The Object-Oriented Thought Process

Chapter 06

Designing with Objects

Design Guidelines

Generally, a solid OO design process includes the following steps:

1. Doing the proper analysis
2. Developing a statement of work that describes the system
3. Gathering the requirements from this statement of work
4. Developing a prototype for the user interface
5. Identifying the classes
6. Determining the responsibilities of each class
7. Determining how the various classes interact with each other
8. Creating a high-level model that describes the system

The Ongoing Design Process

Despite the best intentions and planning, in all but the most trivial cases, the design is an ongoing process.

- Even after a product is in testing, design changes will pop up.
- It is up to the project manager to draw the line that says when to stop changing a product and adding features.

Requirements

The reasons to identify requirements early and keep design changes to a minimum are as follows:

- The cost of a requirement/design change in the design phase is relatively small.
- The cost of a design change in the implementation phase is significantly higher.
- The cost of a design change after the deployment phase is astronomical when compared to the first item.

Safety vs. Economics

Would you want to cross a bridge that has not been inspected and tested?

- Unfortunately, with many software packages, users are left with the responsibility of doing much of the testing.
- This is very costly for both the users and the software providers.
- Unfortunately, short-term economics often seem to be the primary factor in making project decisions.

Performing the Proper Analysis

In the analysis phase, the users and the developers must do the proper research and analysis to determine the statement of work, the requirements of the project, and whether to actually do the project.

- Most of these practices are not specific to OO.
  - They apply to software development in general.

Developing a Statement of Work

The statement of work (SOW) is a document that describes the system.

- The SOW contains everything that must be known about the system.
- Many customers create a request for proposal (RFP) for distribution, which is similar to the statement of work.

Gathering the Requirements

The requirements document describes what the users want the system to do.

- Even though the level of detail of the requirements document does not need to be of a highly technical nature, the requirements must be specific enough to represent the true nature of the user's needs for the end product.
  - Whereas the SOW is a document written in paragraph (even narrative) form, the requirements are usually represented as a summary statement or presented as bulleted items.

Developing a Prototype of the User Interface

A prototype can be just about anything; however, most people consider the prototype to be a simulated user interface.

- By creating actual screens and screen flows, it is easier for people to get an idea of what they will be working with and what the system will feel like.
- In any event, a prototype will almost certainly not contain all the functionality of the final system.

Identifying the Classes

After the requirements are documented, the process of identifying classes can begin.

- Don’t be too fussy about getting all the classes right the first time.
- You might end up eliminating classes, adding classes, and changing classes at various stages throughout the design.
  - Take advantage of the fact that the design is an iterative process. ages throughout the design.

Determining the Responsibilities of Each Class

You need to determine the responsibilities of each class you have identified.

- This includes the data that the class must store and what operations the class must perform.

Determining How the Classes Collaborate with Each Other

Most classes do not exist in isolation.

- Although a class must fulfill certain responsibilities, many times it will have to interact with another class to get something it wants.
- One class can send a message to another class when it needs information from that class, or if it wants the other class to do something for it.

Creating a Class Model to Describe the System

When all the classes are determined and the class responsibilities and collaborations are listed, a class model that represents the complete system can be constructed.

- The class model shows how the various classes interact within the system.

Prototyping the User Interface

During the design process, we must create a prototype of our user interface.

- This prototype will provide invaluable information to help navigate through the iterations of the design process.
- However you develop the user interface prototype, make sure that the users have the final say on the look and feel.

Object Wrappers

When you write a program that uses an object-oriented programming language and are using sound object- oriented design techniques, you are also using structured programming techniques.

- Wrappers are used to encapsulate code within objects (including structured code).

Structured Code

- While this code is written in an object-oriented language, the code inside of the main method is structured code.
  - All three basics of structured programming are present: *sequence, conditions, and iterations.*

```
public static void main(String args[]) {
  int x = 0;
  while (x <= 10) {
    if (x==5) System.out.println("x = " + x);
    x++;
  }
}
```

Wrapping Structured Code

As you can see, the structured code used to perform the addition (*a + b) is wrapped inside the* add method.

– Although this is a trivial example, that is all there is to wrapping structured code.

```
class SomeMath {
  public int add(int a, int b) {
    return a + b;
  }
}
```

Wrapping Nonportable Code

Rather than making the programmer memorize the code (or look it up), you can provide a class called Sound that contains a method called beep as shown next:

```
class Sound  {
  public void beep() {
    System.out.println(“\007”);
  }
}
```

Wrapping Existing Classes

Software developers often utilize code written by someone else.

- Perhaps the code was purchased from a vendor or even written internally within the same organization.
- In this case, legacy code can be wrapped inside new classes and methods.