Advanced Object-Oriented Concepts:

Constructors

Constructors are methods that share the same name as the class.

A constructor does not have a return value, if you provide a return value, the compiler **will not** treat the method as a constructor.

A constructor is called when a new object is created. The "**new**" keyword creates a new instance of the class, thus allocating the required memory. The constructor provides the developer the opportunity to attend to the appropriate initialization.

The most important function of a constructor is to initialize the memory allocated when the new keyword is encountered. Code included inside inside a constrictor should set the newly created object to its initial, stable, safe state.

For example, if you have a counter object with an attribute called count, you need to set count to zero in the constructor.

A constructor-less class will compile. If a class does not have a constructor, a default constructor will be provided. One constructor will always exist, regardless of whether you write one.

Besides the creation of the object itself, the only action that a default constructor takes is to call the constructor of its superclass.

It is a good programming practice to always include at least one constructor in a class.

In many cases, an object can be constructed in more than one way.

Overloading Methods

Overloading allows a programmer to use the same method name over and over, as long as the signature of the method is different each time.

By using different signatures, you can construct objects differently depending on the constructor used. This functionality is very helpful when you don't always know ahead of time how much information you have available.

For example, when creating a shopping cart, customers may already be logged in to their account (and you will have all of their information). On the other hand, a totally new customer may be placing items in the cart with no account information available at all. In each case, the constructor would initialize differently.

How the Superclass is Constructed

When using inheritance, you must know how the parent class is constructed. Remember that when you use inheritance, you are inheriting everything about the parent. Thus, you must become entirely aware of all the parent's data and behavior

The Design of Constructors

When designing a class, it is good practice to initialize all the attributes. In java, you cannot use the attributes until it is initialized.

Constructors are used to ensure that the application is in a stable state (or safe state)

During the design, it is good practice to identify a stable state for all attributes and then initialize them to a this stable state in the constructor.

Error Handling

There are three basic solutions to handling problems that are detected in a program: fix it, ignore the problem by squelching it, or exit the runtime in some graceful manner.

Ignoring a Problem

You should not ignore any known problem. The primary directive for all applications is that the application should never crash. If you do not handle your errors, the application will eventually terminate ungracefully or continue in a mode that can be considered an unstable state.

Checking for Problems and Aborting the Application

Checking for potential problems and aborting the application allow the system to clean up things and put itself in a more stable state, such as closing files and forcing a system restart.

Checking for Problems and Attempting to Recover

Checking for potential problems, catching the mistake, and attempting to recover is a far superior solution than simply checking for problems and aborting. In this case, the problem is detected by the code, and the application attempts to fix itself.

It is not always easy to determine where a problem first appears. And it might take a while for the problem to be detected. It is important to design error handling into the class right from the start, and often the operating system itself can alert you to problems that it detects.

Throwing an Exception

Exceptions are unexpected events that occur within a system. Exceptions provide a way to detect problems and handle them. In Java, C#, C++, Objective-C, and Visual Basic, exceptions are handled by the keyword '**catch**' and '**throw**'.

If a exception is thrown within the '**try**' block, the '**catch**' block will handle it.

It's a good idea to use a combination of the methods described here to make your program as bulletproof to your user as possible.

The Importance of Scope

Multiple objects can be instantiated from a single class. Each of these objects has a unique identity ad state. Each object is constructed separately and is allocated its own separate memory. However, some attributes and methods may if properly declared, be shared by all the objects instantiated from the same class, thus sharing the memory allocated for these class attributes and methods.

A constructor is a good example of a method that is shared by all instances of a class.

Object Attributes

In many design situations, an attribute must be shared by several methods within the same object.

The keyword '**this**' is a reference to the current object

Operator Overloading

Operator overloading allows you to change the meaning of an operator

In the context of strings, the plus sign does not mean addition of integers or floats, but concatenation of strings.

Although these languages (Java, .NET, and Objective-C) do not allow the option of overloading operators, the languages themselves do overload the plus sign for string concatenation. The designers of Java must have decided that operator overloading was more of a problem than it was worth.

Multiple Inheritance

Multiple inheritance allows a class to inherit from more than one class.

Multiple inheritance is a very powerful technique, and in fact, some problems are quite difficult to solve without it. Multiple inheritance can even solve some problems quite elegantly. However, multiple inheritance can significantly increase the complexity of a system, both for the programmer and the compiler writers.

Multiple inheritance does not exist in Java, .NET, and Objective-C since the complexity of it outweigh it's advantages.

Deep Versus Shallow Copies

A deep copy occurs when all the references are followed and new copies are created for all referenced objects.

A shallow copy would simply copy the reference and now follow the levels.