**Как программировать на java**

* Исходные тексты хранятся в текстовом виде в файле .java
* Файл java компилируется в файл .class
* Этот файл содержит байт-код (инструкция для автоматического выполнения)
* Байт-код интерпретируется во время выполнения

Just-In\_time (компилятор)

* Компилирует байт-код в исполняемый для конкретной платформы
* Увеличивает производительность
* Оптимизирует повторяющийся код, например, циклы

Java-приложения

* Клиентские
* JVM выполняет отдельное приложение из командой строки
* Классы загружаются с локального диска
* Серверные
* Обслуживают несколько клиентов
* Применяются для многозвенных приложений

Варианты поставки

* J2ME (Micro edition) – для мобильных устройств
* J2SE (Standard Edition) – разработка обычных приложений
* J2EE (Enterprise Edition) – разработка приложений многозвенной архитектуры

Ключевые компоненты SDK

* Компилятор (javac) – создает из исходного кода байт-код
* Интерпретатор (java) – выполняет байт-код

Пакеты

Классы объединяются в специальные структуры, называемые пакетами

* Стандартные пакеты для
* Поддержки базовых конструкций языка (java.lang)
* Создания оконного интерфейса (javax.swig)
* Управления вводом/выводом (java.io)

Структура исходного файла класса Java

* Исходный файл состоит из следующих частей
* Необязательное слово package, за которым следует наименование пакета, в котором содержится класс
* Необязательный оператор import (может быть несколько), который указывает, какие классы из сторонних пакетов используются создаваемым классом
* Одно или более определение class или interface, за которым следует программный блок
* Файл должен иметь одно и то же имя, что и создаваемый класс
* Ключевые слова языка Java чувствительны к регистру
* В файле может быть только один public класс

Соглашение об именовании

Имена файлов (Customer.java, Preson.java)

Имена пакетов (java.util, javax.swing)

Имена классов (Customer, Person)

Имена свойств классов (firstName, Id)

Имена методов (getName, IsAlive)

Имена констант (SQUARE\_SIZE)

Также могут использовать цифры 1…9, \_, $

Именование переменных

* Имя переменной должен начинаться с буквы, знака подчеркивания или со знака $
* Имя переменной может включать цифры
* Давайте переменным осмысленные имена

Простые типы данных

Восемь простых типов данных

* Шесть числовых
* Символьный
* Логический

Определяемые пользователем типы

* Классы
* Интерфейсы
* Массивы

**Абстрактные классы**

Фундаментальная разница между интерфейсом и абстрактным классом заключается в том, что интерфейс определяет только поведение. Он не сообщает ничего про объект, который будет его реализовывать.

Абстрактный класс описывает некий абстрактный объект, а не только его поведение.

Интерфейс и абстрактный класс понятия не взаимозаменяемые, если нужное поведение – необходимо использовать интерфейс. Если речь про концептуальный объект – мы должны использовать класс.

Синтаксис создания абстрактного класса

public abstract class MyAbstractClass {  
 *//Поля и конструкторы  
 //Абстрактные методы  
 //Методы с реализации*}

Синтаксис использования

public class myClass extends MyAbstractClass {  
 *//Реализация абстрактных методов  
 //иной код*}

Класс может одновременно наследоваться от абстрактного класс (только одного) и реализовывать один или множество интерфейсов

Интерфейс не может реализовывать интерфейс, не может наследовать абстрактный класс, но может наследовать множество других интерфейсов.

Абстрактный класс может наследовать како обычный класс, так и абстрактный. В обоих случаях это будет только одни класс.

Абстрактный класс может реализовывать до 65535 интерфейсов.