\*\*Принципы объектно-ориентированного программирования SOLID и STUPID:\*\*

\*\*SOLID\*\* - это аббревиатура для пяти принципов объектно-ориентированного проектирования, разработанных Майклом Фэзерсом:

1. \*\*SRP (Single Responsibility Principle)\*\* - Принцип единой ответственности: Каждый класс должен иметь одну причину для изменения.

2. \*\*OCP (Open/Closed Principle)\*\* - Принцип открытости/закрытости: Сущности (классы, модули, функции и т. д.) должны быть открыты для расширения, но закрыты для изменения.

3. \*\*LSP (Liskov Substitution Principle)\*\* - Принцип подстановки Лисков: Объекты в программе могут быть заменены их подтипами без изменения правильности программы.

4. \*\*ISP (Interface Segregation Principle)\*\* - Принцип сегментации интерфейса: Клиенты не должны зависеть от интерфейсов, которые они не используют.

5. \*\*DIP (Dependency Inversion Principle)\*\* - Принцип инверсии зависимостей: Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

\*\*STUPID\*\* - это неофициальное упрощенное правило для проектирования программного обеспечения:

1. \*\*STUPID\*\* - не делайте вещи сложнее, чем необходимо.

\*\*Класс Object. Реализация его методов по умолчанию.\*\*

В Java, `Object` - это корневой класс для всех классов. Методы по умолчанию включают `equals()`, `hashCode()`, `toString()`, `getClass()`, `wait()`, `notify()`, и `notifyAll()`. Эти методы обычно переопределяются в наследуемых классах для предоставления поведения, которое соответствует классу.

\*\*Простое и множественное наследование. Особенности реализации наследования в Java.\*\*

Java поддерживает однородное наследование, что означает, что класс может наследовать только один класс напрямую. Однако интерфейсы позволяют множественное наследование, поскольку класс может реализовывать несколько интерфейсов. Особенностью Java в реализации наследования является то, что приватные члены базового класса не наследуются.

\*\*Понятие абстрактного класса. Модификатор abstract.\*\*

Абстрактный класс - это класс, который не может быть инстанцирован напрямую и используется в основном для определения общего интерфейса или поведения, которое должно быть реализовано в производных классах. Модификатор `abstract` используется для объявления абстрактного класса или метода.

\*\*Понятие интерфейса. Реализация интерфейсов в Java. Отличие интерфейсов от абстрактных классов.\*\*

Интерфейс в Java - это referece тип, который может содержать только public методы и переменные-члены, которые по умолчанию являются `public static final`. Интерфейсы используются для определения контракта, который должны реализовать классы. Отличие от абстрактных классов заключается в том, что интерфейсы не могут содержать реализацию методов (до Java 8), и классы могут реализовывать несколько интерфейсов.

\*\*Модификаторы default, static и private для методов интерфейса.\*\*

- \*\*default\*\*: Методы с модификатором default могут иметь тело и могут быть реализованы в классах, реализующих интерфейс.

- \*\*static\*\*: Статические методы интерфейса не могут быть переопределены в классах, реализующих интерфейс, и могут быть вызваны напрямую через интерфейс.

- \*\*private\*\*: Приватные методы интерфейса существуют только для использования в других методах интерфейса.

\*\*Перечисляемый тип данных (enum) в Java. Особенности реализации и использования.\*\*

`enum` в Java - это специальный тип класса, который позволяет создавать ограниченный набор констант. Особенности включают инициализацию полей, методы, и реализацию интерфейсов. Перечисления могут быть использованы для представления фиксированных наборов, таких как дни недели или месяцы года.

\*\*Тип запись (record) в Java. Особенности использования.\*\*

Тип записи (record) - это новый тип, введенный в Java 14, который предназначен для представления данных. Особенности включают автоматическое создание методов `equals()`, `hashCode()`, и `toString()`, а также сокращенное определение конструктора и полей.

\*\*Методы и поля с модификаторами static и final.\*\*

- \*\*static\*\*: Статические методы и поля принадлежат классу, а не инстансу класса. Они могут быть вызваны без создания объекта класса.

- \*\*final\*\*: Методы с модификатором final не могут быть переопределены в производных классах. Поля, помеченные как final, должны быть инициализированы при объявлении или в конструкторе.

\*\*Перегрузка и переопределение методов.\*\*

- \*\*Перегрузка (Overloading)\*\*: Это когда несколько методов в одном классе имеют разные сигнатуры (разное количество и/или типы аргументов).

- \*\*Переопределение (Overriding)\*\*: Это когда метод в производном классе имеет ту же сигнатуру, что и метод в базовом классе, и может изменять или добавлять поведение метода.

\*\*Обработка исключительных ситуаций, три типа исключений.\*\*

В Java есть три типа исключений:

1. \*\*Checked Exceptions\*\*: Исключения, которые должны быть обработаны программистом или объявлены в сигнатуре метода.

2. \*\*Unchecked Exceptions\*\*: Исключения, которые необязательно обрабатывать, например, `NullPointerException` или `ArrayIndexOutOfBoundsException`.

3. \*\*Errors\*\*: Серьезные проблемы, связанные с JVM, такие как `OutOfMemoryError`, которые обычно не обрабатываются.

\*\*Стандартный массив и динамический массив (ArrayList). Основные различия.\*\*

- \*\*Стандартный массив\*\* имеет фиксированную длину и хранит объекты одного типа. Доступ к элементам осуществляется через индексы.

- \*\*ArrayList\*\* - это динамический массив, который может изменять свою длину. Он реализует `List` интерфейс и предоставляет методы для добавления, удаления и поиска элементов.

\*\*Вложенные, локальные и анонимные классы.\*\*

- \*\*Вложенные классы\*\* - классы, определенные внутри других классов, но вне методов.

- \*\*Локальные классы\*\* - классы, определенные внутри методов.

- \*\*Анонимные классы\*\* - классы без имени, определенные в процессе создания экземпляра. Они часто используются для реализации интерфейсов или наследования классов для предоставления альтернативного поведения.