1.What is the purpose of inheritance in object-oriented programming?
a) To allow a class to inherit properties and behaviors from another class.
b) To create new classes from existing ones.
c) To promote code reusability and maintainability.
d) All of the above.

Answer: d) All of the above.
Explanation: Inheritance enables the creation of new classes (derived or child classes) that inherit properties and behaviors from existing classes (base or parent classes), promoting code reusability and maintainability.

2.Which of the following is not a type of inheritance?
a) Single inheritance
b) Multiple inheritance
c) Hybrid inheritance
d) Circular inheritance

Answer: d) Circular inheritance
Explanation: Circular inheritance is not a valid type of inheritance. It refers to a situation where a class is derived from itself or from a class hierarchy where one class is derived from another in a circular manner, which is not allowed in most programming languages.

3.In the context of inheritance, what does the ‘is a’ relationship signify?
a) It signifies a relationship between classes where one class is a specialized version of another.
b) It signifies a relationship between classes where one class contains an instance of another.
c) It signifies a relationship between classes where both classes are equivalent.

d) It signifies a relationship between classes where one class is composed of another.

Answer: a) It signifies a relationship between classes where one class is a specialized version of another.
Explanation: The ‘is a’ relationship, also known as an inheritance relationship, indicates that one class (child class) is a specialized version of another class (parent class), inheriting its properties and behaviors.

4.What is association in object-oriented programming?
a) It represents a strong relationship between classes where one class contains an instance of another.
b) It represents a weak relationship between classes where one class is a specialized version of another.
c) It represents a relationship between classes where both classes are equivalent.
d) It represents a relationship between classes where one class is composed of another.

Answer: a) It represents a strong relationship between classes where one class contains an instance of another.
Explanation: Association signifies a relationship between classes where one class contains an instance of another class. It can be one-to-one, one-to-many, or many-to-many.

5.What is aggregation in object-oriented programming?
a) It represents a strong relationship between classes where one class contains an instance of another.
b) It represents a weak relationship between classes where one class is a specialized version of another.
c) It represents a relationship between classes where both classes are equivalent.

d) It represents a relationship between classes where one class is composed of another.

Answer: d) It represents a relationship between classes where one class is composed of another.
Explanation: Aggregation signifies a relationship between classes where one class is composed of another class. It represents a whole-part relationship, where the part can exist independently of the whole.

6.Which of the following best describes the concept of interfaces in object-oriented programming?
a) Interfaces define the implementation of methods and properties for a class.
b) Interfaces provide a blueprint for creating objects but cannot contain implementation.
c) Interfaces are used to inherit properties and behaviors from another class.
d) Interfaces represent a type of inheritance relationship between classes.

Answer: b) Interfaces provide a blueprint for creating objects but cannot contain implementation.
Explanation: Interfaces define a contract for classes to follow, specifying methods and properties that must be implemented by any class that implements the interface. However, interfaces themselves do not contain any implementation.

7.In object-oriented programming, what is an abstract class?
a) A class that cannot be instantiated and may contain abstract methods.
b) A class that can be instantiated but may contain abstract methods.
c) A class that cannot be inherited but may contain abstract methods.
d) A class that can be inherited but cannot contain abstract methods.

Answer: a) A class that cannot be instantiated and may contain abstract methods.
Explanation: An abstract class is a class that cannot be instantiated on its own and may contain abstract methods, which are declared but not implemented in the abstract class. Abstract classes are designed to be subclassed.

8.Which of the following is not a type of inheritance?
a) Multilevel inheritance
b) Hierarchical inheritance
c) Sequential inheritance
d) Hybrid inheritance

Answer: c) Sequential inheritance
Explanation: Sequential inheritance is not a recognized type of inheritance. The common types include single, multiple, multilevel, hierarchical, and hybrid inheritance.

9.What does the ‘has a’ relationship represent in object-oriented programming?
a) It signifies a relationship between classes where one class is a specialized version of another.
b) It signifies a relationship between classes where one class contains an instance of another.
c) It signifies a relationship between classes where both classes are equivalent.
d) It signifies a relationship between classes where one class is composed of another.

Answer: b) It signifies a relationship between classes where one class contains an instance of another.
Explanation: The ‘has a’ relationship represents a composition relationship between classes, where one class contains an instance of another class.

10.Which of the following is true about multiple inheritance?
a) It allows a class to inherit properties and behaviors from multiple classes.
b) It is supported by most programming languages.
c) It may lead to the diamond problem.
d) All of the above.

Answer: d) All of the above.
Explanation: Multiple inheritance enables a class to inherit properties and behaviors from multiple classes, but it may lead to the diamond problem, where a class inherits from two classes that have a common ancestor, causing ambiguity.

11.Inheritance in object-oriented programming allows for:
a) Reusability of code.
b) Redundancy of code.
c) Limited code functionality.
d) None of the above.

Answer: a) Reusability of code.
Explanation: Inheritance promotes code reusability by allowing new classes to inherit properties and behaviors from existing classes.

12.Which of the following statements about abstract classes is true?
a) Abstract classes cannot have concrete methods.
b) Abstract classes can be instantiated.
c) Abstract classes cannot have constructors.
d) Abstract classes can only have static methods.

Answer: a) Abstract classes cannot have concrete methods.
Explanation: Abstract classes can have both abstract methods (without implementation) and concrete methods (with implementation), but they cannot be instantiated on their own.

13.Which type of inheritance involves a class being derived from multiple base classes?
a) Single inheritance
b) Multiple inheritance
c) Multilevel inheritance
d) Hierarchical inheritance

Answer: b) Multiple inheritance
Explanation: Multiple inheritance involves a class inheriting properties and behaviors from multiple base classes.

14.In association, the relationship between two classes is typically represented by:
a) Solid line with an arrowhead.
b) Dashed line with an arrowhead.
c) Dotted line without arrowheads.
d) Solid line without arrowheads.

Answer: d) Solid line without arrowheads.
Explanation: In association, the relationship between two classes is represented by a solid line without arrowheads.

15.Which type of relationship between classes signifies a “whole-part” relationship?
a) Association
b) Aggregation

c) Inheritance
d) Composition

Answer: d) Composition
Explanation: Composition represents a “whole-part” relationship between classes, where one class contains another class as a part.

16.Interfaces in object-oriented programming are used to:
a) Implement methods and properties.
b) Provide a blueprint for classes.
c) Define constructors.
d) Inherit from other classes.

Answer: b) Provide a blueprint for classes.
Explanation: Interfaces provide a blueprint for classes, specifying methods and properties that must be implemented by any class that implements the interface.

17.What is the main purpose of aggregation?
a) Code reuse.
b) Representing “whole-part” relationships.
c) Achieving polymorphism.
d) Enforcing encapsulation.

Answer: b) Representing “whole-part” relationships.
Explanation: Aggregation is mainly used to represent “whole-part” relationships between classes, where one class contains another class as a part.

18.Which of the following is not a feature of an abstract class?

a) It can have constructors.
b) It can be instantiated.
c) It can have abstract methods.
d) It can have concrete methods.

Answer: b) It can be instantiated.
Explanation: Abstract classes cannot be instantiated on their own; they are meant to be subclassed.

20.What problem may occur with multiple inheritance?
a) Code duplication.
b) Inability to reuse code.
c) Ambiguity in method resolution.
d) Inflexibility in class design.

Answer: c) Ambiguity in method resolution.
Explanation: Multiple inheritance may lead to ambiguity in method resolution, especially in cases where a class inherits from multiple classes that have methods with the same name and signature.

21.Which type of relationship in object-oriented programming indicates a “kind-of” relationship?
a) Association
b) Aggregation
c) Inheritance
d) Composition

Answer: c) Inheritance
Explanation: Inheritance signifies a “kind-of” relationship, where a subclass is a specialized version of its superclass, inheriting its properties and behaviors.

Related posts:

1. Introduction to Information Security
2. Introduction to Information Security MCQ
3. Introduction to Information Security MCQ
4. Symmetric Key Cryptography MCQ
5. Asymmetric Key Cryptography MCQ
6. Authentication & Integrity MCQ
7. E-mail, IP and Web Security MCQ