类设计与实现举例

Designing Classes: From Requirements to Implementation

Choosing classes and member functions is a difficult design process. It's very hard to choose the right number of classes with the right member functions so that the classes are cohesive, loosely coupled, easy to modify, and yield an elegant and correctly working program. Many programmers and computer scientists are good algorithmic thinkers but bad class designers, and vice versa. You shouldn't expect to become an accomplished designer early in your studies, but you can build expertise by studying other designs, and by modifying existing designs before creating your own. In this way you'll learn as an apprentice learns any craft.[1]

Requirement

The **requirements** of a problem or programming task are the constraints and demands asked by the person or group requesting a programming solution to a problem. As a designer/programmer your task in determining requirements is to interact with the user

We want to develop a quiz program that will permit different kinds of questions; that is not just different kinds of arithmetic problems, but questions about state capitals, English literature, rock and roll songs, or whatever you think would be fun or instructive. We'd like the program to be able to give a quiz to more than one student at the same time, so that two people sharing a keyboard at one computer could both participate. If possible, we'd like to allow a student to have more than one chance at a question.

Requirement

1. More than one kind of question can be used in a quiz.
2. Several students can take a quiz sharing a keyboard.
3. Students may be allowed more than one chance to answer a question.

What classes?

- Nouns as classes
  - quiz, question, problem, student, computer, keyboard, chance

> We want to develop a quiz program that will permit different kinds of questions; that is not just different kinds of arithmetic problems, but questions about state capitals, English literature, rock and roll songs, or whatever you think would be fun or instructive. We'd like the program to be able to give a quiz to more than one student at the same time, so that two people sharing a keyboard at one computer could both participate. If possible, we'd like to allow a student to have more than one chance at a question.

Member Functions

- Verbs as Member Functions (Methods)
  - Class behavior
    - Public member functions
  - Responsibility
    - Methods associated with the class and the interactions between classes

Sometimes candidate class methods can be found as verbs in a specification. Often, however, you'll need to anticipate how a program and classes are used to find methods.

Finding Verbs Using Scenarios

- Two students sit at a keyboard, each is asked to enter her name, then a quiz is given and students alternate providing answers to questions.
- When a quiz is given, the student determines the number of questions that will be asked before the quiz starts. If two people are taking a quiz together, both are asked the same number of questions.
- Students have two chances to respond to a question. A simple "correct" or "in-correct" is given as feedback to each student response. If a student doesn't type a correct response, the correct answer is given.
- At the end of a quiz, each student taking the quiz is given a score.

EnterName, ChooseNumberOfQuestions, ChooseKindOfQuestion, RespondToQuestion, GetCorrectAnswer, GetScore, AskQuestion, ProvideFeedback

Which of these verbs goes with which class?

Program Tip 7.1: Concentrate on behavior rather than on state in initial class design. You can change how a class is implemented without affecting client programs that use the class, but you cannot change the member functions (e.g., the parameters used) without affecting client programs.

Assigning Responsibilities

- Not all responsibilities will be assigned to a class
  - **Student**
    - Construct using name (ask for name in `main`)
    - RespondTo a question
    - GetScore
    - GetName (not in scenario, but useful accessor)
  - **Quiz**
    - ChooseKindOfQuestion
    - AskQuestion of/GiveQuestion to a student
  - **Question**
    - Create/Construct question type
    - AskQuestion
    - GetCorrectAnswer

Implementing and Testing Classes

Program Tip 7.2: One cornerstone of iterative enhancement is adding code to a working program. This means that the software program grows, it doesn't spring forth fully functional. The idea is that it's easier to test small pieces and add functionality to an already tested program, than it is to test many methods or a large program at once.

already imple
method probal
to. If we don'
fully functiona

Program 7.1 studentstub.cpp

```
void Student::RespondTo( missing Question parameter )
{
    string answer;
    cout << endl << "type answer after question " << endl;
    cout << "what is your favorite color? ";
    cin >> answer;
    if (answer == "blue")
    {   cout << "that is correct" << endl;
    myCorrect++;
    }
    else
    {   cout << "No! your favorite color is blue" << endl;
    }
}
```

We could use this stub function to test the
and Studer

```
#include <iostream>
#include <string>
using namespace std;
#include "student.h"
#include "prompt.h"

int main()
{
    string name = Prompt
    int numQuest = Prom
    Student st(name);
    int k;
    for(k=0; k < numQuest; k++)
    {   st.RespondTo();   // question parameter missing
    }
    cout << st.Name() << ", your score is "
         << st.Score() << " out of " << numQuest << endl;
    return 0;
}
```

OUTPUT

```
enter name: Owen
number of questions:  between 1 and 10: 3

type answer after question
what is your favorite color? red
No! your favorite color is blue

type answer after question
what is your favorite color? blue
that is correct

type answer after question
what is your favorite color? yellow
No! your favorite color is blue
Owen, your score is 1 out of 3
```

Implementing the Class Quiz

- [John Bentley 1988]

Program Tip 7.3: Always do the hard part first. If the hard part is impossible, why waste time on the easy part? Once the hard part is done, you're home free.

Program Tip 7.4: Always do the easy part first. What you think at first is the easy part often turns out to be the hard part. Once the easy part is done, you can concentrate all your efforts on the hard part.

- **Quiz**
  - ChooseKindOfQuestion
  - AskQuestion of/GiveQuestion to a student

- how the Quiz knows which student to ask?

1. A `Quiz` object knows about all the students and asks the appropriate student. In this case all `Student` objects would be private data in the `Quiz` class, created by the `Quiz`.
2. The student of whom a question will be asked is passed as an argument to the `Quiz::GiveQuestionTo` member function. In this case the `Student` object is created somewhere like `main` and passed to a `Quiz`.
3. The student is created in the function `Quiz::GiveQuestionTo` and then asked a question.

- It corresponds to how Student objects are defined and used

1. As instance variables of the class `Quiz` since private data is global to all `Quiz` methods, so is accessible in `Quiz::GiveQuestionTo`.
2. As parameter(s) to `Quiz::GiveQuestionTo`. Parameters are accessible in the function to which they're passed.
3. As local variables in `Quiz::GiveQuestionTo` since local variables defined in a function are accessible in the function.

- Code of giving a quiz to two students

```
int main()
{
    Student owen("Owen");
    Student susan("Susan");
    Quiz q;
    q.GiveQuestionTo(owen);
    q.GiveQuestionTo(susan);

    cout << owen.Name()  << " score = "
         << owen.Score() << endl;
    cout << susan.Name() << " score = "
         << susan.Score() << endl;
    return 0;
}
```

- design the function Quiz::GiveQuestionTo()

Program 7.3 quizstub.cpp

```
void Quiz::GiveQuestionTo(Student & s)
// postcondition: student s asked a question
{
  cout << endl << "Ok, " << s.Name() << " it's your turn" << endl;
  cout << "type answer after question " << endl;

  myQuestion.Create();
  if (! s.RespondTo(myQuestion))
  { cout << "try one more time" << endl;
    if (! s.RespondTo(myQuestion))
    {  cout << "correct answer is " << myQuestion.Answer() << endl;
    }
  }
}
```

Implementing the Class Question

Program 7.4 question.h

```
#include <iostream>
#include <string>
using namespace std;

// simple Question class

class Question
{
  public:
    Question()
    {   // nothing to ini
    }
    void Create()
    {   // the same quest
    }
    void Ask()
    {   cout << "what is your favorite color? ";
    }
    string Answer() const
    {   return "blue";
    }
};
```

```
void Student::RespondTo(Question & q)
{
    string answer;
    cout << endl << "type answer after question " << endl;
    q.Ask();
    cin >> answer;

    if (answer == q.Answer())
    {   cout << "that is correct" << endl;
        myCorrect++;
    }
    else
    {   cout << "No! your favorite color is "
             << q.Answer() << endl;
    }
}
```

Reference

- http://www.cs.duke.edu/csed/tapestry/final7.pdf

Class Interfaces, Design, and Implementation 7

There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.
Fred P. Brooks
No Silver Bullet — Essence and Accident in Software Engineering

One tenet of object-oriented programming is that a library of well-designed classes makes it easier to design and write programs. The acronym *COTS*, for *commercial, off-the shelf* software, is often used to identify the process of reusing (commercial) libraries of classes and code. Classes are often easier to reuse in programs than non-class code. For example, we've used the `Dice` and `RandGen` classes and the functions from `"prompt.h"` in many programs. In almost every program we've used `cout` and `cin`, objects that are

[1]Programming is both an art and a science. To some, it's only a science or only an art/craft. In my view there are elements of both in becoming an accomplished programmer. You must understand science and mathematics, but good design is not solely a scientific enterprise.

7.1 Designing Classes: From Requirements to Implementation

Choosing classes and member functions is a difficult design process. It's very hard to choose the right number of classes with the right member functions so that the classes are cohesive, loosely coupled, easy to modify, and yield an elegant and correctly working program. Many programmers and computer scientists are good algorithmic thinkers but bad class designers, and vice versa. You shouldn't expect to become an accomplished designer early in your studies, but you can build expertise by studying other designs, and by modifying existing designs before creating your own. In this way you'll learn as an apprentice learns any craft.[1]

[1]Programming is both an art and a science. To some, it's only a science or only an art/craft. In my view there are elements of both in becoming an accomplished programmer. You must understand science and mathematics, but good design is not solely a scientific enterprise.