Keyword: friend

Friend Classes

```
class Account
{
  friend class Bank;
  Public:
    unsigned int Id() const;
  private:
    unsigned int id;
};
```

- How does this affect Bank?

Friend Classes

```
class Account
{
  friend class Bank;
  Public:
    unsigned int Id() const;
  private:
    unsigned int id;
};
```

- How does this affect Bank?
  - Bank can now directly access the id variable
  - All other classes have to go through the Id() function

Friend Classes

```
Account* Bank::AccountData(int id) const
{
  int i;
  for (i = 0; i < Bank::MAX_ACCOUNTS; i++)
    if (accounts[i] != NULL && accounts[i]->id == id)
      break;

  //could not find
  if (i >= Bank::MAX_ACCOUNTS)
    return NULL;

  return accounts[i];
}
```

```
class Account
{
  friend class Bank;
  Public:
    unsigned int Id() const;
  private:
    unsigned int id;
};
```

Friend Classes

```
/* Main */
Account *account = bank.AccountData(accountId);
...
int id = account->Id();
```

```
class Account
{
  friend class Bank;
  Public:
    unsigned int Id() const;
  private:
    unsigned int id;
};
```

Friend Classes?

```
class Money
{
  public:
    int Amount() const;
  private:
    int amt;
};
```

- But what if we didn't want an entire class to have full access?

Friend Functions

```
class Money
{
  friend void PrintDollars(const Money &m);
  public:
    int Amount() const;
  private:
    int amt;
};
...
void PrintDollars(const Money &m);
```

- But what if we didn't want an entire class to have full access?
  - We can grant access to specific functions instead
  - Now only PrintDollars() can access the amt variable

Friend Functions

```
void PrintDollars(const Money &money)
{
  int amt = money.amt;

  if (amt < 0)
  {
    amt *= -1;
    std::cout << "-";
  }

  std::cout << "$" << amt / 100 << "." << amt % 100;
}
```

```
class Money
{
  friend void PrintDollars(const Money &m);
  public:
    int Amount() const;
  private:
    int amt;
};
...
void PrintDollars(const Money &m);
```