Brandon Girard

**1. reference type vs. value type**

Value types are located on the stack, directly contains their data, has its own copy of data, and any operation on it will not affect another. Reference types are located on the heap, store the memory location of the data, can have more than one variable reference the same object, and operations on it can affect another.

**2. boxing vs. unboxing**

Boxing is when a value type is converted to a reference and unboxing is vice versa.

**3. abstract class vs. interface**

Interface is a contract that only contains method signatures. A class can implement multiple interfaces, but inherit from only one abstract class. An abstract class can have both method signatures and finished methods, while interfaces can only have method signatures.

**4. Overriding vs. Overloading**

Overloading means to have multiple methods that have the same signature, but with different parameters and/or return types. Overriding means that a virtual method in the parent class can be changed in the child class.

**5. What does constructor do in a class? Can it be overridden? Can it be overloaded?**

Special method that is called upon instantiation of a class, it’s used to help initialize the class. It can be overloaded, but not overridden.

**6. What does static keyword do in C#?**

Static members belong to the class and are accessible from every derived object.

**7. Difference between Virtual method and Abstract method?**

A virtual method is a method that has logic and can be optionally overridden in a child class, while an abstract method has no logic and must be overridden in the child class.

**8. what are delegates in C#, what are different types of built-in delegates**

Delegates are pointers to methods.

Action: will take functions that have generics input, but return nothing(return void) as parameter

Predicate: will take functions that have generics input, and will return a boolean value as parameter

Func: will take functions that have generics input, and will return a generics type as output to be a parameter

**9. Explain different access modifiers in C#**

Public: The type or member can be accessed by any other code in the same assembly or another assembly that references it.

Private: The type or member can be accessed only by code in the same class or struct.

Protected: The type or member can be accessed only by code in the same class, or in a class that is derived from that class.

Internal: The type or member can be accessed by any code in the same assembly, but not from another assembly.

Protected internal: The type or member can be accessed by any code in the assembly in which it's declared, or from within a derived class in another assembly.

Private internal: The type or member can be accessed by types derived from the class that are declared within its containing assembly.

**10. What is the extension method in C#? examples of built-in extension methods? How to create custom extension methods?**

Extension method is a way to add new functionality into an existing type. Linq and Dapper are examples of extension methods. To create extension methods the class containing the extension method must be static, the extension method must be static, the first parameter to the extension method must be the type to be extended, and the first parameter must be written after this keyword.

**11. Ref vs. Out vs. Params**

Ref allows a method parameter to use an external reference instead of a copy. Out allows a method to assign/return multiple values using predefined variables. Params allows for a method to take in an unknown quantity of same typed inputs.

**12. Pass by reference vs. Pass by Value**

When you pass by reference, you’re passing the memory location of a value, while passing by value is passing a value defined in one location and copying it to another.

**13. array vs. arrayList**

* Array stores data of the same type whereas ArrayList stores data in the form of the object which may be of different types.
* Size of an ArrayList grows dynamically while Array size remains static throughout the program.
* Insertion and deletion operation in ArrayList is slower than an Array.
* Arrays are strongly typed whereas ArrayLists are not strongly typed.

**14. example of encapsulation, where to implement**

Private string password;

Public string name;

Protected decimal salary;

Where to implement encapsulation is up to the developer and the project design.

**15. how do you handle exceptions? Syntax.**

Any code that might throw an error can be put in a try/catch block which will throw an exception when an error occurs.

Try{

//code here

}

catch(Exception e){

//code to be executed when exception occurs

}

**16. what is generic, syntax to define**

Putting <T> next to a class or method then using the T as the assumed generic type in the class/method logic.

**17. what is LINQ**

Extension methods with sql-like syntax that we can use in C# to get data from different data sources.

**18. IEnumerable vs. IQuerable**

IEnumerable: when linq is working with in-memeory data source(list, array..)

IQuerable: when linq is working with out-of-memory data source

**19. First vs. FirstOrDefault vs. Single vs. SingleOrDefault**

First: returns the first record when there are one or more records, if there are no matches an exception will be thrown.

FirstOrDefault: return the first record where there are one or more records, if there are no matches a default value will be assigned.

Single: return the matched single record, if there are no matches or multiple matches then an exception will be thrown.

SingleOrDefault: return the matched single record, if there are no matches assign the default value, if there is more than one match throw an exception.

**20. Any vs. All**

any():check if any of the element satisfy the specific condition, if yes, return true, if not return false

all(): check if all the elements satisfy the specific condition, if yes, return true, if no, return false

**21. Skip vs. Take**

Take: extracts the first n elements

Skip: extracts elements excluding the first n elements.

**22. Deferred execution and Immediate execution in LINQ**

For deferred execution the actual execution of the query is deferred until you iterate over the query variable in a for each statement, whereas for immediate execution the range of source elements are immediately iterated over.