Software Design & Architecture

Categories of Objects in Design

In the last video, Sam discussed the idea of breaking down objects, or splitting them into smaller objects. As you are breaking down objects, you may find that you will identify different kinds of objects. Many developers have noticed the same thing, and there are generally three categories of objects.

Entity Objects

Entity objects are the most familiar, because they correspond to some real-world entity in the problem space. If you have an object representing a chair in your software, then this is an entity object. If you have an object representing a building or a customer, these are all entity objects. Generally, these objects will know attributes about themselves. They will also be able to modify themselves, and have some rules for how to do so.

When you are identifying objects to include in your software and breaking down those objects into smaller objects, you will initially get entity objects. The other categories of objects will come later, as you start to think about the technical design of the software.

Boundary Objects

Boundary objects are objects which sit at the boundary between systems. This could be an object that deals with another software system - like an object that obtains information from the Internet. It could also be an object with the responsibility of showing information to the user and getting their input. If you program a user interface - the visual aspect of software - you are probably mostly working with boundary objects. Any object that deals with another system - a user, another software system, the Internet - can be considered a boundary object.

UNIVERSITY OF ALBERTA
FACULTY OF SCIENCE

Control Objects

Control objects are objects which are responsible for coordination. You will discover control objects when you attempt to break down a large object, and find that it would be useful to have an object that controls the other objects. You will see many examples of these objects in real usage in the next course in the specialization: Design Patterns. A great example is a Mediator: it simply coordinates the activities of many different objects so that they can stay loosely coupled.

Conclusion

At this point it may be difficult to see how these object types can help you. That is okay; breaking down objects in the best way takes real-world practice and experience. The most important thing to realize at this point is that your software will not solely consist of entity objects. Of course there will be objects for real-world items like tables and chairs or invoices and shopping carts, but there must also be objects for coordination and for interfacing with outside systems. They are a little bit harder to see, but no less essential, especially as you move from small projects to more complex software.

Organizing your software into entity objects, boundary objects, and control objects will allow your code to be more flexible, reusable, and maintainable. Try to start using them today to see their power!