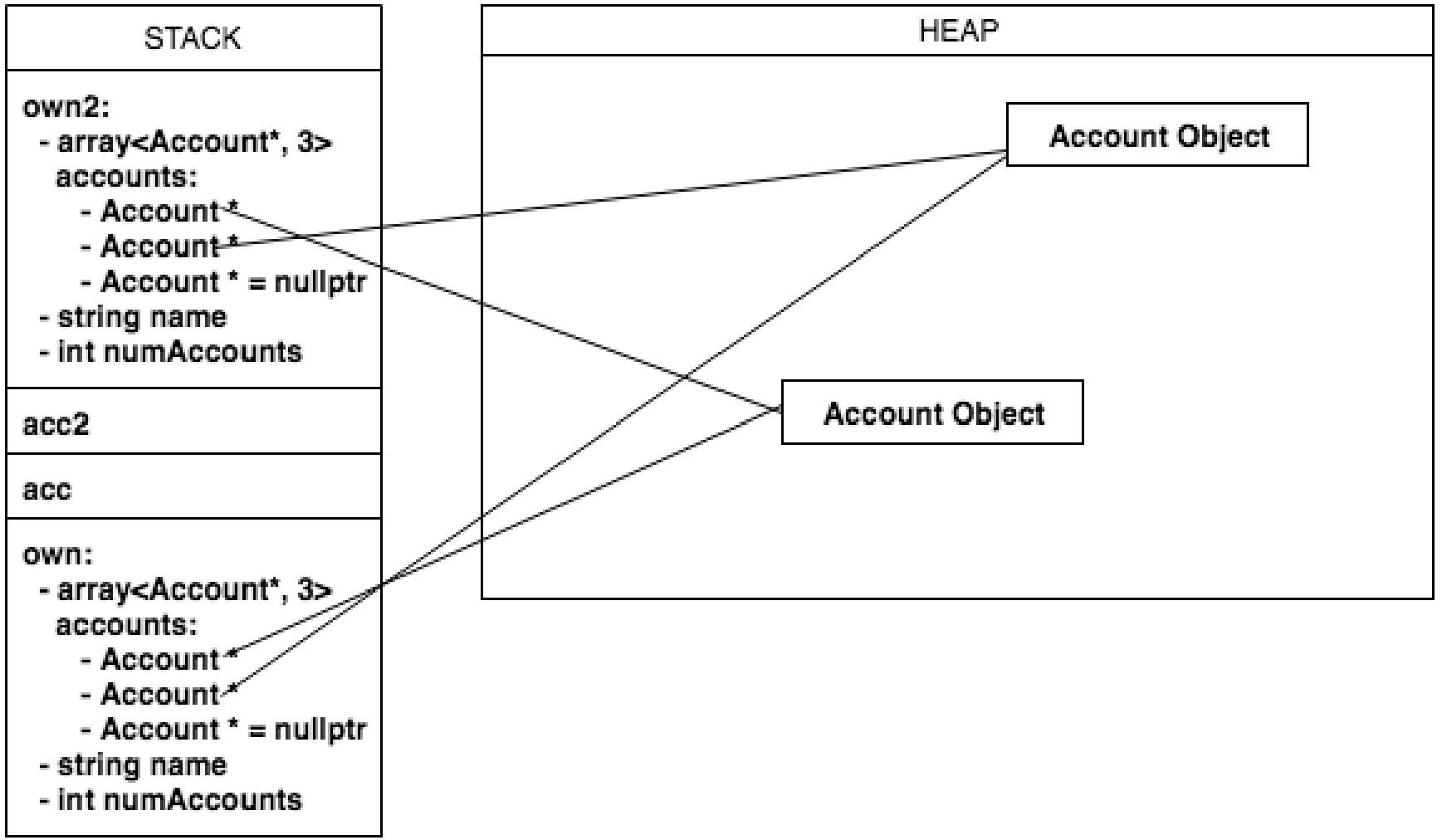
Copy Constructor – Owner Class

```
//copy constructor added to Owner.h
Owner(const Owner& anotherOwner)
{
    name = anotherOwner.name;
    numAccounts = anotherOwner.numAccounts;
    accounts = anotherOwner.accounts;
}
```

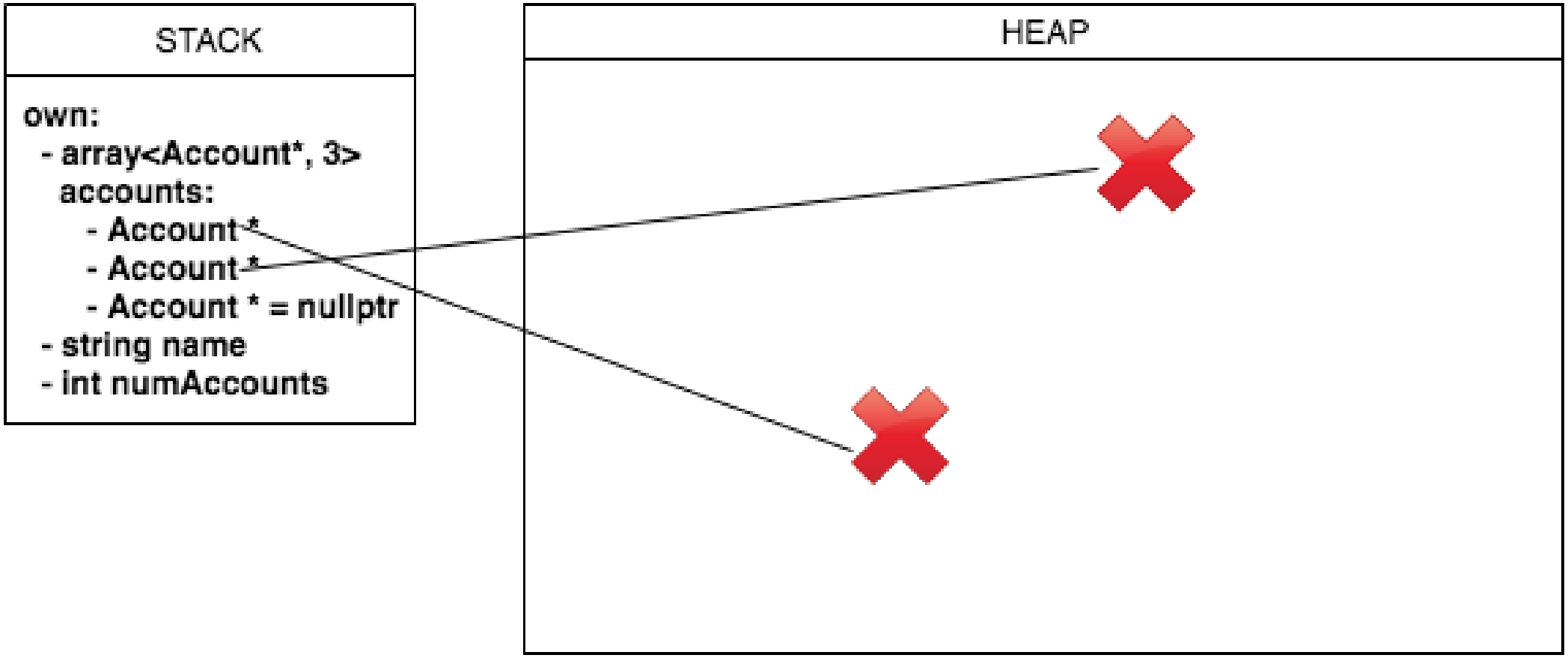
Copy Constructor – Owner Class

```
int main()
{
    Owner own;
    Account acc(20,2.5);
    Account acc2;
    own.addAccount(acc);
    own.addAccount(acc2);
    Owner own2(own);
    return 0;
}
```

Shallow Copy



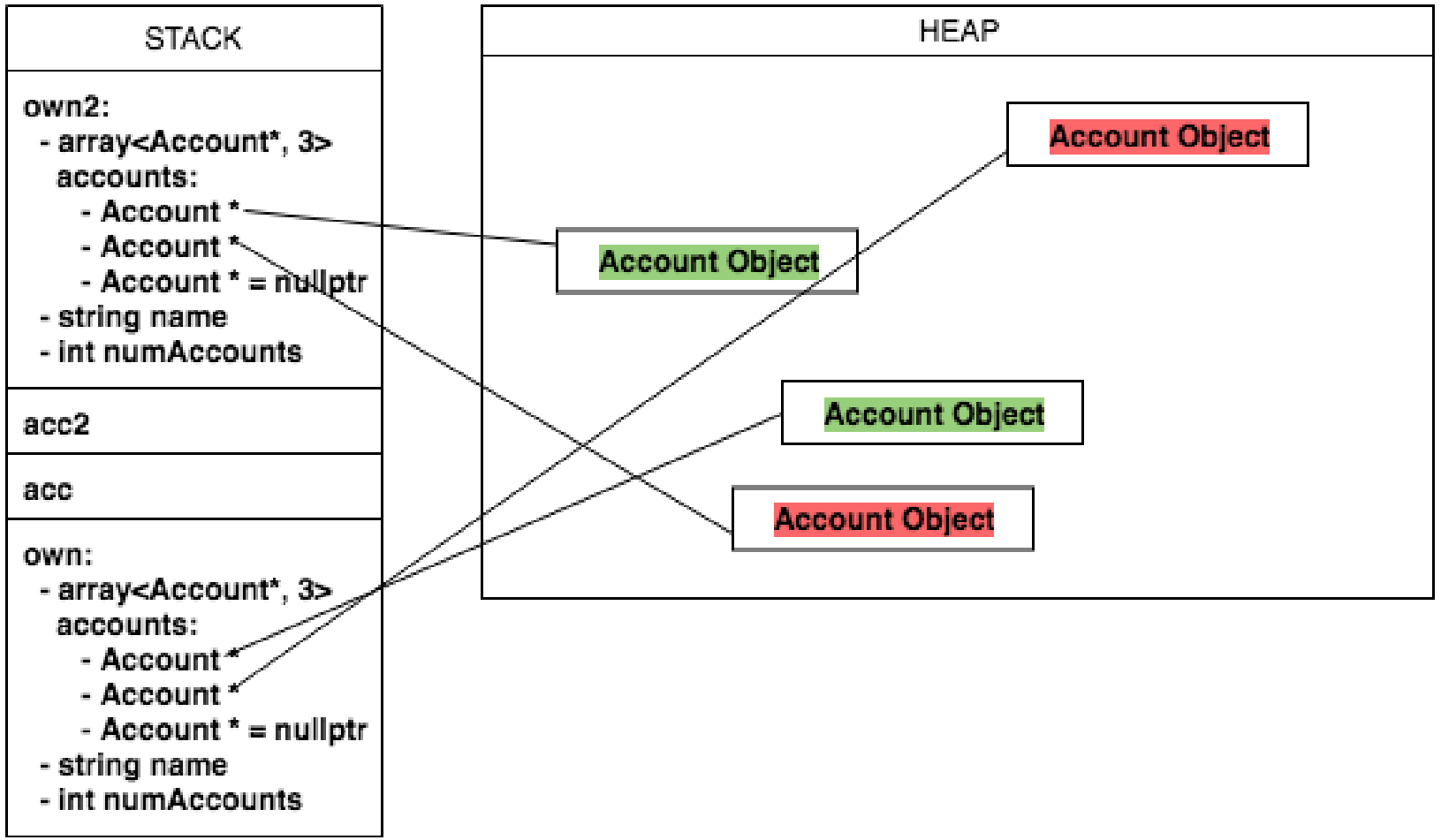
Shallow Copy



Copy Constructor – Owner Class

```
//copy constructor added to Owner.h
Owner(const Owner& anotherOwner)
{
    name = anotherOwner.name;
    numAccounts = anotherOwner.numAccounts;

    for(int i=0; i < anotherOwner.accounts.size(); i++)
    {
        if(anotherOwner.accounts[i] == nullptr)
                accounts[i] = nullptr;
        else
                accounts[i] = new Account(*anotherOwner.accounts[i]);
    }
}
```

Deep Copy


STACK
own2:
- array<Account*, 3>
accounts:
- Account *
- Account *
- Account * = nullptr
- string name
- int numAccounts
acc2
acc
own:
- array<Account*, 3>
accounts:
- Account *
- Account *
- Account * = nullptr
- string name
- int numAccounts
HEAP
Account Object
Account Object
Account Object
Account Object