**Q1. 1 Why would you create an abstract class, if it can have no real instances?**

* to avoid redundant coding in children
* to explore a hypothetical class
* to prevent unwanted method implementation
* to reserve memory for an unspecified class type

#### Q2. 2 Why would you create an abstract class, if it can have no real instances?

* to have common behavior in derived classes
* to explore a hypothetical class
* to prevent unwanted method implementation
* to reserve memory for an unspecified class type

#### Q3. What is the best reason to use a design pattern?

* It will result in code that is more extensible and maintainable
* It will result in a more compact product.
* It will speed initial development.
* It will allow you to add that design pattern to your resume.

#### Q4. What is encapsulation?

* defining classes by focusing on what is important for a purpose
* hiding the data and implementation details within a class
* making all methods private
* using words to define classes

#### Q5. What is an IS-A relationship?

* It implies encapsulation.
* A superclass object has an IS-A relationship with its subclass.
* It implies a virtual method.
* A subclass object has an IS-A relationship with its superclass or interface

#### Q6. You want a method with behavior similar to a virtual method--it is meant to be overridden --expect that it does not have a method body. It just has a method signature. What kind of method should you use?

* an abstract method
* a public internal method
* an internal method
* a protected internal method

#### Q7. Which code creates a new object from the Employee class?

* Employee currentEmployee = Employee.Create();
* Employee currentEmployee = new Employee();
* Employee currentEmployee;
* Employee currentEmployee = Employee.New();

#### Q8. Which type of constructor cannot have a return type?

* default
* copy
* parameterized
* Constructors do not have a return type

#### Q9. 1 When is a constructor executed?

* when an object is created from a class using the new keyword
* when an class is defined using the class keyword
* every time an object is referenced
* when an object is created from a class using the create keyword

#### Q10. If a local class is defined in a function, what is true for an object of that class?

* The object can be accessed, declared, and used locally in that function.
* The object must be declared inside any other function.
* The object is temporarily accessible outside the function.
* The object can call all the other class members anywhere in the program.

#### Q11. Which two blocks are used to handle and check errors?

* do and check
* catching and trying
* try and catch
* do and while

#### Q12. Why would you implement composition using an id instead of a reference?

* It makes it easier to save the entity.
* It can make the entity retrieval more efficient
* It minimizes coupling.
* all of these answers

#### Q13. Which statement best describes the method of inheritance in OOP?

* Inheritance describes the ability to create new classes based on an existing class.
* Inheritance means that a group of related properties, methods, and other members are treated as a single unit or object.
* Inheritance forces a class to have a single responsibility from only one parent.
* Inheritance means that you will never have multiple classes that can be used interchangeably, even though each class implements the same properties or methods in different ways.

#### Q14. Which of the following is NOT an advantage of using getters and setters?

* Getters and setters can speed up compilation.
* Getters and setters provide encapsulation of behavior.
* Getters and setters provide a debugging point for when a property changes at runtime.
* Getters and setters permit different access levels.

#### Q15. if an object is passed by reference, the changes made in the function are reflected \_.

* to the main object of the caller function, too
* on the caller function object and also the called function object
* on the copy of the object that is made during the pass
* only in the local scope of the called function

#### Q16. What is a method?

* a set of instructions designed to perform a frequently used operation within a program and return no values
* the exact same thing as a function and subroutine
* a set of variables that can change over time
* They are functions attached to specific classes (or instances) in object-oriented programming.

#### Q17. Why is code duplication so insidious?

* The duplication uses unnecessary space.
* One has to maintain all the duplicates.
* Duplication can cause intellectual property concerns.
* Duplication is easy to hide.

#### Q18. Can abstract classes be used in multilevel inheritance?

* No, abstract classes can be used only in single-level inheritance since they must be immediately implemented.
* yes, always
* yes, but with only one abstract class
* No, abstract classes do not have constructors.

#### Q19. What is the relationship between abstraction and encapsulation?

* Abstraction is about making relevant information visible, while encapsulation enables a programmer to implement the desired level of abstraction.
* Abstraction and encapsulation are essentially the same.
* Abstraction and encapsulation are unrelated.
* Encapsulation is about making relevant information visible, while abstraction enables a programmer to implement the desired level of encapsulation.

#### Q20. Which of these keywords are access specifiers?

* abstract and public
* public and private
* this and final
* final and abstract

#### Q21. What is a reference to an object?

* It is the address of variable only -- not the method of an object.
* It is a shallow pointer that contains address of an object.
* It is the physical address of an object.
* It is the address where the variables and methods of an object are stored.

#### Q22. The open/closed principle states that classes should be open for \_ but closed for \_.

* refactoring; duplication
* modification; duplication
* extension; modification
* reuse; encapsulation

#### Q23. Why would you override a method of a base class?

* to define a method that must be implemented in a derived class
* to define a custom implementation of an inherited member
* to define a method that must be implemented in a superclass only
* to define a class that can be inherited from

#### Q24. There are five classes. Class E is derived from class D, D from C, C from B, and B from A. Which class constructor(s) will be called first if the object of E or D is created?

* A
* B
* C
* C and B

#### Q25. You have modules that are dependent on each other. If you change one module, you have to make changes in the dependent modules. What term is used to describe this problem, and what is a potential solution?

* Cohesion. A solution is to show that each module has certain responsibilities and to use an anticohesive design pattern.
* Encapsulation. A solution is to implement one of the SOLID principles to ensure the modules do not encapsulate with each other.
* Coupling. A solution is to refactor the code to be loosely coupled by using inversion of control and dependency injection.
* Dependency. A solution is to implement polymorphism and abstraction to change and extract dependent elements of a module so that it functions on its own.

#### Q26. Which type of function can be used for polymorphism?

* virtual function
* inline function
* undefined function
* private function

#### Q27. Which choice is a benefit of using dependency injection?

* loose coupling
* code reusability
* lazy initialization
* data abstraction

#### Q28. Are you required to return an object if it was passed by reference to a function, and why or why not?

* Yes, the caller function needs to reflect the changes.
* No, you should use a global variable instead.
* No, changes will be automatically reflected in the calling function.
* Yes, the object must be the same in the caller function.

#### Q29. What is the main idea behind separation of concerns?

* All of these answers
* Applications are decomposed into parts
* Parts are defined with minimal overlap
* Each part is responsible for a separate concern

#### Q30. What is the purpose of the finally block?

* To always run the finally block of code when the try block exits
* To run code when an exception has not occurred
* To run the block if an exception occurred
* To run code whenever garbage collection requires it

#### Q31. An instance of which type of class cannot be created?

* Protected class
* Base class
* Anonymous class
* Abstract class

#### Q32. What is the difference between an interface and an abstract class?

* Interfaces can contain code or data. Abstract classes do not contain code or data. A class can inherit from more than one abstract class but can implement only one interface.
* Interfaces can contain code or data. Abstract classes do not contain code or data. A class can inherit from only one abstract class but can implement an unlimited number of interfaces.
* Abstract classes can contain code or data. Interfaces do not contain code or data. A class can inherit from only one abstract class but can implement an unlimited number of interfaces.
* Abstract classes can contain code or data. Interfaces do not contain code or data. A class can inherit from more than one abstract class but can implement only one interface.

#### Q33. What are the four principles of object-oriented programming?

* manipulation, encapsulation, inheritance, and dependency inversion
* dependency inversion, open/closed principle, encapsulation, and inheritance
* interface segregation, abstraction, dependency inversion, and inheritance
* abstraction, encapsulation, inheritance, and polymorphism

#### Q34. From the SOLID principles of object-oriented programming, which statement best describes the Liskov substitution principle?

* A class should have only a single responsibility—that is, only changes to one part of the software's specification should be able to affect the specification of the class.
* Software entities should be open for extension, but closed for modification.
* Many client-specific interfaces are better than one general-purpose interface.
* objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

#### Q35. What is a virtual Method?

* a method that you expect may be redefined in derived classes
* a method that you do not expect to be redefined in derived classes
* a private method that you do not expect to be redefined in derived public classes
* a method that exists temporarily - once used, it ceases to be used by any caller

#### Q36. How coupled should your classes be and why?

* You should increase coupling to improve dependencies between classes.
* You should limit coupling to reduce dependencies between classes.
* You should increase coupling so that class members relate to the class purpose.
* You should limit coupling so that class members relate to the class objective.

#### Q37. Can you have two classes with the same name in the same project?

* No, you cannot.
* Yes, as long as their constructors are different.
* Yes, as long as their methods are different.
* Yes, as long as they are in different namespaces.

#### Q38. Objects are passed by ****\_****

* value or reference, depending on the programming language used
* value
* value or reference, depending on program
* reference

#### Q39. Which choice is a benefit of using dependency injection?

* loose coupling
* code reusability
* lazy initialization
* data abstraction

What is Clean Architecture with brief explanation?

What is the SOLID principles with brief explanation?

What is API?

What are the types of API with brief explanation for each type?