

Q and A

Here are some questions that interviewers may ask to test your basic knowledge of object oriented design principles.

What is a...

1. **Class:**

- A class is a fundamental building block in object-oriented programming.
- It serves as a blueprint for defining objects in software, containing properties and methods.
- OOP allows for one class to inherit from another, facilitating code reuse.

2. **Properties and Methods:**

- Properties are the attributes (nouns) of a class, defining characteristics like color, size, etc.
- Methods are the behaviors (verbs) of a class, defining actions like fly(), eat(), etc.

3. **Inheritance:**

- Inheritance allows a child class to use the methods and properties of a parent class.
- Child classes inherit behavior from parent classes, facilitating code reuse and promoting a hierarchical structure.
- For example, a Bird class may inherit methods like eat() and sleep() from a parent Animal class, as birds are a type of animal.
 - **Hierarchy Building:**
 - Inheritance allows for the creation of a hierarchy of classes, mirroring real-world relationships.
 - It simulates how objects relate to each other through an IS A relationship.
 - **Understanding IS A Relationship:**
 - The goal of inheritance is not just to inherit methods but to adhere to the IS A relationship rule.
 - It ensures that subclasses represent specialized versions of their superclass and are related in a meaningful way.
 - **Avoiding Poor Design:**
 - Inheritance should not be used indiscriminately for method reuse.
 - For example, a Bird class should not inherit from a Vehicle class solely to use the move() method. This violates the IS A relationship principle and leads to poor design.
 - **Key Principle:**
 - The IS A relationship is crucial for forming class hierarchies and ensuring proper code reuse where it makes sense.
 - Well-designed applications leverage inheritance to promote code reuse in a meaningful and logical manner.

4. **Public vs. Private Methods/Properties:**

- Public methods and properties are accessible from external classes, while private methods and properties are only accessible within the class itself.
- Private members cannot be inherited by child classes.
- Inner nested classes within the same class definition have access to private members.

5. **Class Constructor:**

- A constructor is a method used to instantiate an instance of a class.
- It typically has the same name as the class and initializes specific properties.
- Overloaded constructors allow for multiple constructor definitions with different parameters.

6. **Overloaded Methods:**

- Overloaded methods have the same name but a different set of parameters.
- The order of parameters matters, and parameters must have different data types to avoid compilation errors.

7. **Abstract Class:**

- An abstract class cannot be instantiated but can be inherited.
- It contains abstract methods that must be implemented by its child classes.
- Abstract methods are declared without implementation and must be implemented by subclasses.

8. **Instantiation:**

- Instantiation refers to the creation of an instance (object) of a class.
- It occurs when the class constructor method is called, providing the object with properties and methods defined in the class blueprint.

9. **Passing Parameters:**

- Passing parameters by value allows methods to access only the parameter's value, not the variable itself.
- Passing parameters by reference passes a pointer to the variable, allowing methods to modify the original variable.

10. **Method Overriding:**

- Method overriding occurs when a subclass provides its own implementation for a method inherited from a superclass.
- It allows for customizing the behavior of inherited methods in child classes.

11. **Exception Handling:**

- Exception handling is a process used to handle errors that occur during program execution.
- It allows for graceful error handling, preventing the application from crashing and providing users with feedback on how to proceed.

12. **"self" Object:**

- The "self" reference refers to the current instance of the object within a class.
- It is used to access instance variables and methods within the class.

13. **Static Methods:**

- Static methods exist independently of class instances and do not have access to instance variables.
- They are useful for defining utility functions that do not require access to instance-specific data.

When to use functional vs object oriented solutions?

- Functions are action based, and classes are state based
- Functions easier to write unit tests
- object oriented solution is better for real world simulation, or needing to keep states

What are dunder methods?

Revision #6

Created 27 February 2024 00:05:02 by victor

Updated 5 March 2024 01:09:19 by victor