Абстрактные классы abstract

<https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html>

Абстрактные классы по сравнению с интерфейсами.

Абстрактные классы похожи на интерфейсы.

Мы не можем создавать их экземпляры, и они могут содержать сочетание методов, объявленных с реализацией или без нее.

Однако с помощью абстрактных классов можно объявлять поля, которые не являются статическими и окончательными, и определять общедоступные, защищенные и частные конкретные методы.

В интерфейсах все поля автоматически становятся общедоступными, статическими и конечными, а все методы, которые вы объявляете или определяете (как методы по умолчанию), являются общедоступными. Кроме того, можно расширить только один класс, независимо от того, абстрактный он или нет, тогда как можно реализовать любое количество интерфейсов.

Когда мы должны понимать : что использовать? абстрактные классы или интерфейсы?

Необходимо рассмотреть возможность использования абстрактных классов, если какое-либо из нижеуказаных утверждений применимо к нашей ситуации:

1. Вы хотите разделить код между несколькими тесно связанными классами.
2. Вы ожидаете, что классы, которые расширяют ваш абстрактный класс, имеют много общих методов или полей или требуют модификаторов доступа, отличных от общедоступных (таких как protected и private).
3. Вы хотите объявить нестатические или неконечные поля. Это позволяет вам определять методы, которые могут получать доступ и изменять состояние объекта, которому они принадлежат.

Необходимо рассмотреть возможность использования интерфейсов, если какое-либо из этих утверждений применимо к нашей ситуации:

1. Вы ожидаете, что несвязанные классы будут реализовывать ваш интерфейс. Например, интерфейсы Comparable и Cloneable реализуются многими несвязанными классами.
2. Вы хотите указать поведение определенного типа данных, но не заботитесь о том, кто реализует его поведение.
3. Вы хотите воспользоваться преимуществами множественного наследования типов.

Интерфейсы

<http://msdn.microsoft.com/ru-ru/library/ms173156.aspx>

В программной инженерии бывает ряд ситуаций, когда разрозненным группам программистов важно договориться о «контракте», в котором прописано, как взаимодействует их программное обеспечение. Каждая группа должна иметь возможность писать свой код, не зная, как написан код другой группы. Вообще говоря, интерфейсы являются такими контрактами.

Интерфейсы в Java

В языке программирования Java интерфейс — это ссылочный тип, аналогичный классу, который может содержать только константы, сигнатуры методов, методы по умолчанию, статические методы и вложенные типы. Тела методов существуют только для методов по умолчанию и статических методов. Интерфейсы не могут быть созданы — они могут быть реализованы только классами или расширены другими интерфейсами. Расширение обсуждается позже в этом уроке.

Интерфейсы как API

API распространены в коммерческих программных продуктах. Как правило, компания продает программный пакет, содержащий сложные методы, которые другая компания хочет использовать в своем программном продукте. Примером может служить пакет методов цифровой обработки изображений, который продается компаниям, производящим графические программы для конечных пользователей. Компания по обработке изображений пишет свои классы для реализации интерфейса, который она делает общедоступным для своих клиентов. Затем графическая компания вызывает методы обработки изображения, используя сигнатуры и возвращаемые типы, определенные в интерфейсе. В то время как API компании, занимающейся обработкой изображений, становится общедоступным (для ее клиентов), ее реализация API хранится в строго охраняемом секрете — фактически, она может пересмотреть реализацию позже, пока она продолжает реализовывать исходный интерфейс. на что полагаются его клиенты.