**C# / General OOPs / Programming**

1. What is OOPS?

* Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic
* An object can be defined as a data field that has unique attributes and behavior

1. What are the four pillars/principles of OOPS?

Abstraction

* + Creates the skeleton structure of what is required for application building
  + Achieved by creating a class and defining member variables, properties, and methods inside that is RELEVANT to the application being created
  + Solves: Design Level Problems

Encapsulation

* + Hides sensitive data: Language feature to restrict the access to members of an object.
  + Bundling of data and methods together: Data and methods that operate on the same data are bundled together.
  + Can be achieved by using access modifiers (private, public, etc.)
  + Solves: Implementation Level Problems

Inheritance

* + Methods and functions of the parent class is passed over to the derived class
  + Purpose of reusing code and saving time

Polymorphism

* + Main feature of allowing user to performing multiple tasks using the same method name, return type, parameters, sequence of parameters
  + Only possible if Inheritance is involved

1. Object called numbers. How to call the add method with parameters 4 and 6?

* Numbers.add(4,6)

1. How would you use encapsulation?

* Fields should be made private and manipulated through getter and setter methods.
* Everything else should be made as private as possible, restricted to the access needed.

1. What is a constructor?

* A constructor is a method in object-oriented programming that initializes a newly-created object.

1. Types of Constructors?
   * Default
   * Parameterized (at least one parameter)
   * Copy (initializes object to the values of an existing object)
   * Static - A static constructor is used to initialize any static data, or to perform a particular action that needs to be performed only once. It is called automatically before the first instance is created or any static members are referenced.
   * Private – Can only accessed within its class
2. How would you write a constructor?
   * A constructor’s name must be the same as its class name
   * Cannot have a return type
   * Parameters are optional
   * Overriding is allowed, but not multiple constructors with the same signature
3. Difference between Abstract class and Interface

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| **NOT** support **Multiple Inheritance** | supports **MULTIPLE Implementations** |
| contains **Data Member** | does **NOT** contain **Data Member** |
| contains **Constructors** | does **NOT** contain **Constructors** |
| contains **INCOMPLETE** (abstract) and  **COMPLETE MEMBER** | contains **ONLY incomplete Member**  (signature of member) |
| **contain Access Modifiers**  for the subs, functions, and properties | **CANNOT** have **Access Modifiers**  By default, everything is assumed as PUBLIC |
| **ONLY COMPLETE member** of the abstract class  can be **STATIC** | member of interface **CANNOT** be static |

1. What is the use of an interface and an abstract class?

* Multiple interfaces can be implemented, but only one abstract class can be inherited.
* Interface don’t contain fields; abstract classes can.
* Interfaces do not contain constructors; abstract classes do.
* Interfaces do not show completed properties (methods); abstract classes can contain either complete or incomplete properties.
* Neither interfaces nor abstract classes can be instantiated. write entries to this file.

1. What are access modifiers?

* Used to set the level of accessibility of classes, methods, and other members (e.g., private, protected, and public)

1. What is Static and its use?

Declares a member of a type so that it is specific to that type

* + Class-level, not object-level
  + CANNOT be instantiated

1. Can you use access modifiers in an interface?

* No. It’s all public and meant to be.

1. Array vs. ArrayList
   * An array is a fixed-length data structure
   * ArrayList is a variable-length data structure
     1. It can be resized itself when needed
     2. It is mandatory to provide the size of an array while initializing it directly or indirectly
     3. It is a class belonging to the Collection framework
2. What is Assembly in C#?

* An assembly is a basic building block of .Net Framework applications.
  + It is basically a compiled code that can be executed by the Common Language Runtime (CLR)
* An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality.

1. Difference between Boxing and Unboxing?

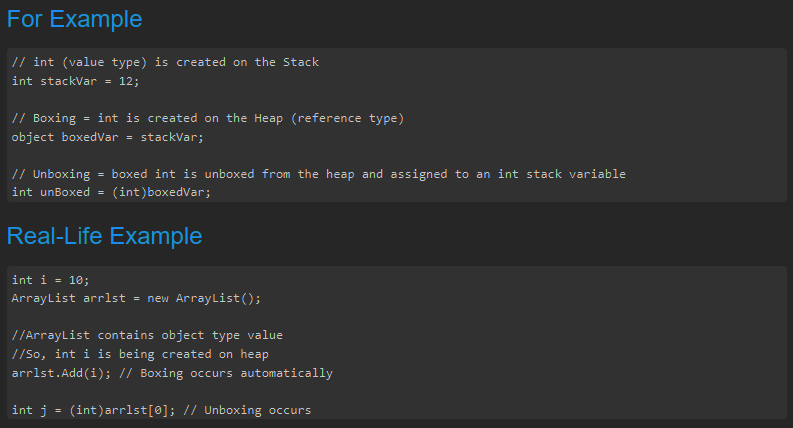
BOXING converts a value type to reference (object) type

* + A boxing conversion is intended for creating a copy of the value being boxed
    1. For example, integer-variable “*stackVar*” is assigned to object “*boxedVar*”
    2. In this case, the object data type is a reference type and base class of all the other classes in C# ultimately, an integer can be assigned to an object type
    3. This process of converting from the integer to the object data type is BOXING

UNBOXING converts the reference data type to the value type (such as a string or integer)

* In the unboxing process, the conversion is a completely reversed process of boxing

1. Unboxing conversion EXTRACTS the value FROM the reference type and assigns it to a value type



1. SOLID Principles

Single Responsibility Principle (SRP)

* Every software module should have only ONE reason to change (one responsibility)

Open/Closed Principle (OCP)

* A software module/class is OPEN for extension, but CLOSED for modification

Liskov Substitution Principle (LSP)

* You should be able to use any derived class instead of a parent class and have it behaved in the same manner that does not affect the base class

Interface Segregation Principle (ISP)

* Clients should NOT be forced to implement interfaces they don’t use.
* Instead, of one big interface, many small interfaces are preferred based on groups of methods, each one serving one submodule

Dependency Inversion Principle (DIP)

* High level modules/classes should NOT depend on low-level modules or classes.
* Both should depend upon abstractions.
* Secondly, abstractions should not depend upon details. Details should depend upon abstractions.

1. Method overLOADING

* Overloading is using the same method name with different parameters/different data types.
* Does NOT need inheritance.
* It is considered as static polymorphism

1. Method overRIDING

* Runtime polymorphism concept where a child class has a method with the same signature as its parent, with the purpose of having a unique implementation

1. Multiple Inheritance

* In C#, a class may implement multiple interfaces, but can only inherit from one class

1. What is CTS and CLS?

Common Type System (CTS)

* Describes the datatypes that can be used by managed code.
* CTS defines how these types are declared, used and managed in the runtime.
* It facilitates cross-language integration, type safety, and high-performance code execution.
* The rules defined in CTS can be used to define your own classes and values.

Common Language Specification (CLS)

* It is a subset of CTS.
* It defines a set of rules and restrictions that every language must follow which runs under the .NET framework.
* The languages which follow these set of rules are said to be CLS Compliant.
* In simple words, CLS enables cross-language integration or Interoperability.

1. What is CLR?

Common Language Runtime (CLR) in C#

* CLR is the basic and Virtual Machine component of the .NET Framework
* It is the run-time environment in the .NET Framework that runs the codes and helps in making the development process easier by providing the various services.
  + Basically, it is responsible for managing the execution of .NET programs regardless of any .NET programming language.
* Internally, CLR implements the Virtual Execution System (VES) which is defined in the Microsoft’s implementation of the Common Language Infrastructure (CLI).
  + The code that runs under the Common Language Runtime is termed as the Managed Code.
* In other words, you can say that CLR provides a managed execution environment for the .NET programs by improving the security, including the cross-language integration and a rich set of class libraries, etc.
  + CLR is present in every .NET framework version.

1. Managed vs. Unmanaged code

* Managed code is executed by the Common Language Runtime (CLR)
* Unmanaged code is executed by the operating system

1. What is garbage collection?

Garbage Collection (GC)

* A memory recovery feature built into programming languages such as C# and Java.
* A GC-enabled programming language includes one or more garbage collectors (GC engines) that automatically free up memory space that has been allocated to objects no longer needed by the program.

1. What is a thread in .NET?

* A thread is defined as the execution path of a program.
* Each thread defines a unique flow of control.

1. What is an ENUM?

* The “enum” keyword is used to declare an enumeration, a distinct type that consists of a set of named constants called the enumerator list.
* Usually, it is best to define an enum directly within a namespace so that all classes in the namespace can access it with equal convenience.

1. Value type vs. Reference Type

Value Type

* Holds the data within its own memory allocation

Reference Type

* Contains a pointer to another memory location that holds the real data.
* Reference Type variables are stored in the heap
* Value Type variables are stored in the stack

1. Class vs. Struct (structure)

<https://www.tutorialsteacher.com/csharp/csharp-struct>

1. What is Serialization?

* Serialization is the process of converting the state of an object into a form that can be persisted or transported.
* The complement of serialization is deserialization, which converts a stream into an object.
* Together, these processes allow data to be stored and transferred.

1. Types of Serialization?

Binary

* Binary serialization is the process where you convert your .NET objects into byte stream.
* In binary serialization all the public, private, even those which are read only, members are serialized and converted into bytes.

SOAP

* SOAP is a protocol based on XML, designed specifically to transport procedure calls using XML.
* Because a SOAP message is built using XML, the XmlSerializer can be used to serialize classes and generate encoded SOAP messages.

XML

* XML Serialization is the process of serializing a .Net Object to the form of XML or from an XML to .Net Object.

1. What is a Collection?

* Collection classes are specialized classes for data storage and retrieval.
* These classes provide support for stacks, queues, lists, and hash tables.
* Most collection classes implement the same interfaces.

1. What are the types of collections? Generic and NON-Generic

NON-GENERIC collections such as ArrayList, Queue, Stack, etc. can store elements of different data types.

* + When obtaining the items, the programmer should type cast them to the correct data type. Else, it can cause a runtime exception.

GENERIC collection classes can be used to overcome this issue. Generic collections store elements internally in arrays of their actual types.

* + Therefore, type casting is not required.
  + They can be used to store elements of the specified type or types.
  + Some Generic collection classes are:
    1. List<T>
    2. Dictionary <TKey, TValue>
    3. SortedList <TKey, TValue>
    4. HashSet<T>
    5. Queue<T>
    6. Stack<T>

1. Which collections are used as a key/value pair?
   * Generic – Dictionary
   * Non-generic – Hashtable
2. What are Exceptions?

* A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.
* Exceptions provide a way to transfer control from one part of a program to another.
* C# exception handling is built upon four keywords: *try*, *catch*, *finally*, and *throw*.

1. Catch block questions

* Catches a thrown exception such as ArithmeticException

1. How to assign a data type? Object?

Object myObject = new Object();

Car myCar = new Car();

1. What is AOT?

Ahead-of-time compilation (AOT compilation)

* The act of compiling an (often) higher-level programming language into an (often) lower-level language before execution of a program, usually at build-time, to reduce the amount of work needed to be performed at run time

1. Tell me about inheritance in C#

* Interfaces, abstract classes, and classes can be inherited.
* A class can only inherit one class, but can inherit any number of interfaces

1. Difference between abstract class and virtual?

* In abstract classes, methods must be overridden.
* In virtual classes, it’s optional

1. Best practices for exception handling?

* It is not good practice to catch a general Exception, but rather each specific exception that can happen

1. Can you have multiple catch blocks?

* Yes. It’s not usually good practice to catch a general Exception, but rather each type of exception that can happen (e.g., DivideByZeroException, FormatException, and OverflowException when attempting to parse a user input into a number and then using it)

1. Difference between constant and read-only?
   1. In C#, a const keyword is used to declare constant fields and constant local.
      1. The value of the constant field is the same throughout the program or in other words, once the constant field is assigned the value of this field is not be changed.
   2. In C#, constant fields and locals are not variables, a constant is a number, string, null reference, Boolean values.
   3. In C#, you can use a read-only keyword to declare a read-only variable.
      1. This read-only keyword shows that you can assign the variable only when you declare a variable or in a constructor of the same class in which it is declared.
2. Can you have multiple constructors in C#?

* Yes, but they must be overloaded (no two can have the same signature).

1. What is .csproj file for?
   * Contains the list of the files in a project, plus references to system assemblies
2. What happens when you compile code in C#?

<https://www.geeksforgeeks.org/how-c-sharp-code-gets-compiled-and-executed/>

1. How would you update a database through C#?
   * Create an SqlConnection initialized with a connection string.
   * Create an SqlCommand, passing in an SQL statement in a string, followed by the SqlConnection as the second parameter.
   * Substitute the appropriate parameters within the string with the variables
   * Execute the scalar from the SqlCommand using ExecuteNonQuery()

Text

Description automatically generated