ECE 217:
Data Structure and Algorithm

Lecture 2: Data Structures, Objects, and Classes

Instructor: Shayan (Sean) Taheri, Ph.D.

Assistant Professor

The Department of Electrical and Cyber Engineering (ECE)

The Institute for Health and Cyber Knowledge (I-HACK)

The Gannon University (GU)

Personal Information

- Name: Shayan (Sean) Taheri.
- Date of Birth: July/28/1991.
- Past Position: Postdoctoral Fellow at University of Florida.
- Ph.D. Degree: Electrical Engineering from the University of Central Florida.
- M.S. Degree: Computer Engineering from the Utah State University.
- University Profile: https://www.gannon.edu/FacultyProfiles.aspx?profile=taheri001

Integrated Development Environment

- When you are in the process of software development, you as a developer have to handle different issues and check everything at once.
  - *Many tools can help software engineers in their hard professional lives.*
  - *The most useful tool for software programming is an integrated development environment (IDE).*
- **IDE** is a toolkit or an application suite made up of basic tools.
  - *These tools help software developers get their work was done quickly and mistake-free.*
  - *The right IDEs are critical in the software development lifecycle.*

Integrated Development Environment (Contd.)

- With an IDE, you can write, check, and accelerate all frequently-performed actions as you are programming:
  - *Automation of tasks*
  - *Optimization of work*
  - *Higher efficiency*
  - *Less time spent on development*
  - *Higher satisfaction with work and with final results*
- IDEs display an application's structure as you are working on it.
- They let you search for pieces of code amongst massive amounts of data, and there is nothing better when you're trying to get rid of any bugs.
- <u>Example IDEs</u>: Visual Studio, IntelliJ IDEA, Xcode, Android Studio, AWS Cloud9 IDE, Eclipse, Zend Studio, PhpStorm, and Arduino IDE.

Objects, Structures, and Classes

- **Object**: It is a basic unit of Object-Oriented Programming and represents real life entities.
  - A typical high-level program creates many objects, which as you know, interact by invoking methods.
  - An object is an entity that combines data with behavior that acts on that data.
  - An object consists of :
    - ✓ ***State***: It is represented by attributes of an object. It also reflects the properties of an object.
    - ✓ ***Behavior***: It is represented by methods of an object. It also reflects the response of an object with other objects.
    - ✓ ***Identity***: It gives a unique name to an object and enables one object to interact with other objects.

Objects, Structures, and Classes (Contd.)

- Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, and “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

Identity	**State/Attributes**	**Behaviors**
Name of dog	*Breed*	*Bark*
	Age	*Sleep*
	Color	*Eat*

- **Structure**: A structure is a user-defined type that is used to store multiple types of data.
  - Structures, or ***structs***, are used to programmatically represent a real-life object in code.
  - Structures are created with the ***struct*** keyword followed by its name and then body containing its properties and methods.

Objects, Structures, and Classes (Contd.)

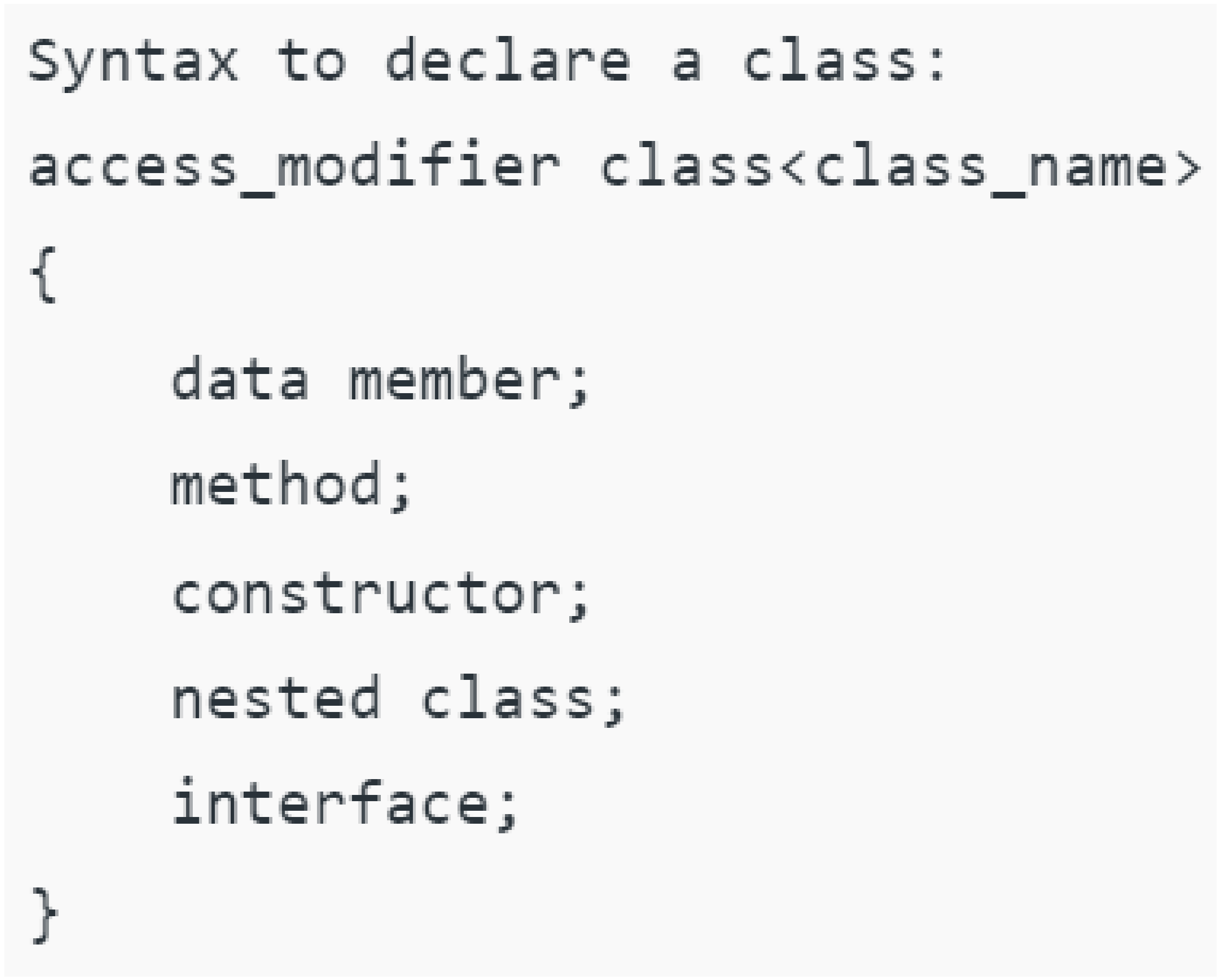
- **Structure**:
  - It groups together multiple data of different types and forms a single data type from those.
  - It is generally considered a variable.
  - It is a set of elements of any type (except for the **void** type), so causes combining logically related data of different types.
  - The structure name can't be used as an identifier (name of a variable or function).
  - Its common description is:

```
struct structure_name
   {
    elements_description
   };
```

Objects, Structures, and Classes (Contd.)

- **Class**: A class is the building block of an Object-Oriented programming language.
  - Class is a set of object which shares common characteristics/behavior and common properties/attributes.
  - Class is not a real world entity. It is just a template or blueprint or prototype from which objects are created.
  - It is a user-defined data type that holds its own data members and member functions that can be accessed and used by creating an instance of that class.
  - It represents the set of properties or methods that are common to all objects of one type.
  - Class does not occupy memory.
  - Class is a group of variables of different data types and group of methods.
  - A class in can contain: ***data member, method, constructor, nested class, and interface***.

Objects, Structures, and Classes (Contd.)



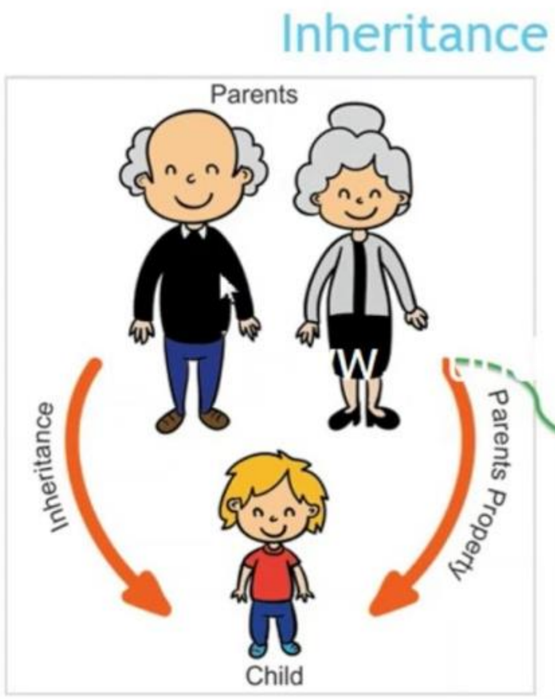
```
Syntax to declare a class:
access_modifier class<class_name>
{
    data member;
    method;
    constructor;
    nested class;
    interface;
}
```

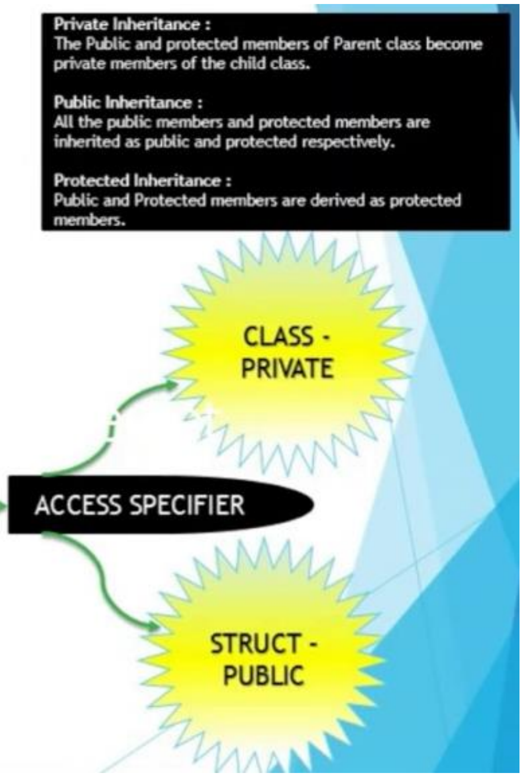
Objects, Structures, and Classes (Contd.)

- In general, class declarations can include these components, in order:
  - ***Modifiers***: A class can be public or has default access (Refer this for details).
  - ***Class keyword***: class keyword is used to create a class.
  - ***Class name***: The name should begin with an initial letter (capitalized by convention).
  - ***Superclass*** (if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
  - ***Interfaces*** (if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
  - ***Body***: The class body is surrounded by braces, **{ }**.
- Constructors are used for initializing new objects. Fields are variables that provide the state of the class and its objects, and methods are used to implement the behavior of the class and its objects.
- There are various types of classes that are used in real time applications such as nested classes, anonymous classes, lambda expressions.

Objects, Structures, and Classes (Contd.)

- **Structure and Class**: A structure works the same way as a class, except for certain differences.
  - The most important of them is hiding implementation details.
  - A structure will by default not hide its implementation details from whoever uses it in code, while a class by default hides all its implementation details and will therefore by default prevent the programmer from accessing them.



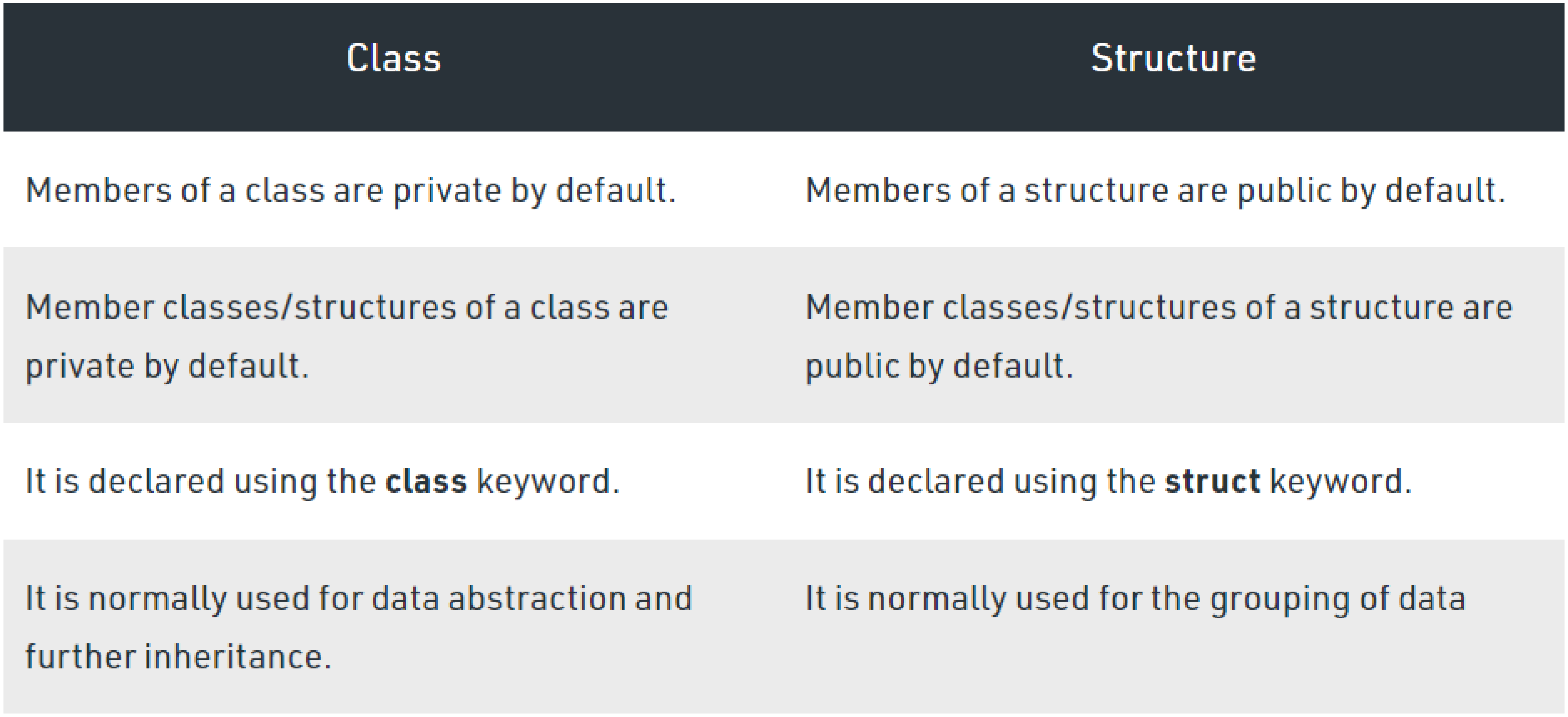

Private Inheritance :
The Public and protected members of Parent class become private members of the child class.

Public Inheritance :
All the public members and protected members are inherited as public and protected respectively.

Protected Inheritance :
Public and Protected members are derived as protected members.

Objects, Structures, and Classes (Contd.)

- **Structure and Class:**



Class	Structure
Members of a class are private by default.	Members of a structure are public by default.
Member classes/structures of a class are private by default.	Member classes/structures of a structure are public by default.
It is declared using the **class** keyword.	It is declared using the **struct** keyword.
It is normally used for data abstraction and further inheritance.	It is normally used for the grouping of data

Questions?