# Основные термины и фразы в Java

### Структура Java программы

Структура Java программы:

* примитивные типы
* операторы (инструкции)
* массивы
* статические методы позволяют нам инкапсулировать и повторно использовать код и разрабатывать программы как набор независимых модулей.
* строки
* input/output - взаимодействие между программами и внешним миром
* абстракция данных: расширяет обособление (encapsulation) и повторное использование кода (reuse), что позволяет определить не примитивные типы, поддерживая ООП.

### Абстракция данных

Абстрактный тип данных - тип данных, представление которого скрыто от клиента. Думай об ADT как об абстрактном понятии (не путайте его с абстрактным классом в Java), которое представляет некоторую модель данных, структура которой определяет только основные характеристики данных и операции, которые можно над ними выполнять. Например, Interfaces List, Map и т.д. То есть для клиента хорошими примерами буквально могут быть интерфейсы, с какими данными они работают и операции доступные над этими данными.

Реализациями АТД могут быть: определение типа данных или библиотека статических методов (функций). Поэтому стоит различать метод экземпляра и статический метод. Метод инкапсулирует вычисление, которое определено как последовательность операторов.

Реализация АТД в виде Java-класса (определение типа данных) - это практически построение библиотеки статических методов с учетом того, что мы связываем данные с реализациями функций и скрываем представление данных от клиента. Создание АТД включает понимание и разработку API.

Из всего вышеперечисленного вытекает, что определение типа данных - реализация в виде класса, у которого могут быть объекты, а эти объекты могут содержать значения этого типа данных :) У класса со статическими методами не может быть экземпляров.

API - это способ документирования или описания поведения АТД. API отделяет клиента от реализации: клиент ничего не знает о реализации, а реализация не учитывает особенностей клиента.

### Инкапсуляция и модульное программирование

Инкапсуляция - это способ в языке программирования поместить Типы Данных внутри их реализаций и разграничить реализации клиентов и Типов Данных. Инкапсуляция изолирует операции с ТД.

Инкапсуляция делает возможным модульное программирование:

* позволяет независимо разрабатывать код клиента и код реализации
* позволяет подставлять улучшенные реализации без влияния на код клиента.
* позволяет поддерживать еще ненаписанные программы.

Парадигма модульного программирования (modular programming) - метод или способ мышления, который применяется к программам и состоит в том, чтобы разбивать большие ПО на малые и независимые модули.

Библиотеки статических методов позволяют осуществлять модульное программирование, где мы создаем библиотеки статических методов (модули) и статический метод в одной библиотеке могут вызывать статические методы, определенные в других библиотеки. Это позволяет нам:

* Работать с модулями разумного размера, даже в программе с участием большого количество кода
* Раздавать и повторно использовать код без необходимости его повторной реализации.
* Легко заменять улучшенные реализации.
* Разрабатывать подходящие абстрактные модели для решения проблем программирования
* Локализовать отладку (**unit testing**)

### Наследование

**fragile base class problem** - фундаментальная проблема ООП, которая состоит в том что производные классы сильно зависят от базового класса и при изменении реализации последнего могут возникнуть ошибки в его наследниках.