CODE FOR TODAY:
Github class-repo-public exercises

Design and Development of Classes

CS3081 Software Design and Development

The phrase “software design” means the conception, invention, or contrivance of a scheme for turning a specification for computer software into operational software. McConnel p. 74

What did you just say ?

Software design is how you get from Here to There.
(from Requirements to Code)

Design: Getting From Requirements to Code ...

- How do we get there?
- What are our end goals?
- What does good design look like?
- What impedes our path?

- OOP and Classes
  - Interfaces
  - Coupling
  - Cohesion
  - Stratification

- Creating and Using Classes in C++

Things to Think About

- Top-down and bottom-up design aren't competing strategies – they're mutually beneficial.
- There are two guaranteed ways to fail in design: design every last detail before coding and don't design anything at all.
- Find a way to "write" about your design – pictures or texts. Starting with a picture might be the best.

The Art of Design

DESIGN

- is a sloppy PROCESS.
- is individualistic.
- is a guessing game (unless you have perfect accuracy).
- emerges.
- requires tradeoffs and prioritizing.
- restricts possibilities.

When asked if he thought he wasted his time testing filament materials and finding no success, Edison said,

"Nonsense. I have discovered a thousand things that don't work."

The Enemy and How to Combat It

Good design and good development practices can:

Make Enhancement Easier
(simplify and isolate future modifications)

Minimize Fixes
(minimize error to minimize future modifications)

bug fixes, added features, requirement changes, new hardware, system migration, ...

Modifications are your enemy.
Modifications are necessary.

Why the Change ?

- Complex complexity complexes us all!
- To err is human.
- You are dealing with a Wicked Problem.
- You have many choices in life and in development.

How to Combat Change

- Test-Driven Development helps.
- OOP helps (a design element).
- Good design helps.

There's that word "good" again. What does that mean? What makes "good" hard to achieve?

How do I create large-scale, sustainable, maintainable, efficient, robust code that meets the client's requirements, under time and budget constraints?

Qualifiying "good" in design:

- GENERIC: minimal complexity, ease of maintenance, extensibility, reusability, and portability.
- SPECIFIC: performance quality, security, compactness, ...

The path to "good" is paved with OOP concepts:

- It is all about reuse, centralization, decomposition.
- Classes, classes, classes.

Design Decisions in C++

It is a class ...

- How big should it be?
- What data (members) and what functions (methods) should it include?

- What should the interface be like?
- What should I hide?

- What interactions should I have accross classes?
- If I combine classes, should I inherit or compose?

OOP is great, BUT you have to do it right!

Strive for these Core Characteristics of Classes:

- Consistent Abstraction
  - Allows for a consistent visualization of the system (stratification)
- Encapsulate Information and Hide Information
  - Hurts your brain less.
  - Easier to read (self-documenting)
  - Makes change easier (refactoring)
- Inherit (when it simplifies)
  - Capitalizes on re-use, less code, more abstraction.
- Identify and Isolate Areas Prone to Change
  - Design for change (if it is relatively easy)
- Loose Coupling Across Classes, Strong Cohesion Within

Classes in C++

- Define
  - Private, public, protected
  - Members and Methods
- Constructor
- Declare
- Destructor

The Public Interface and Private Implementation

```
class RobotClass {
private:
    int direction;
    int radius;
    int speed;
    int position[2];

    // helper function
    void calculatePosition();

public:

    // accessor functions
    void getPosition(int& pos);

    // mutator functions
    void setDirection(int inDir);
    void setRadius(int inRad);
    void setSpeed(int inSpeed);

    // public method (function)
    void draw();

};
```

```
#include "robotClass.h"

void RobotClass::getPosition(int& pos) {
    pos = position;
}

void Robotclass::setDirection(int inDir) {
    direction = inDir;
}
```

Helper Function
Functions can be private too.

Accessor / Mutator
The only way to look at and use private data.

Be Suspicious and Paranoid

Suspicious of *Friends*

```
class RobotClass {
public:
    // friends
    friend void noSecrets(RobotClass*);
```

```
// I'm a friend of RobotClass.
void noSecrets(RobotClass *robot) {

    // look what I can do ...
    robot->speed = 500000000000000000;
}
```

Instead of

```
robot->setSpeed(500000000000000);
```

Paranoid about Users

```
private:
    int direction;
    int radius;
    int speed;
    int position[2];
};
```

```
private:
    struct Cheshire;
    Cheshire *smile;
```

```
// I am hiding the implementation.
// Notice types are distinct from
//       how user is passing in values.
struct RobotClass::Cheshire {
    double direction;
    double radius;
    double speed;
    PosType position;
}
```

A Bad Interface

You are looking at the declaration (i.e. header file) and documentation of someone else's class. You don't know how to use it!

What do you do?

A) Look at the Implementation.

B) Contact the author to say "I'm not sure how to use your class."

Author response?

A) "Oh, this is how you use it. ... "

B) Modify it. Commit it. Ask user "Can you hear me now?"

Consistent Abstraction (Stratification)

Example 6-4.

```
class EmployeeCenus: public ListContainer {
public:
    ...
    // public routines
    void AddEmployee( Employee employee );          <-- 1
    void RemoveEmployee( Employee employee );          <-- 1

    Employee NextItemInList();          <-- 2
    Employee FirstItem();                 |
    Employee LastItem();          <-- 2
    ...
private:
    ...
};
```

(1) The abstraction of these routines is at the "employee" level.

(2) The abstraction of theseroutines is at the "list" level.

Example 6-6.

```
class Employee {
public:
    ...
    // public routines
    FullName GetName() const;
    Address GetAddress() const;
    PhoneNumber GetWorkPhone() const;
    ...
    bool IsJobClassificationValid( JobClassification jobClass );
    bool IsZipCodeValid( Address address );
    bool IsPhoneNumberValid( PhoneNumber phoneNumber );

    SqlQuery GetQueryToCreateNewEmployee() const;
    SqlQuery GetQueryToModifyEmployee() const;
    SqlQuery GetQueryToRetrieveEmployee() const;
    ...
private:
    ...
};
```

Strong Cohension

Example 6-2.

```
class Program {
public:
    ...
    // public routines
    void InitializeCommandStack();
    void PushCommand( Command command );
    Command PopCommand();
    void ShutdownCommandStack();
    void InitializeReportFormatting();
    void FormatReport( Report report );
    void PrintReport( Report report );
    void InitializeGlobalData();
    void ShutdownGlobalData();
    ...
private:
    ...
};
```

What's Wrong With This (Hint: Coupling)?

```
class CartEntry {
public:
    float Price;
    int Quantity;
}
```

```
class CartContents {
public:
    CartEntry[] items;
}
```

```
class Order {
private:
    CartContents cart;
    float salesTax;
public:
    Order(CartContents cart, float salesTax);
    float OrderTotal()
}
```

```
Order::Order(CartContents cart, float salesTax)
{
    this.cart = cart;
    this.salesTax = salesTax;
}

Order::float OrderTotal()
{
    float cartTotal = 0;
    for (int i = 0; i < cart.items.Length; i++)
    {
        cartTotal += cart.items[i].Price * cart.items[i].Quantity;
    }
    cartTotal += cartTotal*salesTax;
    return cartTotal;
}
```

Group Time

- What is wrong with this from a design perspective?
- What problems does it cause?
- How can you fix it?

http://stackoverflow.com/questions/226977/what-is-loose-coupling-please-provide-examples