Q1. What is an example of dynamic binding?

- [ ] any method
- [ ] method overloading
- [x] method overriding
- [ ] compiling

Q2. For which case would the use of a static attribute be appropriate?

- [ ] the number of people in each house in a small neighborhood
- [ ] the lot size for each house in a small neighborhood
- [ ] the color of each house in a small neighborhood
- [x] the weather conditions for each house in a small neighborhood

Q3. 1 Why would you create an abstract class, if it can have no real instances?

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

Q4. 2 Why would you create an abstract class, if it can have no real instances?

- [x] 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

Q5. When does static binding happen?

- [ ] only when you export
- [ ] both at compile time and runtime
- [x] at compile time
- [ ] at runtime

Q6. What is the best reason to use a design pattern?

- [x] 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.

Q7. What is encapsulation?

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

Q8. 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.
- [x] A subclass object has an IS-A relationship with its superclass or interface

Q9. 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?

- [x] an abstract method
- [ ] a public internal method
- [ ] an internal method
- [ ] a protected internal method

Q10. Which code creates a new object from the Employee class?

- [ ] Employee currentEmployee = Employee.Create();
- [x] Employee currentEmployee = new Employee();
- [ ] Employee currentEmployee;
- [ ] Employee currentEmployee = Employee.New();

Q11. Which type of constructor cannot have a return type?

- [ ] default
- [ ] copy
- [ ] parameterized
- [x] Constructors do not have a return type

Q12. 1 When is a constructor executed?

- [x] 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

Q13. 2 When is a constructor executed?

- [x] when an object is created from a class
- [ ] 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

Q14. If a local class is defined in a function, what is true for an object of that class?

- [x] 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.

Q15. Which two blocks are used to handle and check errors?

- [ ] do and check
- [ ] catching and trying
- [x] try and catch
- [ ] do and while

Q16. Why would you implement composition using an id instead of a reference?

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

Q17. Which statement best describes the method of inheritance in OOP?

- [x] 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.

Q18. Which type of inheritance ,when done continuously, is similar to a tree structure?

- [ ] multilevel
- [ ] hierarchical and multiple
- [x] hierarchical
- [ ] multiple

Q19. Which statement is true?

- [x] A default parameter's constructor is not equivalent to the default constructor
- [ ] A default constructor is inherited from a parent class
- [ ] A default constructor can be called explicitly
- [ ] A default constructor cannot be defined by the coder

Q20. Which of the following is NOT an advantage of using getters and setters?

- [x] 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.

Q21. In context of OOP, what is association?

- [x] Association is a relationship where all objects have their own life cycle and there is no owner.
- [ ] Association is the process where model elements cooperate to provide higher-level behavior.
- [ ] Association is whole/part relationship where one object is composed of one or more other objects, each of which is considered a part of the whole.
- [ ] Association is where all objects have their own life cycle, but there is ownerhip, and child objects can not belong to another parent object.

Q22. How are user stories different from use cases?

- [x] User Stories are shorter and less detailed.
- [ ] User stories are more accurate.
- [ ] User stories are more detailed and structured.
- [ ] User storised are more anecdotal and personal.

Q23. Which type of inheritance must be used so that the resultant is hybrid?

- [x] multiple
- [ ] any type of inheritance
- [ ] multilevel
- [ ] hierarchical

Single Inheritance is where a derived class inherits properties and behaviour from a single base class. Example: Class A → Class B. Hierarchical Inheritance is where more than one derived class is created from a single base class. Example: Class A → Class B → Class C. Multiple Inheritance is for deriving a class from multiple base classes. Here, the child objects programmers create will have combined aspects of characteristics and features from multiple parent classes. These objects do follow their hierarchies of base classes. Multilevel Inheritance is where a child class is derived from another derived class. This feature carries combined aspects of multiple classes and follows their hierarchies. Hybrid Inheritance is a heterogeneous feature of using multiple inheritances. Here a child class is derived from one or more

combinations of single, hierarchical, and multilevel inheritances. This inheritance is adopted for programs to mix different types of inheritance; for example, when mixing a single inheritance with multiple inheritances or maybe a situation when multiple inheritances are mixed within a single program.

reference

Q24. A language that does not support polymorphism but supports classes is considered what?

- [x] an object-based language
- [ ] a class-based language
- [ ] a procedure-oriented language
- [ ] if classes are supported, polymorphism will be supported

Q25. Two classes combine private data members and provide public member functions to access and manipulate those data members. Where is abstraction used?

- [ ] Abstraction is using a private access specifier for the data members.
- [x] Abstraction is using public member functions to access and manipulate the data members.
- [ ] Abstraction is using the class concept with both data members and member functions.
- [ ] There is insufficient information to decide where abstraction is being used.

Q26. What are the five Creational Design patterns by the Gang of Four ?

- [ ] Observer, State, Strategy, Template Method, and Visitor.
- [ ] Composite, Visitor, State, Prototype, and Singleton.
- [ ] Composite, Builder, Factory Method, Prototype, and Singleton.
- [x] Abstract Factory, Builder, Factory Method, Prototype, and Singleton.

Q27. In multilevel inheritance, one class inherits how many classes?

- [x] one class only
- [ ] two classes
- [ ] as many classes as required
- [ ] at least two classes

Q28. if an object is passed by reference, the changes made in the function are reflected _.

- [x] 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

Q29. What is a method?

- [ ] a set of instructions designed to perform a frequently used operation within a program and return no values
- [x] the exact same thing as a function and subroutine
- [ ] a set of variables that can change over time
- [ ] a procedure associated with data and behaviour

Q30. A mobile phone is made up of components such as a motherboard, camera, and sensors. The motherboard represents all the functions of a phone, the display shows the display only, and the phone is represented as a whole. Which of the following has the highest level of abstraction?

- [ ] camera
- [ ] display
- [ ] motherboard
- [x] mobile phone

Q31. Which class has the highest degree of abstraction in a multilevel inheritance relationship of five levels?

- [ ] the class at the third level
- [x] the class at the first level
- [ ] All have the same degree of abstraction.
- [ ] the class at the second level

Q32. Which is NOT one of the basic types of inheritance?

- [ ] multilevel inheritance
- [x] double inheritance

- [ ] single inheritance
- [ ] hierarchical inheritance

Q33. Why is code duplication so insidious?

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

Q34. When and how often is a static constructor called?

- [ ] It is called initially when an object is created and called with every new object instance.
- [ ] It is called when an object is destroyed and only one time.
- [x] It is called initially when an object is created and only one time.
- [ ] It is created at time when the object is discarded.

Q35. What does the code shown below demonstrate, and why?

```
static void Multiply(int num1, int num2) {};
static void Multiply(double num1, double num2, double num3) {};
static void Multiply(float num1, float num2) {};
```

- [ ] polymorphism, because each method can perform different task
- [ ] method overriding, because it display the same method name, different or same parameters, and same return type
- [x] method overloading, because it allows the creation of several methods with the same name, wich differ by the type of input via parameter
- [ ] method overriding, because it display the same method name, different parameters, and same return type

Q36. What is the purpose of static constructor?

- [x] to initialize all the members with static value
- [ ] to delete the static members when not required
- [ ] to initialize the static members of class
- [ ] to clear all the static members' initialized values

Q37. What are CRC Cards?

- [ ] Code Responsibility Collection cards are a brainstorming tool used in the design of procedural software
- [x] Class Responsibility collaboration cards are a brainstorming tool used in the design of oop software
- [ ] Code Responsibility Correction cards are tools used for debugging
- [ ] Code Responsibility Correction cards are tools for modeling

reference link

Q38. 1 How are contents of a composition different from those of aggregation?

- [ ] if one element of an aggregation is dereferenced, all its elements are eligible for garbage collection
- [x] if a composition dies, the contents die
- [ ] the contents of a composition are all siblings
- [ ] an aggregation contains only abstract classes

Q39. 2 Which statement about compositions and aggregations is true?

- [ ] if one element of an aggregation is dereferenced, all its elements are eligible for garbage collection
- [x] if a composition dies, the contents die
- [ ] the contents of a composition are all siblings
- [ ] an aggregation contains only abstract classes

Q40. What is the result of using more abstraction?

- [ ] it can increase code vulnerability
- [ ] it can make code unsafe
- [x] it can limit code readability
- [ ] it can be safer for coding

Q41. Which is false for a member function of a class?

- [ ] Member functions can be defined only inside or outside the class body.
- [ ] Member functions can be made to be friends of another class.

- [x] Member functions do not need to be declared inside the class definition.
- [ ] All member functions need to be defined.

Q42. Why is inheritance used when creating a new class?

- [ ] to protect attributes from unwanted changes
- [ ] to delegate coding responsibility more efficiently
- [ ] to conserve memory
- [x] to avoid writing duplicate code
- [x] to separate class behavior from the more general

NOTE: I don't have 4th variant in my test, it changed to new 5th variant. Is it also true?

Q43. In addition to attributes and behaviours, what quality must a class possess?

- [x] a name
- [ ] a state
- [ ] a color
- [ ] an object

Q44. Which type of function among the following shows polymorphism?

- [ ] inline function
- [ ] undefined function
- [x] virtual function
- [ ] class member function

Q45. Which words in the following list are candidates for objects: trumpet, clean, enrage, leaf, tree, collapse, active, and lively?

- [ ] leaf and tree
- [ ] clean, enrage, and collapse
- [ ] clean, active, and lively
- [x] leaf, tree, and trumpet

Q46. What best describes what object-oriented programming does?

- [x] It focuses on objects that interact cleanly with one another.
- [ ] It programs exclusively to interfaces.
- [ ] It programs exclusively to classes.
- [ ] It creates one class for all business logic.

Q47. 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.
- [x] yes, always
- [ ] yes, but with only one abstract class
- [ ] No, abstract classes do not have constructors.

Q48. What type of inheritance may lead to the diamond problem?

- [ ] single level
- [ ] multilevel
- [ ] hierarchical
- [x] multiple

Q49. What is the relationship between abstraction and encapsulation?

- [x] 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.

Q50. Which of these keywords are access specifiers?

- [ ] abstract and public
- [x] public and private
- [ ] this and final

- [ ] final and abstract

Q51. 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.
- [x] It is the address where the variables and methods of an object are stored.

Q52. Why is unit testing harder in OOP than functional programming?

- [x] Objects may maintain internal state, which is not easily accessible by the tests.
- [ ] The quality of unit testing frameworks for functional languages is better.
- [ ] OOP promotes code reuse, which means that your tests have to consider more use cases.
- [ ] Object-oriented languages tend to rely on frameworks such as Spring or Hibernate, which make them difficult to test.

Q53. What is the function of a user diagram?

- [ ] It connects actors to use cases.
- [x] It links actors to roles played in all use cases.
- [ ] It lists all actors for each use case.
- [ ] It minimizes the number of actors required.

Q54. How do object behaviour and attributes differ?

- [ ] Behaviour describe dynamic properties; attributes are static.
- [x] Attributes describe a state; behaviours describe a change.
- [ ] Attributes apply only to a specified object; behaviour apply to other linked objects.
- [ ] Behaviours are vector quantities; attributes are scalars.

Q55. The open/closed principle states that classes should be open for _ but closed for _.

- [ ] refactoring; duplication

- [ ] modification; duplication
- [x] extension; modification
- [ ] reuse; encapsulation

Q56. Why would you override a method of a base class?

- [ ] to define a method that must be implemented in a derived class
- [x] 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

Q57. What is a copy constructor?

- [x] It is a unique constructor for creating a new object as a copy of an object that already exists. There will always be only one copy constructor that can be either defined by the user or the system.
- [ ] It is a constructor that duplicates itself when requested on demand.
- [ ] It is a common constructor for preventing the creation of a new object as a copy of an object that already exists. There will always be multiple standard constructors that can be either defined by the user or the system.
- [ ] It is a constructor that duplicates itself on its own, based on memory available.

Q58. What defines the catch block most accurately?

- [x] The catch block that will be executed is the one that best matches the type of exception thrown.
- [ ] Multiple catch blocks can never be associated with a single try block.
- [ ] Multiple catch blocks are mandatory for each try block.
- [ ] Multiple catch blocks will all be executed in the case of an exception.

Q59. 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?

- [x] A
- [ ] B
- [ ] C

- [ ] C and B

Q60. 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.
- [x] 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.

Q61. _ describes an aggregation

- [ ] A class of resources
- [ ] A group of methods
- [x] A collection of objects
- [ ] A list of children

Q62. Which type of function can be used for polymorphism?

- [x] virtual function
- [ ] inline function
- [ ] undefined function
- [ ] private function

Q63. Which choice is a benefit of using dependency injection?

- [x] loose coupling
- [ ] code reusability
- [ ] lazy initialization
- [ ] data abstraction

Q64. 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.
- [x] No, changes will be automatically reflected in the calling function.
- [ ] Yes, the object must be the same in the caller function.

Q65. Why is inheritance?

- [ ] ...

Q66. What is the best example of a superclass and subclass relationship?

- [x] car:toyota
- [ ] ducks:pond
- [ ] toes:feet
- [ ] rock:stone

Q67. Which statements best describe the Gang of Four design patterns called Memento and Observer?

- [ ] Memento notifies multiple classes of changes. Observer captures and restores an object's internal state.
- [ ] Memento defers the exact steps of an algorithm to a subclass. Observer defines a new operation to a class without change.
- [ ] Memento alters an object's behavior when its state changes. Observer encapsulates an algorithm inside a class.
- [x] Memento captures and restores an object's internal state. Observer notifies multiple classes of changes.

Q68. What does the value (0.5,0.5,0.5) indicate in the class diagram specification position: Coordinate = (0.5,0.5,0.5)?

- [ ] a default value of the Coordinate attribute
- [ ] the size of the position array
- [ ] an increment of the position attribute value
- [x] a default value of the position attribute

Q69. What is the most accurate example of the Liskov substitution principle?

- [ ] A

```
public class Car{
}
public class FlyingCars extends Car{
   public void fly(){}
}
public class Tesla FlyingCar{}
public class Honda Car{}
```

- [ ] B

```
public class Car{
   public void fly(){}
}
public class Tesla extends Car{}
public class Honda extends Car{}
```

- [ ] C

```
public class Car{
public void fly(){}
}
public class Tesla Car{}
public class Honda Car{}
```

- [x] D

```
public class Car{
}
public class FlyingCars extends Car{
   public void fly(){}
}
public class Tesla extends FlyingCar{}
public class Honda extends Car{}
```

reference link

Q70. What is the difference between a parameter and an argument?

- [ ] An argument can have many values while a parameter can have only one value.
- [ ] An argument is the variable used for input values in a method. A parameter is the specific input value passed to the method.
- [x] A parameter is a variable in the declaration of a function. An argument is the value of this variable that gets passed to the function.
- [ ] Parameters and arguments are the same

Q71. What is the scope of a class nested inside another class?

- [ ] Protected scope
- [ ] Private scope
- [ ] Global scope
- [x] Depends on access specifier and inheritance used

Explanation: It depends on the access specifier and the type of inheritance used with the class, because if the class is inherited then the nested class can be used by subclass too, provided it's not of private type.

Q72. Methods and attributes that define an object are a kind of blueprint called what?

- [ ] a collection
- [ ] a variable
- [x] a class
- [ ] a procedure

Q73. Assume single inheritance is used with classes A and B while A is the base class. Then assume classes C, D, and E, where C is a base class and D is derived from C, then E is derived from D. Class C is made to inherit from class B. Which type of inheritance is reflected?

- [x] Multilevel
- [ ] Hybrid
- [ ] Single level
- [ ] Multiple

Q74. What is the main idea behind separation of concerns?

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

Q75. What is the purpose of the finally block?

- [x] 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

Q76. Which choice is not an OOP language?

- [ ] C#
- [ ] Java
- [x] C
- [ ] Python

Q77. What is the function of a finalizer or destructor?

- [x] To relinquish resources that are no longer needed
- [ ] To delete a variable name
- [ ] To reset an attribute value
- [ ] To hold space, even after an object is no longer being used

Q78. An instance of which type of class cannot be created?

- [ ] Protected class
- [ ] Base class
- [ ] Anonymous class
- [x] Abstract class

Q79. In the context of OOP, what is composition?

- [ ] Composition is the act of one object passing to another object an operation to be performed on behalf of the initial object.
- [x] Composition is a part/hole relationship where an object is composed of one or more other objects, each of which is considered a part of the whole.
- [ ] Composition is a binding where the class/name association is not made until the object designated by the name is created at execution time
- [ ] Composition is a process of collecting classes that provide a set of services for a particular domain

Q80. Static polymorphism uses method _ ?

- [x] overloading
- [ ] inheritance
- [ ] abstraction
- [ ] overriding

Q81. What does a concrete class not have?

- [ ] parents
- [x] pure virtual functions
- [ ] attributes
- [ ] purposes

Q82. How does dynamic typing complicate troubleshooting?

- [x] It can be difficult to identify variables that are incorrectly typed
- [ ] The dynamic variables can assume only limited values
- [ ] Storage is fixed for dynamic variables
- [ ] Static variables are more flexible than dynamic variables

Q83. What is the difference between early binding and late binding?

- [ ] Early binding is when a variable is assigned a value when a scope is created. Late binding is when a variable is assigned a value after a scope is exited
- [ ] Early binding is when a variable is assigned a value when the program starts. Late binding is when a variable is assigned after the program is running
- [ ] There is no difference. In both cases, variables are assigned values when a program has completed startup and is running
- [x] Early binding is when a variable is assigned its value at compile time. Late binding is when a variable is assigned a value at run time

Q84. 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 only implement 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 interface
- [x] Abstract classes can contain code or data. Interface 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. Interface do not contain code or data. A class can inherit from more than one abstract class but can only implement one interface

Q85. What parameters are required to be passed to a class constructor?

`Here they haven't mentioned any specific language so let's consider all languages.`

- [ ] reference to subclass // References to subclass are never required as you can simply Initialize subclass & use their object.
- [ ] reference to base class // References to the base class are not required in Java, Javascript & Python
- [ ] reference to this pointer // While Python & Javascript may require passing this or self in the constructor, It is not passed in Java constructor.
- [x] none // Above 3 are incorrect so "none" is the answer

Q86. 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
- [x] abstraction, encapsulation, inheritance, and polymorphism

Q87. 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.

- [x] objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

Q88. In addition to responsabilities, what should be listed on Class-responsability-collaboration (CRC) cards?

- [ ] which programming language will be used.
- [ ] the programmer responsible for implementation.
- [ ] interacting classes.
- [x] attributes.