Pro Java №14

Функциональный интерфейс в Java - это интерфейс, который содержит только один абстрактный метод. Такой интерфейс может иметь и другие методы по умолчанию или статические методы, но должен содержать только один абстрактный метод.

В Java можно обратиться к функциональному интерфейсу, используя лямбда-выражения или ссылки на методы без создания промежуточного класса для его реализации.

Анонимные классы в Java позволяют создавать экземпляры классов на лету без необходимости создания отдельного класса.

Лямбда-выражения в Java позволяют сократить код по сравнению с анонимными классами и делают его более читабельным и выразительным.

В Java существует аннотация `@FunctionalInterface`, которая используется для указания, что интерфейс является функциональным. Эта аннотация сигнализирует компилятору о том, что интерфейс должен содержать только один абстрактный метод.

В Java имеется несколько встроенных функциональных интерфейсов, которые широко используются для обработки данных и функционального программирования.

1. **`Function<T, R>`**: Применяет функцию к объекту типа `T` и возвращает результат типа `R`.

2. **`Consumer<T>`**: Принимает объект типа `T` и не возвращает результат. Используется для операций, которые выполняются с объектом, но не производят результат.

3. **`Supplier<T>`**: Возвращает объект типа `T`. Используется для генерации или поставки объектов.

4. **`Predicate<T>`**: Проверяет условие для объекта типа `T` и возвращает `boolean`. Обычно используется для фильтрации данных.

5. **`BinaryOperator<T>`**: Применяет операцию к двум объектам типа `T` и возвращает результат типа `T`. Это специализированный вариант `BiFunction<T, T, T>`.

6. **`BiConsumer<T, U>`**: Принимает два аргумента типов `T` и `U` и не возвращает результат.

7. **`BiFunction<T, U, R>`**: Применяет функцию к двум объектам типов `T` и `U` и возвращает результат типа `R`.