Level up: Classes and objects

The following code is taken from the Roll-a-ball tutorial[1] provided with Unity.

Object-oriented programming is different from procedural programming. In normal procedural programming, you write a list of actions that are performed one after the other (in sequence). With object-oriented programming, the programming is defined by sets of objects that have particular properties that relate to each other in different ways.

For example, an object might be a person. That person has particular properties, such as a name, an address, a hair colour, a weight and a height. Another object might be a chair. It has particular properties such as colour, materials and strength. The person relates to the chair by sitting on it. The chair relates to the person by holding that person up.

The beauty of object-oriented programming is in three main concepts: encapsulation, inheritance and abstraction.

Concept	Explanation	Example	Advantages
Encapsulation			
Inheritance			
Abstraction			

[1] https://unity3d.com/learn/tutorials/projects/roll-ball-tutorial/moving-camera?playlist=17141

Classes and objects: Answers

Concept	Explanation	Example
Encapsulation	An object contains properties that are relevant to that object. These objects can be fully contained within their class. So, variables can be declared private and relevant only to that object.	In the program that we are working on, we create an object called PlayerController. This object contains variables that are only relevant and needed within the player controller. For example, it creates a local object of type RigidBody (rb) that is only accessed from within the player controller. If this was accessible outside of the object, then objects might interact with each other in odd ways. The main advantage of this though, is that if the code that the object needs stays within the object, then you can reuse the object multiple times with different uses
Inheritance	An object can be created that is a sub-object of another object, inheriting all of the properties of that object. You can then change particular properties.	An object called Animal might be created that has certain properties (is alive, needs sleep, needs to eat). Then, a FourLeggedAnimal object could be created that is a type of animal. This means that it would inherit all of the Animal properties, but might have its own properties as well (has four legs). A dog object may then be created that has the properties of all of its parent objects (inherits all of the properties of FourLeggedAnimal and that of Animal)
Abstraction	You can create an object of any type; for example, an object doesn't have to be a data type, it creates its own special data type.	In the program we are working on, we have created many different objects that are different types. For example: Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical); This helps you; for example, in game design you might create an enemy object that you could clone by creating different instances of that object.