Module 1 Day 6

Introduction to Objects

What makes an application?

- Program Data
  - ✓ Variables & .NET Data Types
  - ✓ Arrays
  - ❑ More Collections (list, dictionary, stack, queue)
  - ❖ Classes and objects (OOP)
- Program Logic
  - ✓ Statements and expressions
  - ✓ Conditional logic (if)
  - ✓ Repeating logic (for, foreach, do, while)
  - ✓ Methods (functions / procedures)
  - ❖ Classes and objects (OOP)
  - ❑ Frameworks (MVC)
- Input / Output
  - User
    - ✓ Console read / write
    - ❑ HTML / CSS
    - ❑ Front-end frameworks (HTML / CSS / JavaScript)
  - Storage
    - ❑ File I/O
    - ❑ Relational database
    - ❑ APIs

Abstraction

- From dictionary.com (abstract):
  - Adj., thought of apart from concrete realities, specific objects, or actual instances: *an abstract idea.*
  - Noun, a summary of a text, scientific article, document, speech, etc.; *epitome*.
  - Noun, something that concentrates the essential qualities of anything more extensive or more general, or of several things; *essence*.
- The essence of abstraction is preserving information that is relevant **in a given context**, and forgetting information that is irrelevant in that context

Abstraction, cont’d

- Closely related to Modelling
  - any model is an abstraction (CAD, blueprint, model airplane, model railroad)
- Sometimes referred to as one of the pillars of OO
  - Encapsulation, Inheritance, Polymorphism
  - Really, it’s a feature of ALL programming
- Allows us the “think at a higher level”
  - Think about our use of *Console* so far in this course

Object-Oriented Programming

- Objects are a further level of abstraction
- Combine Data (variables) and Behavior (logic / flow) into an abstraction of a real-world “thing”
- e.g., Car
  - Data (State) - describes it - adjectives
    - Make, model, color, Engine State, Gear
  - Behavior – what it can do - verbs
    - Start, Change Gear, Speed Up, Slow Down, Turn
- e.g., Contact
  - Data
    - First Name, Last Name, Birthday, Email Address, Phone
  - Behavior
    - Send Mail, Call, Text

Class

- Until now, we've used Data Types available to us
  - int, double, string, Console, Array
- Now we are going to write our own Data Types
  - These are called Classes in OO parlance
  - Classes and Types are synonymous in C#/.NET

```
// Data Type to represent a person's contact information
class Contact
{
    public string FirstName;
    public string LastName;
    public DateTime BirthDate;
    public string EmailAddress;
    public string PhoneNumber;
}
```

Creating Objects from Classes

- Remember that a Class is just another word for a Type
- To use a type we Declare, Allocate and Assign (as always)

```
// **Declare** a place to hold a Contact **object**
Contact contact;

// **Allocate** memory to hold a contact and assign default values
contact = new Contact();

// **Assign** new data into the contact's **instance variables**
contact.FirstName = "Mike";
contact.LastName = "Morel";
```

- The “contact” variable holds an ***instance of*** the Contact class, also known as an ***object***

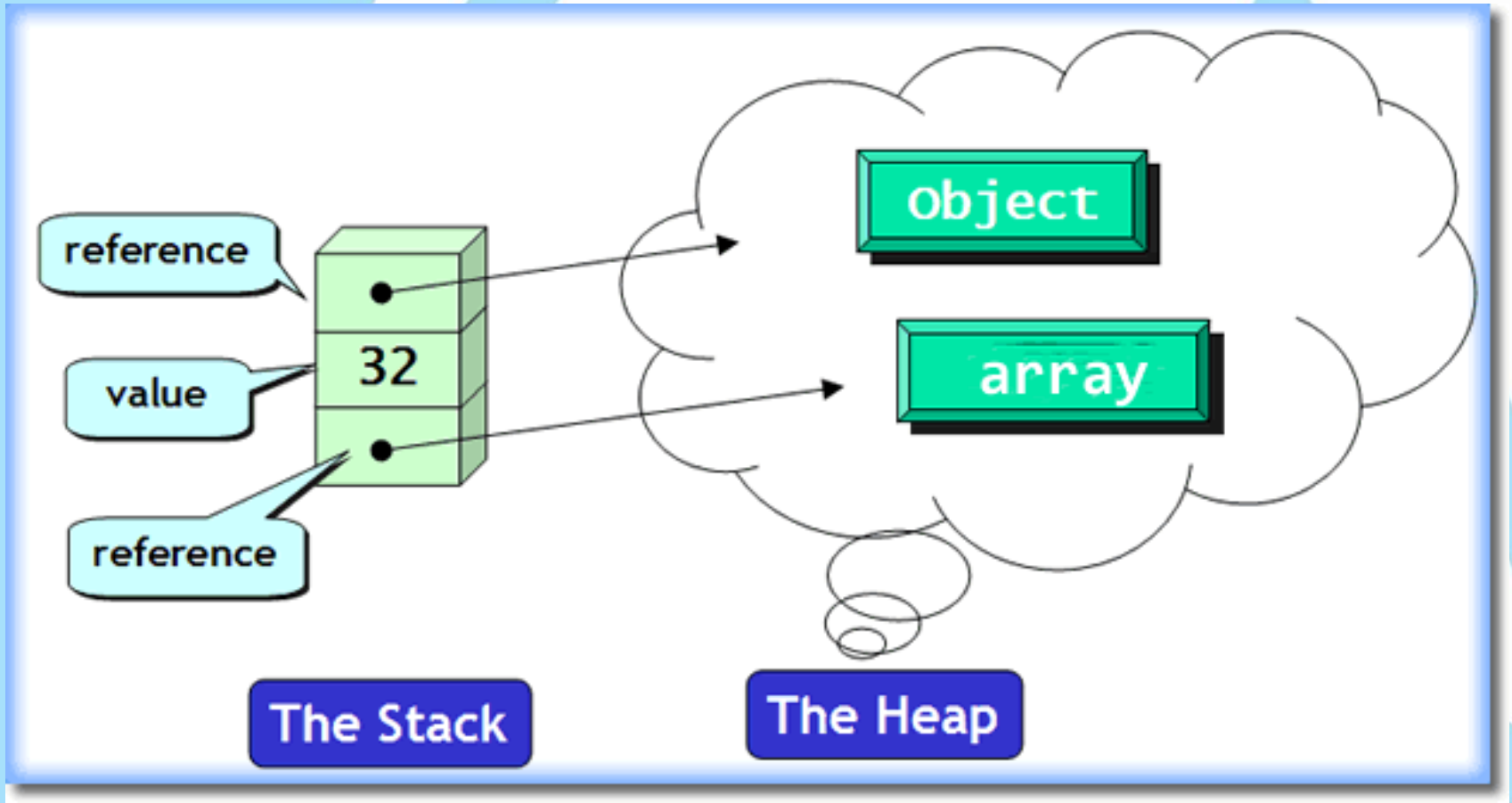
Class vs. Object

Class (Type)	Object (Instance)
int	3 21
string	"The quick brown fox" "Ohio"
int[]	{1,4,7,12,3} {10,20}
Car	1965 Ford Mustang with VIN 12345 2015 Toyota Corolla with VIN 99988
Person	Mike Morel, mike@techelevator.com

Stack and Heap Memory

- Stack Page
  - Fixed memory allocation (size known at compile-time)
  - Created when a method is invoked, destroyed when the method exits
  - Fast access
  - Runtime maintains a Stack (LIFO) of these as your program runs
  - C# Value types are stored here
- Heap
  - Dynamic memory allocation (determined at run-time)
  - Global in scope
  - Slower access as it can fragment
  - C# Reference types are allocated (**new**) here, and their address stored in Stack memory
- Assignment (=), Comparison (==) and parameter-passing all work on Stack memory!!!

Stack and Heap Memory



Strings

- String is a reference type:
  - Memory is allocated on the Heap
  - Address of memory is placed into stack variable
- However, string is “special” in a couple ways
  - You don’t have to use new() to allocate string memory (" does it)
  - Strings are **immutable** (you can't change it in place)
  - You can compare strings using ==
  - Sometimes you'll hear "C# strings have 'value' semantics"

String Methods

- Length (Property)
- Substring
- Contains
- StartsWith / EndsWith
- IndexOf
- Replace
- ToUpper / ToLower
- Split / Join
- Trim