Object Basics

Bank Example

- In today's example, we'll iteratively build a simple banking system
- Download the example and untar it to follow along
  - I won't be able to display everything at once
  - You can use this to get a broader view of what's going on

Files

- **Bank.[h/cpp]**
- Account.[h/cpp]
- Money.[h/cpp]
- Main.cpp
- Makefile

- Main interface for users
- Provides a means of creating and accessing a personalized **Account**
- Allows users to withdraw and deposit **Money**

Files

- Bank.[h/cpp]
- **Account.[h/cpp]**
- Money.[h/cpp]
- Main.cpp
- Makefile

- Container for account information (e.g. name, id, and **Money**)
- Provides only accessor functions so users can't modify the account
- **Bank** is a friend class allowing it to modify the account

Files

- Bank.[h/cpp]
- Account.[h/cpp]
- **Money.[h/cpp]**
- Main.cpp
- Makefile

- Special integer representation
- Provides several constructors and operators

Files

- Bank.[h/cpp]
- Account.[h/cpp]
- Money.[h/cpp]
- **Main.cpp**
- Makefile

- Driver to illustrate how the everything works
- Doesn't fully test everything

Files

- Bank.[h/cpp]
- Account.[h/cpp]
- Money.[h/cpp]
- Main.cpp
- **Makefile**

- Builds the example

Makefile

```
.PHONY: wipe build compile run clean

build: wipe bank_example.x

compile: wipe money.o account.o bank.o main.o

run: wipe bank_example.x
    ./bank_example.x

wipe:
    clear; clear

clean:
    rm -f account.o bank.o money.o main.o bank_example.x
```

Specifies the targets which are not actually files

Makefile

```
.PHONY: wipe build compile run clean

build: wipe bank_example.x

compile: wipe money.o account.o bank.o main.o

run: wipe bank_example.x
	./bank_example.x

wipe:
	clear; clear

clean:
	rm -f account.o bank.o money.o main.o bank_example.x
```

Default command
Builds everything

Makefile

```
.PHONY: wipe build compile run clean

build: wipe bank_example.x

compile: wipe money.o account.o bank.o main.o

run: wipe bank_example.x
    ./bank_example.x

wipe:
    clear; clear

clean:
    rm -f account.o bank.o money.o main.o bank_example.x
```

Only handles compilation

Makefile

```
.PHONY: wipe build compile run clean

build: wipe bank_example.x

compile: wipe money.o account.o bank.o main.o

run: wipe bank_example.x
    ./bank_example.x

wipe:
    clear; clear

clean:
    rm -f account.o bank.o money.o main.o bank_example.x
```

Runs the program

Makefile

```
.PHONY: wipe build compile run clean

build: wipe bank_example.x

compile: wipe money.o account.o bank.o main.o

run: wipe bank_example.x
	./bank_example.x

wipe:
	clear; clear

clean:
	rm -f account.o bank.o money.o main.o bank_example.x
```

Clears the screen
Makes debugging easier

Makefile

```
.PHONY: wipe build compile run clean

build: wipe bank_example.x

compile: wipe money.o account.o bank.o main.o

run: wipe bank_example.x
    ./bank_example.x

wipe:
    clear; clear

clean:
    rm -f account.o bank.o money.o main.o bank_example.x
```

Removes all dynamically generated files

Makefile

```
#Compile
money.o: money.h money.cpp
	g++ -c -o money.o money.cpp
account.o: money.h account.h account.cpp
	g++ -c -o account.o account.cpp
bank.o: money.h account.h bank.h bank.cpp
	g++ -c -o bank.o bank.cpp
main.o: money.h account.h bank.h main.cpp
	g++ -c -o main.o main.cpp

#Link
bank_example.x: money.o account.o bank.o main.o
	g++ -o bank_example.x money.o account.o bank.o main.o
```

Makefile Comments

Makefile

Create object files
Makes sure necessary files exist

```
#Compile
money.o: money.h money.cpp
	g++ -c -o money.o money.cpp
account.o: money.h account.h account.cpp
	g++ -c -o account.o account.cpp
bank.o: money.h account.h bank.h bank.cpp
	g++ -c -o bank.o bank.cpp
main.o: money.h account.h bank.h main.cpp
	g++ -c -o main.o main.cpp

#Link
bank_example.x: money.o account.o bank.o main.o
	g++ -o bank_example.x money.o account.o bank.o main.o
```

Makefile

```
#Compile
money.o: money.h money.cpp
	g++ -c -o money.o money.cpp
account.o: money.h account.h account.cpp
	g++ -c -o account.o account.cpp
bank.o: money.h account.h bank.h bank.cpp
	g++ -c -o bank.o bank.cpp
main.o: money.h account.h bank.h main.cpp
	g++ -c -o main.o main.cpp

#Link
bank_example.x: money.o account.o bank.o main.o
	g++ -o bank_example.x money.o account.o bank.o main.o
```

Create executable file
Needs to link together all object files

Header Files: bank.h

```
#ifndef __BANK_H
#define __BANK_H

#include <string.h>
#include "account.h"
#include "money.h"

class Bank
{
  public:
    Bank();
    ~Bank();

    int NewAccount(std::string name);
    Money AccountBalance(int id) const;
    Account* AccountData(int id) const;

    int Withdraw(int id, const Money &money);
    int Deposit(int id, const Money &money);

  private:
    Account **accounts;
    const static unsigned int MAX_ACCOUNTS = 10;
};

#endif
```

Header Guards

Constructor and Destructor

Resolution Operator

Const Member Functions

Passing Objects

Memory Allocation

Class Variables

Header Files: account.h

```
#ifndef __ACCOUNT_H
#define __ACCOUNT_H

#include <string>
#include "money.h"

class Account
{
  friend class Bank;

  public:
    Account(unsigned int identifier, std::string owner);
    ~Account();

    unsigned int Id() const;
    std::string Owner() const;
    Money AmountSaved() const;

  private:
    unsigned int id;
    std::string name;
    Money amountSaved;
};

#endif
```

Friend Classes

Header Files: money.h

```
#ifndef __MONEY_H
#define __MONEY_H

class Money
{
  friend void PrintDollars(const Money &m);   Friend Function

  public:         Default Constructor
    Money();
    Money(int amount);         Conversion Constructor
    Money(int dollars, int cents);
    ~Money();

    int Amount() const;
    Money operator+(const Money &money) const;
    Money operator-(const Money &money) const;   Operator Overloading
    Money operator+=(const Money &money);
    Money operator-=(const Money &money);

  private:
    int amt;
};

void PrintDollars(const Money &m);   Non-Member Function

#endif
```