- [Classes](https://www.codecademy.com/resources/docs/c-sharp/classes) are **reference types**. That means that when we create a new instance of a class and store it in a variable, the variable is a **reference** to the object.



- A new Encyclopedia instance is constructed and stored in the computer’s memory. We can imagine a slot in our computer holding the instance’s type, property values, etc. enc1 refers to that location in memory.  
- enc1 is NOT the actual object — it is a **reference** to the object. An object can have multiple references:

A black background with white text

Description automatically generated

- enc1 and enc2 are 2 references to the same location in memory: we can say that enc1 and enc2 refer to the same object.

A computer screen shot of a program code

Description automatically generated

**Value Types:**

- While reference-type [variables](https://www.codecademy.com/resources/docs/c-sharp/variables) refer to a place in memory, value-type variables hold the actual data.  
- int is a value type, so the variable num holds the value 6:



- Because value types hold the data itself, if we create a variable of a value type and set it equal to another existing variable of the same type, the value will be *copied* into a new memory address and stored in the new variable. The 2 variables will not be linked as with reference types:

A computer screen shot of numbers and equations

Description automatically generated

- A reference is like directions to a house, which “points” to the location but isn’t the actual house. A value type is the house itself!

**Comparisons:**

- When we compare value types with ==, the C# compiler performs a **value** comparison. For example, this prints true because the value 6 is equal to the value 6:

A computer screen shot of a code

Description automatically generated

- When we compare reference types with ==, the C# compiler performs a **referential** comparison, which means it checks if two [variables](https://www.codecademy.com/resources/docs/c-sharp/variables) refer to the same memory location   
- For example, this prints false:

A computer code with white and orange text

Description automatically generated

- We would need to assign one as a pointer to another for them to be considered equal:

A computer code with white and orange text

Description automatically generated

**References of Different Types:**

- In addition to referencing an instance with its class name, it can also be referenced using the name of any type in its [inheritance](https://www.codecademy.com/resources/docs/c-sharp/inheritance) hierarchy or of any interface it implements. Encyclopedia inherits from the base class Book and implements the interface IFlippable.

A computer screen shot of a code

Description automatically generated

- This means we can reference Encyclopedia objects as either Book or IFlippable:

A black background with white text

Description automatically generated

- Now enc, fEnc, and bEnc all refer to the same object.  
- A reference only has access to members of the type it is defined as. fEnc can only access members in the IFlippable interface, and bEnc can only access members in the Book superclass.

**Arrays of References:**

- Imagine we wanted to Flip() each object from a group of Encyclopedia and Novel types:

A computer screen with text and numbers

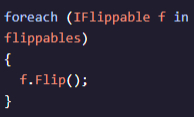
Description automatically generated

- The preceding code works, but it would be faster and safer to store the references in an array and loop through them. But would it be an array of Novel[], an array of Encyclopedia[], or something else?  
- We can create references to Encyclopedias and Novels as IFlippables. Instead of dealing with individual [variables](https://www.codecademy.com/resources/docs/c-sharp/variables), we can use an array of IFlippable references:

A screen shot of a computer code

Description automatically generated

- To Flip() each element, we can write a foreach loop:



- We can only access the functionality defined in the interface. For example, we couldn’t access f.Title because Title isn’t a property defined in IFlippable.