Q. Types of classes for modelling the real world object.

ChatGPT

In object-oriented programming, there are several types of classes used for modelling real-world objects. These classes capture different aspects of the objects they represent. Here are some common types of classes used for modelling real-world objects:

Entity Classes:

Entity classes represent objects that have a distinct identity and can exist independently. These classes model real-world entities with well-defined boundaries and individuality.

Example: Person, Car, Book, Customer, etc.

Value Classes:

Value classes represent objects that are defined solely by their value rather than identity. They are immutable, and two instances with the same value are considered equal.

Example: Date, Time, Money, etc.

Service Classes:

Service classes model behavior or actions rather than representing physical objects. They contain methods that perform specific tasks.

Example: FileReader, EmailService, PaymentProcessor, etc.

Controller Classes:

Controller classes manage the flow of data and communication between other classes. They often act as intermediaries, coordinating the interactions of different classes.

Example: OrderController, UserInputController, etc.

Utility Classes:

Utility classes contain helper methods or functions that provide commonly used functionalities, but they do not represent real-world objects themselves.

Example: MathUtility, StringUtils, etc.

Abstract Classes:

Abstract classes cannot be instantiated on their own but provide a blueprint for other classes to inherit from. They often contain abstract (pure virtual) methods that must be implemented by their derived classes.

Example: Shape (with pure virtual methods like calculateArea() and calculatePerimeter()), which serves as a base for classes like Circle, Square, etc.

Singleton Classes:

Singleton classes ensure that only one instance of the class can exist throughout the program. They are useful when you need a single point of access to a particular resource or functionality.

Example: Logger, ConfigurationManager, etc.

Aggregate Classes:

Aggregate classes represent a collection of objects or a whole-part relationship. They encapsulate a group of related objects and manage their interactions.

Example: Library (containing multiple Book objects), Company (containing multiple Employee objects), etc.