**Инкапсуляция** - способность объектов скрывать часть своего состояния и поведения.

**Наследование** - повторное использование кода, позволяющее на основе одного класса, создать другой.

**Полиморфизм** - способность объекта изменять свое поведение в зависимости от контекста его использования.

**Абстракция** - выделение основных характеристик объекта в наружный контекст и сокрытие второстепенных деталей.

**Класс** –Шаблон для создания типа объектов.

**Конструктор** - метод который вызывается при создании объекта для инициализации свойств и полей этого класса.

**Шаблон** описывает тип объекта.

**Интерфейс** - перечень сигнатур методов класса, предназначенных для реализации.

**Паттерн проектирования** - схема решения часто встречаемой задачи, прием программирования

**Поле** — это переменная, определенная в классе. Поля могут иметь различные модификаторы доступа (например, `private`, `public`) и хранят данные, связанные с экземпляром класса или статические данные для самого класса.

**Свойство** — это механизм, позволяющий контролировать доступ к полям. Свойства могут иметь методы доступа (`get` и `set`), которые позволяют читать и записывать значения полей, а также включать логику при этих операциях. Это делает свойства более гибкими и безопасными по сравнению с полями.

**События в C#** — это механизм, позволяющий объектам сообщать другим объектам о произошедших изменениях или действиях.

**Индексаторы** позволяют объектам вести себя как массивы, что упрощает доступ к элементам коллекций. Индексаторы определяются с помощью ключевого слова `this` и могут принимать один или несколько параметров. Индексаторы позволяют обращаться к элементам коллекции, как если бы они были элементами массива.

**UML (Unified Modeling Language)** — это стандартизированный язык для визуального моделирования систем. UML позволяет создавать диаграммы, которые описывают архитектуру, компоненты и взаимодействие различных частей системы.

**Стереотип в UML** — это расширение стандартной семантики модели. Он позволяет добавлять дополнительные характеристики элементам модели, обозначая их особые свойства. Стереотипы могут использоваться для создания пользовательских типов, которые помогают лучше описывать специфические аспекты системы. Например, можно создать стереотип `<<interface>>`, чтобы обозначить, что класс является интерфейсом.