**РЕФЕРАТ**

Разработана программная система автоматизации оценки недвижимости, способная реализовывать таблицы для расчетов различными способами стоимости объектов оценки. Программно реализована связь пользователя с внешними базами данных, что значительно упростит поиск аналогов, необходимый для профессиональной оценки стоимости объекта. Результатом выполнения программы является сгенерированные таблицы, в которых содержатся все необходимые данные необходимые для повышения эффективности работы оценщика, а значит, повысит производительность оценочной фирмы.

**ANNOTATION**

Capable to realize the table the program system of automation of an estimation of the real estate is developed for calculations by various ways of cost of objects of an estimation. Program communication of the user with external databases is realized that considerably will simplify the search of analogs necessary for professional estimation of cost of object. Result of performance of the program is the generated tables in which all necessary data necessary for increase of overall performance of the appraiser so will raise productivity of estimated firm contains.

Оглавление

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 9](#_Toc317026738)

[1. ВВЕДЕНИЕ 10](#_Toc317026739)

[2. Введение в предметную область 12](#_Toc317026740)

[2.1 Виды стоимости 12](#_Toc317026741)

[2.2 Основные подходы оценки недвижимости 13](#_Toc317026742)

[2.3 Основные методы оценки недвижимости 14](#_Toc317026743)

[3. Постановка задачи дипломного проекта и выбор языка программирования 17](#_Toc317026744)

[3.1 Постановка задачи дипломного проекта 17](#_Toc317026745)

[3.2 Выбор языка программирования 17](#_Toc317026746)

[4. Особенности создания программ на объектно-ориентированном языке программирования Java 19](#_Toc317026747)

[4.1 Основные особенности языка 19](#_Toc317026748)

[4.2 Классификация платформ Java 24](#_Toc317026749)

[4.3 Java и Microsoft 25](#_Toc317026750)

[4.4 Применения платформы Java 25](#_Toc317026751)

[4.5 Производительность 25](#_Toc317026752)

[4.6 Основные возможности 26](#_Toc317026753)

[4.7 Пример программы 26](#_Toc317026754)

[4.8 Примитивные типы 28](#_Toc317026755)

[4.9 Объектные переменные, объекты, ссылки и указатели 30](#_Toc317026756)

[4.10 Классы и функции 32](#_Toc317026757)

[4.11 Библиотеки классов 36](#_Toc317026758)

[4.12 Средства разработки ПО 36](#_Toc317026759)

[5. БД и ее связь с Java 37](#_Toc317026760)

[5.1 История 38](#_Toc317026761)

[5.2 Виды баз данных 39](#_Toc317026762)

[5.3 Классификация по модели данных 39](#_Toc317026763)

[5.4 Классификация по среде постоянного хранения 40](#_Toc317026764)

[5.5 Классификация по содержимому 40](#_Toc317026765)

[5.6 Классификация по степени распределённости 40](#_Toc317026766)

[5.7 Другие виды БД 41](#_Toc317026767)

[5.8 Сверхбольшие базы данных 41](#_Toc317026768)

[5.9 JDBC - средство общения между Java и базами данных 42](#_Toc317026769)

[6. Создание сайта при помощи html и css 47](#_Toc317026770)

[6.1 Возможности HTML 47](#_Toc317026771)

[6.1.1 Общее представление 47](#_Toc317026772)

[6.1.2 Браузеры 48](#_Toc317026773)

[6.1.3 Версии 48](#_Toc317026774)

[6.1.4 Перспективы 49](#_Toc317026775)

[6.1.5 Структура HTML-документа 49](#_Toc317026776)

[6.2 Основные принципы использования CSS 50](#_Toc317026777)

[6.2.1 Цель создания CSS 51](#_Toc317026778)

[6.2.2 Способы подключения CSS к документу 51](#_Toc317026779)

[6.2.3 Иерархия элементов внутри документа 53](#_Toc317026780)

[6.2.4Построение правила CSS 53](#_Toc317026781)

[6.2.5 Классы элементов. Идентификаторы элементов. 55](#_Toc317026782)

[6.2.6 Наследование. Каскадирование. Приоритеты стилей CSS. 55](#_Toc317026783)

[6.2.7 Пример таблицы стилей 57](#_Toc317026784)

[6.2.8 CSS-вёрстка 58](#_Toc317026785)

[6.2.9 История создания и развития CSS 59](#_Toc317026786)

[6.2.10 Уровень 1 (CSS1) 60](#_Toc317026787)

[6.2.11 Уровень 2 (CSS2) 60](#_Toc317026788)

[6.2.12 Уровень 3 (CSS3) 61](#_Toc317026789)

[6.2.13 Поддержка CSS браузерами 61](#_Toc317026790)

[6.2.14 Различные блоковые модели 61](#_Toc317026791)

[6.2.15 CSS-фильтры 62](#_Toc317026792)

[6.2.16 Безопасность 62](#_Toc317026793)

[6.2.17 CSS Framework 62](#_Toc317026794)

[6.2.18 Расширения CSS 63](#_Toc317026795)

[7. Создание программного комплекса, используя возможности выбранных языков программирования 64](#_Toc317026796)

[7.1 Реализация интерфейса пользовательского приложения 64](#_Toc317026797)

[7.2 Реализация информационного блока программы: 66](#_Toc317026798)

[7.3 Реализация меню «Настройки» 67](#_Toc317026799)

[7.4 Реализация меню «аналоги» 68](#_Toc317026800)

[7.5 Реализация меню «Фаил» 74](#_Toc317026801)

[8. Технико-экономическое обоснование проекта 78](#_Toc317026802)

[8.1 Концепция экономического обоснования научно-технической разработки 78](#_Toc317026803)

[8.2 Потребительские свойства продукта, связанного с совершенствованием продукта 80](#_Toc317026804)

[8.3 Рынок и план маркетинга 81](#_Toc317026805)

[8.4 Производство продукта 85](#_Toc317026806)

[8.5 Организационный план проекта 89](#_Toc317026808)

[8.6 Прогноз финансовых показателей проекта 90](#_Toc317026809)

[8.7 Экономическая эффективность 96](#_Toc317026810)

[8.8 Вывод 97](#_Toc317026811)

[9. Охрана интеллектуальной собственности 99](#_Toc317026812)

[9.1 Программа для ЭВМ как объект правовой охраны 99](#_Toc317026813)

[9.2 Официальная регистрация программ для ЭВМ 101](#_Toc317026814)

[9.3 Коммерческая реализация программ для ЭВМ 105](#_Toc317026815)

[ЗАКЛЮЧЕНИЕ 110](#_Toc317026816)

[СПИСОК ЛИТЕРАТУРЫ 112](#_Toc317026817)

[ПРИЛОЖЕНИЕ 113](#_Toc317026818)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

|  |  |  |
| --- | --- | --- |
| БД | - | база данных |
| ЖЦ | - | жизненный цикл |
| ИС | - | информационная система |
| КМ | - | концептуальная модель |
| НИР | - | научно – исследовательская работа |
| П (P) | - | Показатель |
| ПО | - | предметная область |
| РИД | - | результат интеллектуальной деятельности |
| СУБД | - | система управления базами данных |
| ЭД | - | элемент данных |
| ЭС | - | экспертная система |
| ЭВМ | - | электронная вычислительная машина |
| ER-модель | - | модель типа “сущность-связь” |

# 1. ВВЕДЕНИЕ

В настоящее время, особенно в крупных мегаполисах, постоянно совершаются операции над недвижимостью, но чтобы продать, обменять, сдать в аренду и даже разрешить спор в судебном порядке, необходимо конкретно выяснить стоимость объекта. На оценку влияет множество факторов, начиная от простых размеров объекта, удаленности от центра и заканчивая экономической ситуацией в целом. Работа оценщика подразумевает огромное количество вычислительной работы, поиск аналогов объекта и т.д..

Целью создания дипломного проекта является программа, предназначенная для определения рыночной стоимости объектов недвижимости и земельных участков с последующим составлением отчета. Использование современных технологий, Java, SQL обеспечит быстрое и надежное функционирование объекта, а главное обеспечит легкость интегрирования в различные системы, за счет гибкой системы настройки под критерии пользователя, доступности и распространенности.

В рамках настоящего дипломного проекта разработана программа для автоматизации основных аспектов оценки недвижимости. Данная программа обеспечивает удобное управление расчетами, в том числе калькуляциями.

Описание работ по разработке программы автоматизации оценки недвижимости:

* Формулирование целей и требований к проекту.   
  На данном этапе идет анализ тех частей работы оценщика, которые подразумевают монотонную работу, расчеты и т.д.
* Введение в предметную область.

На данном этапе идет изучение предметной области, ознакомление с материалами (документами, экономическими расчетами), при необходимости консультации специалиста данной предметной области.

* Изучение литературы.

На данном этапе идет изучение функциональных возможностей различного рода программных сред, их взаимодействие с различного вида системами.

* Анализ требований к программному комплексу.

На данном этапе производится анализ основных, функциональных, системных и других требований к программному комплексу.

* Разработка концептуальных моделей.

На данном этапе идет разработка архитектуры системы, подсистем, а так же принципы их взаимодействия.

* Разработка графического интерфейса.

На данном этапе идет разработка графического интерфейса.

* Тестирование программного продукта.

На данном этапе проводится тестирование работы отдельных модулей и системы в целом.

* Оформление проекта, разработка документации, отладка

На данном этапе идет оформление проекта, разработка технической документации, руководства пользователя и отладка.

# 2. Введение в предметную область

**Оценка стоимости недвижимости** — процесс определения рыночной стоимости объекта или отдельных прав в отношении оцениваемого объекта недвижимости. Оценка стоимости недвижимости включает: определение стоимости права собственности или иных прав, например, права аренды, права пользования и т. д. в отношении различных объектов недвижимости.

|  |
| --- |
|  |

## 2.1 Виды стоимости

В оценочной практике существуют различные [виды стоимости](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D0%BE%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_%D0%BD%D0%B5%D0%B4%D0%B2%D0%B8%D0%B6%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1). Стоимость недвижимости подразделяется на следующие виды:

* рыночная стоимость;
* инвестиционная стоимость;
* ликвидационная стоимость;
* кадастровая стоимость.

При определении **рыночной стоимости объекта оценки** определяется наиболее вероятная цена, по которой объект оценки может быть отчужден на дату оценки на открытом рынке в условиях конкуренции, когда стороны сделки действуют разумно, располагая всей необходимой информацией, а на величине цены сделки не отражаются какие-либо чрезвычайные обстоятельства, то есть когда:

* одна из сторон сделки не обязана отчуждать объект оценки, а другая сторона не обязана принимать исполнение;
* стороны сделки хорошо осведомлены о предмете сделки и действуют в своих интересах;
* объект оценки представлен на открытом рынке посредством публичной оферты, типичной для аналогичных объектов оценки;
* цена сделки представляет собой разумное вознаграждение за объект оценки и принуждения к совершению сделки в отношении сторон сделки с чьей-либо стороны не было;
* платеж за объект оценки выражен в денежной форме.

При определении **инвестиционной стоимости объекта оценки** определяется стоимость для конкретного лица или группы лиц при установленных данным лицом (лицами) инвестиционных целях использования объекта оценки — [инвестиционной недвижимости](http://ru.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D0%B2%D0%B5%D1%81%D1%82%D0%B8%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B4%D0%B2%D0%B8%D0%B6%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C&action=edit&redlink=1) для получения дохода в виде арендной платы и прироста стоимости. При определении инвестиционной стоимости, в отличие от определения рыночной стоимости, учет возможности отчуждения по инвестиционной стоимости на открытом рынке не обязателен.

При определении **ликвидационной стоимости объекта оценки** определяется расчетная величина, отражающая наиболее вероятную цену, по которой данный объект оценки может быть отчужден за срок экспозиции объекта оценки, меньший типичного срока экспозиции для рыночных условий, в условиях, когда продавец вынужден совершить сделку по отчуждению имущества. При определении ликвидационной стоимости, в отличие от определения рыночной стоимости, учитывается влияние чрезвычайных обстоятельств, вынуждающих продавца продавать объект оценки на условиях, не соответствующих рыночным.

При определении **кадастровой стоимости объекта оценки** определяется методами массовой оценки рыночная стоимость, установленная и утвержденная в соответствии с законодательством, регулирующим проведение кадастровой оценки. Кадастровая стоимость определяется оценщиком, в частности, для целей налогообложения.

### 2.2 Основные подходы оценки недвижимости

#### 2.2.1 Доходный подход

Доходный подход является одним из наиболее применяемых методов оценки недвижимости в оценочной практике на сегодняшний день.

Суть метода — доходный подход оценивает стоимость недвижимости, как текущую стоимость будущих денежных потоков. При этом данный подход отражает уровень риска для оцениваемого объекта недвижимости, а также качество и количество дохода, который сможет принести объект оценки в течение срока службы.

При расчёте доходным подходом используются следующие методы: метод капитализации доходов и метод дисконтирования денежных потоков.

#### 2.2.2 Затратный подход

Затратный подход (подход на основе активов) — совокупность методов оценки, в которых стоимость объекта равна сумме стоимости земельного участка и стоимости воспроизводства (замещения) всех улучшений, за вычетом накопленного износа, и стоимости обязательств, то есть стоимость объекта оценки зависит от стоимости создания аналогичного объекта. Данный подход применяется для оценки отдельно стоящих зданий, домовладений.

#### 2.2.3 Сравнительный подход

Сравнительный подход — совокупность методов оценки стоимости объекта оценки, основанных на сравнении объекта оценки с объектами — аналогами объекта оценки, в отношении которых имеется информация о ценах. Объектом — аналогом объекта оценки для целей оценки признается объект, сходный объекту оценки по основным экономическим, материальным, техническим и другим характеристикам, определяющим его стоимость. (см. Федеральный стандарт оценки № 1)

## 2.3 Основные методы оценки недвижимости

#### 2.3.1 Метод капитализации доходов

Суть метода капитализации доходов — определение стоимости объекта недвижимости происходит путём преобразования *годового чистого операционного дохода* (ЧОД) в текущую стоимость.

Базовая формула:

*C* = *NOI* / *Rk*

где

*C* — стоимость объекта недвижимости (ден.ед)

*NOI* — Чистый операционный доход или ЧОД ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Net operation income (NOI)*)

*Rk* — Коэффициент капитализации

*Коэффициент капитализации* — показатель отражающий отношение ожидаемого годового дохода (ЧОД) от стоимости недвижимости:

*Rk* = *Rn* + *Nv*

где

*Rk* — коэффициент капитализации

*Rn* — ставка доходности инвестора на вложенный капитал — ожидаемая инвестором норма доходности от инвестирования с учётом риска на рынке.

*Nv* — норма возврата капитала — процентная ставка которая обеспечивает возврат первоначальных инвестиций. Ставка разделяет доход на две части: возмещение капитальных вложений в недвижимость и получение дохода от владения объектом.

В оценочной практике используются 3 метода расчёта нормы возврата капитала, [метод Ринга](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%A0%D0%B8%D0%BD%D0%B3%D0%B0&action=edit&redlink=1), [метод Инвуда](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%98%D0%BD%D0%B2%D1%83%D0%B4%D0%B0&action=edit&redlink=1) и [метод Хоскольда](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%A5%D0%BE%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%B4%D0%B0&action=edit&redlink=1).

**Метод Ринга** Этот метод используется, когда ожидается, что возмещение основной суммы будет осуществляться равными частями. Норма возврата представляет собой ежегодную долю первоначального капитала, отчисляемую в беспроцентный фонд возмещения. Эта доля при 100-процентном возврате капитала.

**Метод Инвуда** Данный метод используется, если сумма возврата капитала реинвестируется по ставке доходности инвестиций. В этом случае норма возврата как составная часть коэффициента капитализации равна фактору фонда возмещения при той же ставке процента.

**Метод Хоскольда** Используется в тех случаях, когда ставка дохода первоначальных инвестиций несколько высока, и маловероятно реинвестирование по той же ставке. Для реинвестируемых средств предполагается получение дохода по безрисковой ставке.

Данный метод используется, если потоки доходов стабильны на протяжении длительного периода и являются значительной величиной, а также, если поток доходов возрастает устойчивыми темпами.

Метод неприменим в ситуации, когда объект оценки представляет собой объект незавершённого строительства или требует значительной реконструкции, т.е в ближайший период не может приносить стабильного дохода.

«Информационная непрозрачность» российского рынка затрудняет расчёт Чистого операционного дохода и ставки капитализации, ввиду недостатка информации по реальным сделкам продажи, эксплуатационным платежам и т. д.

#### 2.3.2 Метод дисконтирования денежных потоков

Метод дисконтированных денежных потоков (ДДП) более сложен, детален и позволяет оценить объект в случае получения от него нестабильных денежных потоков, моделируя характерные черты их поступления.

Применяется метод ДДП, когда:

* предполагается, что будущие денежные потоки будут существенно отличаться от текущих;
* имеются данные, позволяющие обосновать размер будущих потоков денежных средств от недвижимости;
* потоки доходов и расходов носят сезонный характер;
* оцениваемая недвижимость — крупный многофункциональный коммерческий объект;
* объект недвижимости строится или только что построен и ввод: (или введен в действие).

Метод дисконтированных денежных потоков — наиболее универсальный метод, позволяющий определить настоящую стоимость будущих денежных потоков. Денежные потоки могут произвольно изменяться, неравномерно поступать и отличаться высоким уровнем риска. Это связано со спецификой такого понятия, как недвижимое имущество. Недвижимое имущество приобретается инвестором в основном из-за определенных выгод в будущем. Инвестор рассматривает объект недвижимости в виде набора будущих преимуществ и оценивает его привлекательность с позиций того, как денежное выражение этих будущих преимуществ соотносится с ценой, по которой объект может быть приобретен.

Метод ДДП позволяет оценить стоимость недвижимости на основе текущей стоимости дохода, состоящего из прогнозируемых денежных потоков и остаточной стоимости.

# Постановка задачи дипломного проекта и выбор языка программирования

### 3.1 Постановка задачи дипломного проекта

Для оценки недвижимости используют несколько подходов:

1. *затратный* – состоит в определении затрат на покупку такого же земельного участка и постройку аналогичного здания, при этом учитывается износ недвижимости,
2. *сравнительный* – основан на сравнении оцениваемого объекта с подобными объектами недвижимости, уже выставленными на продажу,
3. *доходный* – рассматривается возможность окупаемости вложенных в недвижимость денежных средств за определённый период.

На основе рассмотренной ранее предметной области мы можем выделить задачи, которые должен выполнять программный комплекс:

Основные:

* Реализация поиска аналогов для сравнительного подхода
* Реализация автоматического составления таблиц для доходного и затратного подходов.

Второстепенные:

* Предоставление информации необходимой в процессе оценки
* Введение дополнительных функций программы для удобства ее использования

## 3.2 Выбор языка программирования

***3.2.1 Пользовательский интерфейс***

Для написания пользовательских интерфейсов были рассмотрены несколько языков программирования ориентированными на написание приложений и возможностями связи с БД:

* Java - [объектно-ориентированный язык программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный компанией [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем, приобретённой компанией Oracle). Приложения Java обычно [компилируются](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой [виртуальной Java-машине](http://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) независимо от [компьютерной архитектуры](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0). Дата официального выпуска — [23 мая](http://ru.wikipedia.org/wiki/23_%D0%BC%D0%B0%D1%8F) [1995 года](http://ru.wikipedia.org/wiki/1995_%D0%B3%D0%BE%D0%B4).
* C++ - [компилируемый](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) [статически типизированный](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения.Поддерживает разные [парадигмы программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), сочетает свойства как [высокоуровневых](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), так и [низкоуровневых](http://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) языков. В сравнении с его предшественником — языком [C](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29), — наибольшее внимание уделено поддержке [объектно-ориентированного](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [обобщённого программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Название «C++» происходит от названия языка C, в котором унарный оператор ++ обозначает [инкремент](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D1%82) переменной.
* Python ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) python — [питон](http://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%82%D0%BE%D0%BD), произносится [[ˈpaɪθ⟨ə⟩n]](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) — па́йтон, также, широко используется русскоязычное произношение пито́н) — [высокоуровневый язык программирования](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения с акцентом на производительность разработчика и читаемость кода. [Синтаксис](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) ядра Python минималистичен. В то же время [стандартная библиотека](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой объём полезных функций.

Java был разработан максимально похожим на C++. Синтаксис языка Java, по существу, представляет собой упрощенный вариант синтаксиса С++.

Python использует динамическую типизацию, что может усложнить разработку программы.

Также следует отметить, что одна из целей языка Java – обеспечить разработку независимых программ, способных выполнятся на машинах с ограниченным объемом ресурсов.

ВЫВОД: Язык программирования Java наиболее полно подходит для реализации поставленных задач

#### 3.2.2 Интернет ресурс

Для реализации интернет ресурса нам не понадобится рассматривать несколько языков для их написания, из-за небольшой функциональной составляющей для его реализации будет использоваться html и css

* HTML (от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HyperText Markup Language — «язык разметки [гипертекста](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82)») — стандартный [язык разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](http://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) создаются при помощи языка HTML (или [XHTML](http://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) и отображается в виде документа, в удобной для человека форме.
* CSS ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Cascading Style Sheets — каскадные таблицы стилей) — [формальный язык](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) описания внешнего вида документа, написанного с использованием [языка разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8).

# Особенности создания программ на объектно-ориентированном языке программирования Java

**Java** — [объектно-ориентированный язык программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный компанией [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем, приобретённой компанией Oracle). Приложения Java обычно [компилируются](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой [виртуальной Java-машине](http://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) независимо от [компьютерной архитектуры](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0). Дата официального выпуска — [23 мая](http://ru.wikipedia.org/wiki/23_%D0%BC%D0%B0%D1%8F) [1995 года](http://ru.wikipedia.org/wiki/1995_%D0%B3%D0%BE%D0%B4).

Произношение, применение названия

Правильная форма произношения — заимствованная англоязычная [/ˈdʒɑːvə/](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) («джава» с ударением на первом слоге). Традиционно-национальная «ява», соответствующая традиционному произношению названия острова [Ява](http://ru.wikipedia.org/wiki/%D0%AF%D0%B2%D0%B0) не признана правообладателями торговой марки Java правильной. Компания Sun (ныне Oracle) придерживается англоязычного произношения во всех странах мира[. Иногда в обиходе используют также жаргонное слово «Жаба» (например, изображение жабы есть на календариках группы российских пользователей Java (Java Users Group).

Java — так называют не только сам язык, но и [платформу](http://ru.wikipedia.org/wiki/Java_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0%29) для создания и исполнения приложений на основе данного языка.

Изначально язык назывался Oak («дуб») и разрабатывался [Джеймсом Гослингом](http://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%81%D0%BB%D0%B8%D0%BD%D0%B3,_%D0%94%D0%B6%D0%B5%D0%B9%D0%BC%D1%81) для программирования бытовых электронных устройств. Впоследствии он был переименован в Java и стал использоваться для написания клиентских приложений и [серверного](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%29) программного обеспечения. Назван в честь марки [кофе](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%84%D0%B5) Java, любимого некоторыми [программистами](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82), поэтому на официальной эмблеме языка изображена чашка с дымящимся кофе. Существует и другая версия происхождения названия Java, а именно, Java это [сленговое](http://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B5%D0%BD%D0%B3) обозначение кофе (по имени одноименного острова, где производится популярный сорт кофе) с [аллюзией](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%BB%D1%8E%D0%B7%D0%B8%D1%8F) на кофе-машину, как пример бытового устройства, для программирования которого изначально язык создавался.

## 4.1 Основные особенности языка

Программы на Java [транслируются](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), выполняемый [виртуальной машиной Java](http://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).

Достоинство подобного способа выполнения программ — в полной независимости байт-кода от [операционной системы](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [оборудования](http://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. В последнее время был внесен ряд усовершенствований, которые несколько увеличили скорость выполнения программ на Java:

* применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы ([JIT](http://ru.wikipedia.org/wiki/JIT)-технология) с возможностью сохранения версий класса в машинном коде,
* широкое использование [платформенно-ориентированного кода](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (native-код) в стандартных библиотеках,
* аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология [Jazelle](http://ru.wikipedia.org/wiki/Jazelle), поддерживаемая некоторыми процессорами фирмы [ARM](http://ru.wikipedia.org/wiki/ARM_%28%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F%29)).

По данным сайта shootout.alioth.debian.org, для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее. С другой стороны, для большинства из них потребление памяти Java-машиной было в 10-30 раз больше, чем программой на C/C++. Также примечательно исследование, проведённое компанией [Google](http://ru.wikipedia.org/wiki/Google_%28%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F%29), согласно которому отмечается существенно более низкая производительность и бо́льшее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на [C++](http://ru.wikipedia.org/wiki/C%2B%2B).

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине. Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры [CLI](http://ru.wikipedia.org/wiki/Common_Language_Infrastructure), заложенной в основу платформы [.NET](http://ru.wikipedia.org/wiki/.NET_Framework) компанией [Microsoft](http://ru.wikipedia.org/wiki/Microsoft).

#### Java 1.0

Разработка Java началась в 1990 году, первая официальная версия — Java 1.0, — была выпущена только в 1996 году.

#### Java 1.2

К [1998 году](http://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4) была разработана обновлённая спецификация JDK 1.2, вышедшая под наименованием Java 2. Язык практически не изменился — было добавлено одно [ключевое слово](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%BE) strictfp. Платформа получила следующие дополнения:

* [Библиотека](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) [Swing](http://ru.wikipedia.org/wiki/Swing) для создания [графического интерфейса пользователя](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) .
* Коллекции (JSR 166).
* Поддержка файлов Policy и цифровых сертификатов пользователя.
* Библиотека Accessibility.
* Java 2D.
* Поддержка технологии [drag-and-drop](http://ru.wikipedia.org/wiki/Drag-and-drop).
* Полная поддержка [Unicode](http://ru.wikipedia.org/wiki/Unicode), включая поддержку ввода на японском, китайском и корейском языках.
* Поддержка воспроизведения аудиофайлов нескольких популярных форматов.
* Полная поддержка технологии [CORBA](http://ru.wikipedia.org/wiki/CORBA).
* JIT-компилятор, улучшенная производительность.
* Усовершенствования инструментальных средств JDK, в том числе поддержка профилирования Java-программ

#### Java 5.0

Спецификация Java 5.0 была выпущена в сентябре [2004](http://ru.wikipedia.org/wiki/2004) года. C этой версии изменена официальная индексация, вместо Java 1.5 правильнее называть Java 5.0. Внутренняя же индексация Sun осталась прежней - 1.x. Минорные изменения теперь включаются без изменения индексации, для этого используется слово "Update" или буква "u", например Java Development Kit 5.0 Update 22. Предполагается, что в обновления могут входить как исправления ошибок так и небольшие добавления в API, JVM.

В данной версии разработчики внесли в язык целый ряд принципиальных дополнений:

* [Перечислимые типы](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D0%BC%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) enum). Ранее отсутствовавшие в Java типы оформлены по аналогии с [C++](http://ru.wikipedia.org/wiki/C%2B%2B), но при этом имеют ряд дополнительных возможностей.
  + Перечислимый тип является полноценным классом Java, то есть может иметь конструктор, поля, методы, в том числе скрытые и абстрактные.
  + Перечисление может реализовывать [интерфейсы](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%28%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29).
  + Для перечислений имеются встроенные методы, дающие возможность получения значений типа по имени, символьных значений, соответствующих именам, преобразования между номером и значением, проверки типа на то, что он является перечислимым.
* [Аннотации](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F) — возможность добавления в текст программы метаданных, не влияющих на выполнение кода, но допускающих использование для получения различных сведений о коде и его исполнении. Одновременно выпущен инструментарий для использования аннотированного кода. Одно из применений аннотаций — упрощение создания тестовых модулей для Java-кода.
* Средства [обобщённого программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) generics) — механизм, аналогичный [Eiffel](http://ru.wikipedia.org/wiki/Eiffel) (позже также появились и в C#, принципиально отличаются от шаблонов C++), дающий возможность создавать классы и методы с полями и параметрами произвольного объектного типа. С использованием данного механизма реализованы новые версии коллекций стандартной библиотеки Java.
* Методы с неопределённым числом параметров.
* Autoboxing/Unboxing — автоматическое преобразование между скалярными типами Java и соответствующими типами-врапперами (например, между int — Integer). Наличие такой возможности сокращает код, поскольку исключает необходимость выполнения явных преобразований типов в очевидных случаях.
* Разрешён импорт статических полей и методов.
* В язык введён цикл по коллекции объектов (итератор, [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) foreach).
* Было введено использование [Javadoc](http://ru.wikipedia.org/wiki/Javadoc)-комментариев, которые используются для автоматического оформления документации по комментариям в исходном коде.

#### Java 6

Релиз версии состоялся 11 декабря 2006 года. Изменена официальная индексация - вместо ожидаемой 6.0 версия значится как 6. Минорные изменения как и в Java 5.0 вносятся в обычные обновления версии, например Java SE Development Kit 6 Update 27.

#### Java 7

Релиз версии состоялся 28 июля 2011 года. В финальную версию Java SE 7 не были включены все ранее запланированные изменения. Согласно плану развития (план «Б»), включение нововведений будет разбито на две части: Java SE 7 (без [лямбд](http://ru.wikipedia.org/wiki/%D0%9B%D1%8F%D0%BC%D0%B1%D0%B4%D0%B0-%D0%B8%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), проекта Jigsaw, и части улучшений [Coin](http://en.wikipedia.org/wiki/COIN-OR)) и Java SE 8 (все остальное), намеченный на конец 2012 года.

В новой версии, получившей название Java SE 7 (Java Platform, Standard Edition 7), помимо исправления большого количества ошибок были представлено несколько новшеств. Так, например, в качестве эталонной реализации Java SE 7 использован не проприетарный пакет [JDK](http://ru.wikipedia.org/wiki/JDK), а его открытая реализация [OpenJDK](http://ru.wikipedia.org/wiki/OpenJDK), а сам релиз новой версии платформы готовился при тесном сотрудничестве инженеров [Oracle](http://ru.wikipedia.org/wiki/Oracle) с участниками мировой экосистемы Java, комитетом [JCP](http://ru.wikipedia.org/wiki/JCP) (Java Community Process) и сообществом OpenJDK. Все поставляемые Oracle бинарные файлы эталонной реализации Java SE 7 собраны на основе кодовой базы OpenJDK, сама эталонная реализация полностью открыта под лицензией [GPLv2](http://ru.wikipedia.org/wiki/GPLv2) с исключениями [GNU](http://ru.wikipedia.org/wiki/GNU) ClassPath, разрешающими динамическое связывание с проприетарными продуктами. К другим нововведениям относится интеграция набора небольших языковых улучшений Java, развиваемых в рамках проекта Coin, добавлена поддержка языков программирования с динамической типизацией, таких как [Ruby](http://ru.wikipedia.org/wiki/Ruby), [Python](http://ru.wikipedia.org/wiki/Python) и [JavaScript](http://ru.wikipedia.org/wiki/JavaScript), поддержка загрузки классов по [URL](http://ru.wikipedia.org/wiki/URL), обновленный [XML](http://ru.wikipedia.org/wiki/XML)-стек, включающий [JAXP](http://ru.wikipedia.org/wiki/JAXP) 1.4, [JAXB](http://ru.wikipedia.org/w/index.php?title=JAXB&action=edit&redlink=1) 2.2a и [JAX-WS](http://ru.wikipedia.org/wiki/JAX-WS) 2.2 и другие.

За 5 дней до выхода релиза Java SE 7 было обнаружено несколько серьёзных ошибок в горячей оптимизации циклов, которая включена по умолчанию и приводит виртуальную машину Java к краху. Специалисты Oracle найденные ошибки за столь короткий срок исправить не могли, но пообещали, что они будут исправлены во втором обновлении (Java 7 Update 2) и частично в первом.

Список нововведений:

* Поддержка динамически-типизированных языков (InvokeDynamic) — расширение JVM (семантики байт-кода), языка Java для поддержки динамически-типизированных языков.
* Строгая проверка class-файлов — class-файлы версии 51 (SE 7) или более поздней версии должны быть проверены typechecking верификатором; JVM не должна переключаться на старый верификатор.
* Изменение синтаксиса языка Java (Project Coin) — частичные изменения в языке Java, предназначенные для упрощения общих задач программирования:
  + Использование класса String в блоке switch.
  + Закрытие используемых ресурсов в блоке try (try-with-resources) — работает при использовании интерфейса AutoCloseable.
  + Объединённая обработка исключений в блоке catch (multi-catch exceptions) — перечисление обрабатываемых исключений в catch (… | … | …).
  + Повторное выбрасывание исключений (rethrowing exceptions) — передача возникшего исключения «вверх» по [стеку вызовов](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D0%BE%D0%B2).
  + Подчёркивания в числовых литералах для лучшего восприятия больших чисел.
  + Изменение вывода типа в Java generic при создании объекта.
  + Использование двоичных чисел (binary literals) — [префикс](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81) «0b» укажет, что используется двоичное число.
  + Упрощение вызова методов varargs — уменьшение предупреждений при вызове метода с переменным числом входящих переменных.
* Модификация загрузчика классов (class-loader) — избежание тупиковых ситуаций в неиерархической топологии загрузки классов.
* Закрытие ресурсов, открытых URLClassLoader.
* Обновление коллекций (JSR 166y).
* Поддержка Unicode 6.0.
* Отделение языка пользователя и языка пользовательского интерфейса — обновление обработки языков для отделения локали от языка пользовательского интерфейса.
* Новые интерфейсы I/O для платформы Java (nio.2).
* Использование JDBC 4.1 и Rowset 1.1.
* (не закончено) …

## 4.2 Классификация платформ Java

Внутри Java существуют несколько основных семейств технологий:

* [Java SE](http://ru.wikipedia.org/wiki/Java_SE) — Java Standard Edition, основное издание Java, содержит компиляторы, API, Java Runtime Environment; подходит для создания пользовательских приложений, в первую очередь — для настольных систем.
* [Java EE](http://ru.wikipedia.org/wiki/Java_EE) — Java Enterprise Edition, представляет собой набор спецификаций для создания программного обеспечения уровня предприятия.
* [Java ME](http://ru.wikipedia.org/wiki/Java_ME) — Java Micro Edition, создана для использования в устройствах, ограниченных по вычислительной мощности, например в [мобильных телефонах](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [КПК](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), встроенных системах;
* [JavaFX](http://ru.wikipedia.org/wiki/JavaFX) — технология, являющаяся следующим шагом в эволюции Java как Rich Client Platform; предназначена для создания графических интерфейсов корпоративных приложений и бизнеса.
* [Java Card](http://ru.wikipedia.org/wiki/Java_Card) — технология предоставляет безопасную среду для приложений, работающих на смарт-картах и ​​других устройствах с очень ограниченным объемом памяти и возможностями обработки.

## 4.3 Java и Microsoft

Компанией [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) была разработана собственная реализация [JVM](http://ru.wikipedia.org/wiki/JVM) (MSJVM), включавшаяся в состав различных [операционных систем](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), начиная с [Windows 98](http://ru.wikipedia.org/wiki/Windows_98) (также входила в Internet Explorer от версии 3 и выше, что позволяло использовать MSJVM в ОС Windows 95 и Windows NT 4 после установки IE3+ на данные ОС).

MSJVM имела cущественные отличия от Sun Java, во многом ломающие основополагающую концепцию переносимости программ между разными платформами:

* отсутствие поддержки [программного интерфейса](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) [вызова удаленных методов](http://ru.wikipedia.org/wiki/Remote_Procedure_Call) ([RMI](http://ru.wikipedia.org/wiki/RMI));
* отсутствие поддержки технологии [JNI](http://ru.wikipedia.org/wiki/JNI);
* наличие нестандартных расширений, таких как средства интеграции Java и [DCOM](http://ru.wikipedia.org/wiki/DCOM), работающих только на платформе Windows.

Тесная интеграция Java с DCOM и Win32 ломала многоплатформенную суть языка. Впоследствии это явилось поводом для судебных исков со стороны Sun Microsystems к Microsoft. Суд принял сторону компании Sun Microsystems. В конечном счёте между двумя компаниями была достигнута договорённость о возможности продления срока официальной поддержки пользователей нестандартной Microsoft JVM до конца 2007 года, но не более.

В 2005 году компанией Microsoft для платформы [.NET](http://ru.wikipedia.org/wiki/.NET_Framework) был представлен Java-подобный язык [J#](http://ru.wikipedia.org/wiki/Visual_J_Sharp), не соответствующий официальной спецификации языка Java и исключённый впоследствии из стандартного инструментария разработчика [Microsoft Visual Studio](http://ru.wikipedia.org/wiki/Microsoft_Visual_Studio), начиная с Visual Studio 2008.

## 4.4 Применения платформы Java

Следующие успешные проекты реализованы с привлечением Java ([J2EE](http://ru.wikipedia.org/wiki/J2EE)) технологий: [RuneScape](http://ru.wikipedia.org/wiki/RuneScape), [Amazon](http://ru.wikipedia.org/wiki/Amazon), [eBay](http://ru.wikipedia.org/wiki/EBay), [Yandex](http://ru.wikipedia.org/wiki/Yandex) (неоднозначная информация в отношении Java), [LinkedIn](http://ru.wikipedia.org/wiki/LinkedIn), [Yahoo!](http://ru.wikipedia.org/wiki/Yahoo%21).

Следующие компании в основном фокусируются на Java ([J2EE](http://ru.wikipedia.org/wiki/J2EE)) технологиях: [SAP](http://ru.wikipedia.org/wiki/SAP), [IBM](http://ru.wikipedia.org/wiki/IBM), [Oracle](http://ru.wikipedia.org/wiki/Oracle_Corporation). В частности, [СУБД](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) Oracle включает JVM как свою составную часть, обеспечивающую возможность непосредственного программирования СУБД на языке Java, включая, например, [хранимые процедуры](http://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0).

## 4.5 Производительность

Программы, написанные на Java, имеют репутацию более медленных и занимающих больше оперативной памяти, чем написанные на языке Си. Тем не менее, скорость выполнения программ, написанных на языке Java, была существенно улучшена с выпуском в 1997—1998 годах так называемого JIT-компилятора в версии 1.1 в дополнение к другим особенностям языка для поддержки лучшего анализа кода (такие как внутренние классы, класс StringBuffer, упрощенные логические вычисления и т. д.). Кроме того была произведена оптимизация виртуальной машины Java — с 2000 года для этого используется виртуальная машина [HotSpot](http://ru.wikipedia.org/wiki/HotSpot). В настоящее время, код Java 2.0 приблизительно лишь в 2 раза медленнее кода, написанного на языке Cи

Некоторые платформы предлагают аппаратную поддержку выполнения для Java. К примеру, микроконтроллеры выполняющие код Java на аппаратном обеспечении вместо программной JVM, а также основанные на ARM процессоры, которые поддерживают выполнение байткода Java через опцию Jazelle.

## 4.6 Основные возможности

* [автоматическое управление памятью](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0);
* расширенные возможности обработки исключительных ситуаций;
* богатый набор средств фильтрации ввода/вывода;
* набор стандартных коллекций, таких как [массив](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), [список](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29), [стек](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) и т. п.;
* наличие простых средств создания сетевых приложений (в том числе с использованием [протокола](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [RMI](http://ru.wikipedia.org/wiki/RMI));
* наличие классов, позволяющих выполнять [HTTP](http://ru.wikipedia.org/wiki/HTTP)-запросы и обрабатывать ответы;
* встроенные в язык средства создания многопоточных приложений;
* унифицированный доступ к [базам данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85):
* на уровне отдельных [SQL](http://ru.wikipedia.org/wiki/SQL)-запросов — на основе [JDBC](http://ru.wikipedia.org/wiki/JDBC), [SQLJ](http://ru.wikipedia.org/wiki/SQLJ);
* на уровне концепции объектов, обладающих способностью к хранению в базе данных — на основе [Java Data Objects](http://ru.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1) ([англ.](http://en.wikipedia.org/wiki/Java_Data_Objects)) и [Java Persistence API](http://ru.wikipedia.org/wiki/Java_Persistence_API);
* поддержка шаблонов (начиная с версии 1.5);
* параллельное выполнение программ.

## 4.7 Пример программы

Программа, выводящая «[Hello, world!](http://ru.wikipedia.org/wiki/Hello,_world%21)»:

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, world!");

}

}

Пример использования [generics](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5):

import java.util.\*;

public class Sample {

public static void main(String[] args) {

// Создание объекта по шаблону.

List<String> strings = new LinkedList<String>();

strings.add("Hello");

strings.add("world");

strings.add("!");

for (String s : strings) {

System.out.print(s);

System.out.print(" ");

}

}

}

Пример использования [reflection](http://ru.wikipedia.org/wiki/Reflection):

import java.lang.reflect.Field;

import java.lang.reflect.Method;

class TestClass {

private int value;

public int getValue() { return value; }

public void setValue(int value) { this.value = value; }

}

public class Main {

public static void main(String[] args) {

TestClass testClass = new TestClass();

for (Field field: testClass.getClass().getDeclaredFields()) {

System.out.printf("name: %s, type: %s \n", field.getName(), field.getType().getCanonicalName());

}

for (Method method : testClass.getClass().getDeclaredMethods()) {

System.out.printf("name: %s, return type: %s \n", method.getName(), method.getReturnType().getCanonicalName());

}

}

}

Пример использования [аннотаций](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BD%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29&action=edit&redlink=1):

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.TYPE)

public @interface Annotation {

public boolean value() default false;

}

@Annotation(value=true)

public class TestClass {

}

public class Main {

public static void main(String[] args) {

TestClass testClass = new TestClass();

Annotation annotation = testClass.getClass().getAnnotation(Annotation.class);

if (annotation != null) {

System.out.printf("value: %s \n", annotation.value());

}

}

}

## 4.8 Примитивные типы

В языке Java только 8 примитивных (скалярных, простых) [типов](http://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85): [boolean](http://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B8%D0%BF), [byte](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82), [char](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), [short](http://ru.wikipedia.org/w/index.php?title=Short&action=edit&redlink=1), [int](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), [long](http://ru.wikipedia.org/w/index.php?title=Long&action=edit&redlink=1), [float](http://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE_%D0%BE%D0%B4%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%BE%D0%B9_%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8), [double](http://ru.wikipedia.org/wiki/Double). Существует также вспомогательный девятый примитивный тип — [void](http://ru.wikipedia.org/wiki/Void), однако переменные и поля такого типа не могут быть объявлены в коде, а сам тип используется только для описания соответствующего ему класса, для использовании при [рефлексии](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29).

Длины и диапазоны значений примитивных типов определяются стандартом, а не реализацией и приведены в таблице. Тип char сделали двухбайтовым для удобства локализации (один из идеологических принципов Java): когда складывался стандарт, уже существовал [Unicode](http://ru.wikipedia.org/wiki/Unicode)-16, но не Unicode-32. Поскольку в результате не осталось однобайтового типа, добавили новый тип byte, причем в Java, в отличие от других языков, он не является беззнаковым. Типы float и double могут иметь специальные значения +\infty, -\inftyи «не число» ([NaN](http://ru.wikipedia.org/wiki/NaN)). Для типа double они обозначаются Double.POSITIVE\_INFINITY, Double.NEGATIVE\_INFINITY, Double.NaN; для типа float — так же, но с приставкой Float вместо Double. Минимальные положительные значения, принимаемые типами float и double, тоже стандартизованы.

|  |  |  |
| --- | --- | --- |
| **Тип** | **Длина (в байтах)** | **Диапазон или набор значений** |
| boolean | 1 в массивах, 4 в переменных | true, false |
| byte | 1 | −128..127 |
| char | 2 | 0..216−1, или 0..65535 |
| short | 2 | −215..215−1, или −32768..32767 |
| int | 4 | −231..231−1, или −2147483648..2147483647 |
| long | 8 | −263..263−1, или примерно −9.2·1018..9.2·1018 |
| float | 4 | -(2-2−23)·2127..(2-2−23)·2127, или примерно −3.4·1038..3.4·1038, а также -\infty, \infty, NaN |
| double | 8 | -(2-2−52)·21023..(2-2−52)·21023, или примерно −1.8·10308..1.8·10308, а также -\infty, \infty, NaN |

Такая жёсткая стандартизация была необходима, чтобы сделать язык платформенно-независимым, что является одним из идеологических требований к Java. Тем не менее одна небольшая проблема с платформенной независимостью всё же осталась. Некоторые процессоры используют для промежуточного хранения результатов 10-байтовые [регистры](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0) или другими способами улучшают точность вычислений. Для того, чтобы сделать Java максимально совместимой между разными системами, в ранних версиях любые способы повышения точности вычислений были запрещены. Однако это приводило к снижению быстродействия. Выяснилось, что ухудшение точности ради платформенной независимости мало кому нужно, тем более если за это приходится платить замедлением работы программ. После многочисленных протестов этот запрет отменили, но добавили ключевое слово strictfp, запрещающее повышение точности.

#### Преобразования при математических операциях

В языке Java действуют следующие правила:

1. Если один [операнд](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4) имеет тип double, другой тоже преобразуется к типу double.
2. Иначе, если один операнд имеет тип float, другой тоже преобразуется к типу float.
3. Иначе, если один операнд имеет тип long, другой тоже преобразуется к типу long.
4. Иначе оба операнда преобразуются к типу int.

Данный способ неявного преобразования встроенных типов полностью совпадает с преобразованием типов в C++.

## 4.9 Объектные переменные, объекты, ссылки и указатели

В языке Java имеются только динамически создаваемые объекты. Причем [переменные](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) объектного типа и объекты в Java — совершенно разные сущности. Переменные объектного типа являются [ссылками](http://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29), то есть неявными [указателями](http://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_%28%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%29) на динамически создаваемые объекты. Это подчёркивается [синтаксисом](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) описания переменных. Так, в Java нельзя писать:

double a[10][20];

Foo b(30);

а нужно:

double[][] a = new double[10][20];

Foo b = new Foo(30);

При присваиваниях, передаче в [подпрограммы](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) и сравнениях объектные переменные ведут себя как указатели, то есть присваиваются, копируются и сравниваются [адреса](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D1%80%D0%B5%D1%81_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) объектов. А при доступе с помощью объектной переменной к полям данных или методам объекта не требуется никаких специальных операций [разыменовывания](http://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B0%D0%B7%D1%8B%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) — этот доступ осуществляется так, как если бы объектная переменная была самим объектом.

Объектными являются переменные любого типа, кроме примитивного. Явных указателей в Java нет. В отличие от указателей C, C++ и других языков программирования, ссылки в Java в высокой степени безопасны благодаря жёстким ограничениям на их использование, в частности:

* Нельзя преобразовывать объект типа int или любого другого примитивного типа в указатель или ссылку и наоборот.
* Над ссылками запрещено выполнять операции ++, −−, +, − или любые другие арифметические операции.
* Преобразование типов между ссылками жёстко регламентировано. За исключением ссылок на массивы, разрешено преобразовывать ссылки только между наследуемым типом и его наследником, причём преобразование наследуемого типа в наследующий должно быть явно задано и во время выполнения производится проверка его осмысленности. Преобразования ссылок на массивы разрешены лишь тогда, когда разрешены преобразования их базовых типов, а также нет конфликтов размерности.
* В Java нет операций взятия адреса (&) или взятия объекта по адресу (\*). Звёздочка в Java означает умножение, и только. [Амперсанд](http://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%BF%D0%B5%D1%80%D1%81%D0%B0%D0%BD%D0%B4#.D0.9F.D1.80.D0.BE.D1.87.D0.B8.D0.B5_.D0.BF.D1.80.D0.B8.D0.BC.D0.B5.D0.BD.D0.B5.D0.BD.D0.B8.D1.8F) (&) означает всего лишь «побитовое и» (двойной амперсанд — «логическое и»).

Благодаря таким специально введенным ограничениям в Java невозможно прямое манипулирование памятью на уровне физических адресов (хотя ссылки, не указывающие ни на что, есть: значение такой ссылки обозначается null).

Если нужен указатель на примитивный тип, используются классы-обёртки примитивных типов: [Boolean](http://java.sun.com/javase/6/docs/api/java/lang/Boolean.html), [Byte](http://java.sun.com/javase/6/docs/api/java/lang/Byte.html), [Character](http://java.sun.com/javase/6/docs/api/java/lang/Character.html), [Short](http://java.sun.com/javase/6/docs/api/java/lang/Short.html), [Integer](http://java.sun.com/javase/6/docs/api/java/lang/Integer.html), [Long](http://java.sun.com/javase/6/docs/api/java/lang/Long.html), [Float](http://java.sun.com/javase/6/docs/api/java/lang/Float.html), [Double](http://java.sun.com/javase/6/docs/api/java/lang/Double.html).

#### Дублирование ссылок и клонирование

Из-за того, что объектные переменные являются ссылочными, при присваивании не происходит копирования объекта. Так, если написать

Foo foo, bar;

…

bar = foo;

то произойдет копирование адреса из переменной foo в переменную bar. То есть foo и bar будут указывать на одну и ту же область памяти, то есть на один и тот же объект; попытка изменить поля объекта, на который ссылается переменная foo, будет менять объект, с которым связана переменная bar, и наоборот. Если же необходимо получить именно ещё одну *копию* исходного объекта, пользуются или [методом](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%28%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29) (функцией-членом, в терминологии C++) clone(), создающим копию объекта, или (реже) копирующим [конструктором](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) (конструкторы в Java не могут быть виртуальными, поэтому экземпляр класса-потомка будет неправильно скопирован конструктором класса-предка; метод клонирования вызывает нужный конструктор и тем самым позволяет обойти это ограничение).

Метод clone() требует, чтобы класс реализовывал [интерфейс](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%28%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) Cloneable (об интерфейсах см. ниже). Если класс реализует интерфейс Cloneable, по умолчанию clone() копирует все поля (мелкая копия). Если требуется не копировать, а клонировать поля (а также их поля и так далее), надо переопределять метод clone(). Определение и использование метода clone() часто является нетривиальной задачей.

#### Сборка мусора

В языке Java невозможно явное удаление объекта из памяти — вместо этого реализована [сборка мусора](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0). Традиционным приёмом, дающим сборщику мусора «намёк» на освобождение памяти, является присваивание переменной пустого значения null. Это, однако, не значит, что объект, заменённый значением null, будет непременно и немедленно удалён, но есть гарантия, что этот объект будет удалён именно в будущем. Данный приём всего лишь устраняет ссылку на объект, то есть отвязывает указатель от объекта в памяти. При этом следует учитывать, что объект не будет удален сборщиком мусора, пока на него указывает хотя бы одна ссылка из используемых переменных или объектов. Существуют также методы для инициации принудительной сборки мусора, но не гарантируется, что они будут вызваны исполняющей средой, и их не рекомендуется использовать для обычной работы.

### 4.10 Классы и функции

Java не является процедурным языком: любая функция может существовать только внутри класса. Это подчёркивает терминология языка Java, где нет понятий «функция» или «функция-член» ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) member function), а только метод. В методы превратились и стандартные функции. Например, в Java нет функции sin(), а есть метод Math.sin() класса Math (содержащего, кроме sin(), методы cos(), exp(), sqrt(), abs() и многие другие). Конструкторы в Java не считаются методами. Деструкторов в Java не существует, а метод finalize() ни в коем случае нельзя считать аналогом деструктора.

#### Статические методы и поля

В Java (как и в C++) используются *статические метод*ы ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *static method* — в теории программирования их также называют методами класса), которые задаются при помощи ключевого слова static. Статические поля (переменные класса) имеют тот же смысл, что и в C++: каждое такое поле является собственностью класса, поэтому для доступа к статическим полям не требуется создавать экземпляры соответствующего класса.

Например, математические функции, реализованные в классе Math, представляют собой как раз статические методы данного класса. Поэтому можно писать

double x = Math.sin(1);

вместо

Math m = new Math();

double x = m.sin(1);

Поскольку статические методы существуют независимо от объектов (экземпляров класса), они не имеют доступа к обычным (нестатическим) полям и методам данного класса. В частности, при реализации статического метода недопустимо использовать идентификатор this.

#### Завершённость (final)

Ключевое слово final (финальный) означает разные вещи при описании переменной, метода или класса. Финальное поле класса инициализируется при описании, или в конструкторе класса (а статичное поле — в статичном блоке инициализации) и дальше его значение не может быть изменено. Значения локальных переменных, а также параметров метода, помеченных ключевым словом final, не могут быть изменены после присвоения. Но при этом, их значения могут использоваться внутри [анонимных классов](http://ru.wikipedia.org/wiki/%D0%92%D0%BD%D1%83%D1%82%D1%80%D0%B5%D0%BD%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81#.D0.90.D0.BD.D0.BE.D0.BD.D0.B8.D0.BC.D0.BD.D1.8B.D0.B5_.28.D0.B1.D0.B5.D0.B7.D1.8B.D0.BC.D1.8F.D0.BD.D0.BD.D1.8B.D0.B5.29_.D0.BA.D0.BB.D0.B0.D1.81.D1.81.D1.8B). Если статичное поле класса, или переменная проинициализированны константным выражением, они рассматриваются компилятором, как [именованная константа](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29); в таком случае их значение может быть использовано в [операторах switch](http://ru.wikipedia.org/wiki/Switch_%28%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%29) (для констант типа int), а также для условной компиляции (для констант типа boolean) при использовании с [оператором if](http://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80).

Метод класса, отмеченный словом final, не может быть переопределён при наследовании. [Финальный класс](http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81) не может иметь наследников вообще.

#### Абстрактность

В Java методы, не объявленные явно как static, final или private, являются [виртуальными](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4) в терминологии C++: при вызове метода, по-разному определённого в базовом и наследующем классах, всегда производится проверка времени выполнения.

[Абстрактным методом](http://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4) (модификатор abstract) в Java называется метод, для которого заданы параметры и тип возвращаемого значения, но не тело. Абстрактный метод определяется в классах-наследниках. В C++ то же самое называется чисто виртуальной функцией. Для того чтобы в классе можно было описывать абстрактные методы, сам класс тоже должен быть описан как абстрактный. Объекты абстрактного класса создавать нельзя.

#### Интерфейсы

Высшей степенью абстрактности в Java является интерфейс (interface). Все методы интерфейса абстрактны: описатель abstract даже не требуется. Интерфейс в Java не считается классом, хотя по сути является полностью абстрактным классом. Класс может наследовать/*расширять* (extends) другой класс или *реализовывать* (implements) интерфейс. Кроме того, интерфейс может наследовать/расширять другой интерфейс.

В Java класс не может наследовать более одного класса, зато может реализовывать сколько угодно интерфейсов. Множественное наследование интерфейсов не запрещено, то есть один интерфейс может наследоваться от нескольких.

Интерфейсы можно передавать методам как параметры, но нельзя создавать объекты их типов.

#### Маркерные интерфейсы

В Java есть некоторые интерфейсы, которые не содержат методов для реализации, а специальным образом обрабатываются JVM. Это интерфейсы:

* java.lang.Cloneable
* java.io.Serializable
* java.rmi.Remote

#### Шаблоны в Java (generics)

Начиная с версии Java 5 в языке появился механизм [обобщённого программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — шаблоны, внешне близкие к шаблонам C++. С помощью специального синтаксиса в описании классов и методов можно указать параметры-типы, которые внутри описания могут использоваться в качестве типов полей, параметров и возвращаемых значений методов.

// Объявление обобщённого класса

class GenericClass<E> {

E getFirst() { ... }

void add(E obj) { ... }

}

// Использование обобщённого класса в коде

GenericClass<String> var = new GenericClass<String>();

var.add("qwerty");

String p = var.getFirst();

Допускается обобщённое объявление классов, интерфейсов и методов. Кроме того, синтаксис поддерживает ограниченные объявления типов-параметров: указание в объявлении конструкции вида <T extends A & B & C...> требует, чтобы тип-параметр T реализовывал интерфейсы A, B, C и так далее, а конструкция <T super C> требует, чтобы тип-параметр T был типом C или одним из его предков.

В отличие от шаблонов C#, шаблоны Java не поддерживаются средой исполнения — компилятор просто создаёт байт-код, в котором никаких шаблонов уже нет. Реализация шаблонов в Java принципиально отличается от реализации аналогичных механизмов в C++: компилятор не порождает для каждого случая использования шаблона отдельный вариант класса или метода-шаблона, а просто создаёт одну реализацию байт-кода, содержащую необходимые проверки и преобразования типов. Это приводит к ряду ограничений использования шаблонов в программах на Java.

#### Проверка принадлежности к классу

В Java можно явно проверить, к какому классу принадлежит объект. Выражение foo instanceof Foo истинно, если объект foo принадлежит классу Foo или его наследнику, или реализует интерфейс Foo (или, в общем виде, наследует класс, который реализует интерфейс, который наследует Foo).

Далее, функция getClass(), определённая для всех объектов, выдаёт объект типа Class. Для каждого класса создаётся не более одного описывающего его объекта типа Class, поэтому эти объекты можно сравнивать. Так, например, foo.getClass() == bar.getClass() будет истинно, если объекты foo и bar принадлежат в точности к одному классу (но это не означает что это два одинаковых объекта).

Кроме того, объект типа Class любого типа можно получить так: Integer.class, Object.class.

Однако прямое сравнение классов не всегда является оптимальным средством проверки на принадлежность к классу. Зачастую вместо него используют функцию isAssignableFrom(). Эта функция определена у объекта типа Class и принимает объект типа Class в качестве параметра. Таким образом, вызов Foo.class.isAssignableFrom(Bar.class) вернёт true в случае, если Foo является предком класса Bar. Так как все объекты являются потомками типа Object, вызов Object.class.isAssignableFrom() всегда вернёт true.

В паре с упомянутыми функциями объекта типа Class используются также функции isInstance() (эквивалентно instanceof), а также cast() (преобразует параметр в объект выбранного класса).

## 4.11 Библиотеки классов

* [JDK](http://ru.wikipedia.org/wiki/JDK) — содержит набор базовых классов для всего ПО на платформах [Java SE](http://ru.wikipedia.org/wiki/Java_SE) и [Java EE](http://ru.wikipedia.org/wiki/Java_EE)
* [CDK](http://ru.wikipedia.org/wiki/Chemistry_Development_Kit) — для создания химического ПО
* [MARF](http://ru.wikipedia.org/wiki/MARF) — модульная библиотека для распознавания аудио
* [JDOM](http://ru.wikipedia.org/wiki/JDOM) — реализация XML DOM, учитывающая особенности платформы Java.

## 4.12 Средства разработки ПО

* [JDK](http://ru.wikipedia.org/wiki/JDK) — помимо набора библиотек для платформ [Java SE](http://ru.wikipedia.org/wiki/Java_SE) и [Java EE](http://ru.wikipedia.org/wiki/Java_EE) содержит компилятор командной строки javac и набор утилит, также работающих в режиме командной строки.
* [NetBeans IDE](http://ru.wikipedia.org/wiki/NetBeans_IDE) — свободная интегрированная среда разработки для всех платформ Java — [Java ME](http://ru.wikipedia.org/wiki/Java_ME), [Java SE](http://ru.wikipedia.org/wiki/Java_SE) и [Java EE](http://ru.wikipedia.org/wiki/Java_EE). Пропагандируется [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems), разработчиком Java, как базовое средство для разработки ПО на языке Java и других языках ([C](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29), [C++](http://ru.wikipedia.org/wiki/C%2B%2B), [PHP](http://ru.wikipedia.org/wiki/PHP), [Fortran](http://ru.wikipedia.org/wiki/Fortran) и др.).
* [Eclipse IDE](http://ru.wikipedia.org/wiki/Eclipse_%28%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%29) — свободная интегрированная среда разработки для [Java SE](http://ru.wikipedia.org/wiki/Java_SE), [Java EE](http://ru.wikipedia.org/wiki/Java_EE) и [Java ME](http://ru.wikipedia.org/wiki/Java_ME). Пропагандируется [IBM](http://ru.wikipedia.org/wiki/IBM), одним из важнейших разработчиков корпоративного ПО, как базовое средство для разработки ПО на языке Java и других языках ([C](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29), [C++](http://ru.wikipedia.org/wiki/C%2B%2B), [Ruby](http://ru.wikipedia.org/wiki/Ruby), [Fortran](http://ru.wikipedia.org/wiki/Fortran) и др.)
* [IntelliJ IDEA](http://ru.wikipedia.org/wiki/IntelliJ_IDEA) — среда разработки для платформ [Java SE](http://ru.wikipedia.org/wiki/Java_SE), [Java EE](http://ru.wikipedia.org/wiki/Java_EE) и [Java ME](http://ru.wikipedia.org/wiki/Java_ME). Разработчик — компания [JetBrains](http://ru.wikipedia.org/wiki/JetBrains). Распространяется в двух версиях: свободной бесплатной (Community Edition) и коммерческой проприетарной (Ultimate Edition).
* [JDeveloper](http://ru.wikipedia.org/wiki/JDeveloper) — среда разработки для платформ [Java SE](http://ru.wikipedia.org/wiki/Java_SE), [Java EE](http://ru.wikipedia.org/wiki/Java_EE) и [Java ME](http://ru.wikipedia.org/wiki/Java_ME). Разработчик — компания [Oracle](http://ru.wikipedia.org/wiki/Oracle).
* [BlueJ](http://ru.wikipedia.org/wiki/BlueJ) — Среда разработки программного обеспечения на языке Java, созданная в основном для использования в обучении, но также подходящая для разработки небольших программ.

# БД и ее связь с Java

**База данных** — представленная в [объективной](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%28%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F%29) форме [совокупность](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) самостоятельных материалов ([статей](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F), [расчётов](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), [нормативных актов](http://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B0%D0%BA%D1%82), [судебных решений](http://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5) и иных подобных материалов), [систематизированных](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) таким образом, чтобы эти материалы могли быть [найдены](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [обработаны](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с помощью [электронной вычислительной машины (ЭВМ)](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) ([Гражданский кодекс РФ, ст. 1260](http://ru.wikisource.org/wiki/ru:%D0%93%D1%80%D0%B0%D0%B6%D0%B4%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4%D0%B5%D0%BA%D1%81_%D0%A0%D0%A4/%D0%93%D0%BB%D0%B0%D0%B2%D0%B0_70#.D0.A1.D1.82.D0.B0.D1.82.D1.8C.D1.8F_1260._.D0.9F.D0.B5.D1.80.D0.B5.D0.B2.D0.BE.D0.B4.D1.8B.2C_.D0.B8.D0.BD.D1.8B.D0.B5_.D0.BF.D1.80.D0.BE.D0.B8.D0.B7.D0.B2.D0.BE.D0.B4.D0.BD.D1.8B.D0.B5_.D0.BF.D1.80.D0.BE.D0.B8.D0.B7.D0.B2.D0.B5.D0.B4.D0.B5.D0.BD.D0.B8.D)).

Другие определения из авторитетных [монографий](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и [стандартов](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82):

* **База данных** — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность [данных](http://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), характеризующая актуальное состояние некоторой [предметной области](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C) и используемая для удовлетворения информационных [потребностей](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%BD%D0%BE%D1%81%D1%82%D1%8C) пользователей.[[1]](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#cite_note-KogMR-0)
* **База данных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.
* **База данных** — некоторый набор перманентных (постоянно хранимых) данных, используемых прикладными программными системами какого-либо предприятия.
* **База данных** — совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

Существует множество других определений, отражающих скорее [субъективное](http://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%28%D1%84%D0%B8%D0%BB%D0%BE%D1%81%D0%BE%D1%84%D0%B8%D1%8F%29) мнение тех или иных авторов о том, что означает база данных (БД) в их понимании, однако общепризнанная единая формулировка отсутствует. Наиболее часто используются следующие отличительные [признаки](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B7%D0%BD%D0%B0%D0%BA):

1. *БД хранится и обрабатывается в* [вычислительной системе](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80).  
   Таким образом, любые внекомпьютерные хранилища информации ([архивы](http://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D0%B2), [библиотеки](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0), [картотеки](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%82%D0%BE%D1%82%D0%B5%D0%BA%D0%B0) и т. п.) базами данных не являются.
2. *Данные в БД логически структурированы (*[систематизированы](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)*)* с целью обеспечения возможности их эффективного поиска и обработки в вычислительной системе.  
   Структурированность подразумевает явное выделение составных частей ([элементов](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82)), связей между ними, а также [типизацию](http://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1) элементов и связей, при которой с типом элемента (связи) соотносится определённая [семантика](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0) и допустимые операции.
3. *БД включает* [*метаданные*](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)*, описывающие* логическую структуру БД в формальном виде (в соответствии с некоторой [метамоделью](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)).  
   В соответствии с *ГОСТ Р ИСО МЭК ТО 10032-2007*, «постоянные данные в среде базы данных включают в себя [схему](http://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и базу данных. Схема включает в себя описания содержания, структуры и ограничений [целостности](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), используемые для создания и поддержки базы данных. База данных включает в себя набор постоянных данных, определенных с помощью схемы. Система управления данными использует определения данных в схеме для обеспечения доступа и управления доступом к данным в базе данных».

Из перечисленных признаков только первый является строгим, а другие допускает различные трактовки и различные степени оценки. Можно лишь установить некоторую степень соответствия требованиям к БД.

В такой ситуации не последнюю роль играет общепринятая практика. В соответствии с ней, например, не называют базами данных *файловые архивы*, [Интернет-порталы](http://ru.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D0%BE%D1%80%D1%82%D0%B0%D0%BB&action=edit&redlink=1) или [электронные таблицы](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0), несмотря на то, что они в некоторой степени обладают признаками БД. Принято считать, что эта степень в большинстве случаев недостаточна (хотя могут быть исключения).

Многие специалисты указывают на распространённую ошибку, состоящую в некорректном использовании термина «база данных» вместо термина «[система управления базами данных](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)», и указывают на необходимость различения этих понятий.

## 5.1 История

История возникновения и развития технологий баз данных может рассматриваться как в широком, так и в узком аспекте.

В **широком аспекте** понятие истории баз данных обобщается до истории любых средств, с помощью которых человечество хранило и обрабатывало данные. В таком контексте упоминаются, например, средства учёта царской казны и налогов в древнем [Шумере](http://ru.wikipedia.org/wiki/%D0%A8%D1%83%D0%BC%D0%B5%D1%80) (4000 г. до [н. э.](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%88%D0%B0_%D1%8D%D1%80%D0%B0)), узелковая письменность [инков](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B8) — [кипу](http://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BF%D1%83), [клинописи](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%BD%D0%BE%D0%BF%D0%B8%D1%81%D1%8C), содержащие документы [Ассирийского царства](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%B8%D1%80%D0%B8%D0%B9%D1%81%D0%BA%D0%BE%D0%B5_%D1%86%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE) и т. п. Следует помнить, что недостатком этого подхода является размывание понятия «база данных» и фактическое его слияние с понятиями «[архив](http://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D0%B2)» и даже «[письменность](http://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%81%D1%8C%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C)».

История баз данных в **узком аспекте** рассматривает базы данных в традиционном (современном) понимании. Эта история начинается с [1955 года](http://ru.wikipedia.org/wiki/1955_%D0%B3%D0%BE%D0%B4), когда появилось программируемое оборудование обработки записей. [Программное обеспечение](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) этого времени поддерживало модель обработки записей на основе файлов. Для хранения данных использовались [перфокарты](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%84%D0%BE%D0%BA%D0%B0%D1%80%D1%82%D0%B0).

Оперативные сетевые базы данных появились в середине [1960-х](http://ru.wikipedia.org/wiki/1960-%D0%B5). Операции над оперативными базами данных обрабатывались в интерактивном режиме с помощью терминалов. Простые индексно-последовательные организации записей быстро развились к более мощной модели записей, ориентированной на наборы. За руководство работой [Data Base Task Group](http://ru.wikipedia.org/wiki/Data_Base_Task_Group) (DBTG), разработавшей стандартный [язык описания данных](http://ru.wikipedia.org/wiki/Data_Definition_Language) и [манипулирования данными](http://ru.wikipedia.org/wiki/Data_Manipulation_Language), [Чарльз Бахман](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D1%85%D0%BC%D0%B0%D0%BD,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7_%D0%A3%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC) получил [Тьюринговскую премию](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BC%D0%B8%D1%8F_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0).

В это же время в сообществе баз данных [COBOL](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B1%D0%BE%D0%BB) была проработана концепция [схем баз данных](http://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и концепция независимости данных.

Следующий важный этап связан с появлением в начале [1970-х](http://ru.wikipedia.org/wiki/1970-%D0%B5) [реляционной модели данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), благодаря работам [Эдгара Ф. Кодда](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B4,_%D0%AD%D0%B4%D0%B3%D0%B0%D1%80). Работы Кодда открыли путь к тесной связи прикладной технологии баз данных с математикой и логикой. За свой вклад в теорию и практику Эдгар Ф. Кодд также получил [премию Тьюринга](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BC%D0%B8%D1%8F_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0).

Сам термин database (база данных) появился в начале 1960-х годов, и был введён в употребление на симпозиумах, организованных фирмой SDC (System Development Corporation) в [1964](http://ru.wikipedia.org/wiki/1964_%D0%B3%D0%BE%D0%B4) и [1965](http://ru.wikipedia.org/wiki/1965_%D0%B3%D0%BE%D0%B4) годах, хотя понимался сначала в довольно узком смысле, в контексте систем [искусственного интеллекта](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82). В широкое употребление в современном понимании термин вошёл лишь в 1970-е годы.

## 5.2 Виды баз данных

Существует огромное количество разновидностей баз данных, отличающихся по различным [критериям](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B9). Например, в «Энциклопедии технологий баз данных»,[[1]](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#cite_note-KogMR-0) по материалам которой написан данный раздел, определяются свыше 50 видов БД.

Основные классификации приведены ниже.

### 5.3 Классификация по [модели данных](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)

Примеры:

* [Иерархическая](http://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
* [Сетевая](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
* [Реляционная](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
* [Объектная](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [объектно-ориентированная](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)
* [Объектно-реляционная](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94)
* [Функциональная](http://ru.wikipedia.org/w/index.php?title=%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1).

### 5.4 Классификация по среде постоянного хранения

* Во вторичной памяти, или традиционная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *conventional database*): средой постоянного хранения является периферийная энергонезависимая память (вторичная память) — как правило [жёсткий диск](http://ru.wikipedia.org/wiki/%D0%96%D1%91%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA).  
  В оперативную память СУБД помещает лишь [кеш](http://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D1%88) и данные для текущей обработки.
* В оперативной памяти ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) in-memory database, memory-resident database, main memory database): все данные на стадии исполнения находятся в [оперативной памяти](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C).
* В третичной памяти ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) tertiary database): средой постоянного хранения является отсоединяемое от сервера устройство массового хранения (третичная память), как правило на основе [магнитных лент](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B8%D0%BC%D0%B5%D1%80) или [оптических дисков](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA).  
  Во вторичной памяти сервера хранится лишь каталог данных третичной памяти, файловый кеш и данные для текущей обработки; загрузка же самих данных требует специальной процедуры.

### 5.5 Классификация по содержимому

Примеры:

* Географическая
* Историческая
* Научная
* Мультимедийная.

### 5.6 Классификация по степени распределённости

* Централизованная, или сосредоточенная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) centralized database): БД, полностью поддерживаемая на одном компьютере.
* [Распределённая](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) distributed database): БД, составные части которой размещаются в различных узлах компьютерной сети в соответствии с каким-либо критерием.
  + Неоднородная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) heterogeneous distributed database): фрагменты распределённой БД в разных узлах сети поддерживаются средствами более одной СУБД
  + Однородная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) homogeneous distributed database): фрагменты распределённой БД в разных узлах сети поддерживаются средствами одной и той же СУБД.
  + Фрагментированная, или секционированная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) partitioned database): методом распределения данных является фрагментирование (партиционирование, [секционирование](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), вертикальное или горизонтальное.
  + Тиражированная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) replicated database): методом распределения данных является тиражирование ([репликация](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%28%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0%29)).

### 5.7 Другие виды БД

* [Пространственная](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [spatial database](http://meta.wikimedia.org/wiki/w:en:spatial_database)): БД, в которой поддерживаются пространственные свойства сущностей предметной области. Такие БД широко используются в [геоинформационных системах](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).
* Временная, или темпоральная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) temporal database): БД, в которой поддерживается какой-либо аспект времени, не считая времени, определяемого пользователем.
* Пространственно-временная ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) spatial-temporal database) БД: БД, в которой одновременно поддерживается одно или более измерений в аспектах как пространства, так и времени.
* [Циклическая](http://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) round-robin database): БД, объём хранимых данных которой не меняется со временем, поскольку в процессе сохранения данных одни и те же записи используются циклически.

## 5.8 Сверхбольшие базы данных

**Сверхбольшая база данных** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Very Large Database, VLDB) — это база данных, которая занимает чрезвычайно большой объём на [устройстве физического хранения](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C). Термин подразумевает максимально возможные объёмы БД, которые определяются последними достижениями в технологиях физического хранения данных и в технологиях программного оперирования данными.

Количественное определение понятия «чрезвычайно большой объём» меняется во времени; в настоящее время считается, что это объём, измеряемый по меньшей мере [петабайтами](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%82%D0%B0%D0%B1%D0%B0%D0%B9%D1%82). Для сравнения, в 2005 г. самыми крупными в мире считались базы данных с объёмом хранилища порядка 100 [терабайт](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%B0%D0%B1%D0%B0%D0%B9%D1%82).

Специалисты отмечают необходимость особых подходов к [проектированию](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) сверхбольших БД. Для их создания нередко выполняются специальные [проекты](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82) с целью поиска таких [системотехнических](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0) решений, которые позволили бы хоть как-то работать с такими большими объёмами данных. Как правило необходимы специальные решения для дисковой подсистемы, специальные версии [операционной среды](http://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0&action=edit&redlink=1) и специальные механизмы обращения [СУБД](http://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94) к данным.

Исследования в области хранения и обработки сверхбольших баз данных VLDB всегда находятся на острие теории и практики баз данных. В частности, с [1975 года](http://ru.wikipedia.org/wiki/1975_%D0%B3%D0%BE%D0%B4) проходит ежегодная конференция International Conference on Very Large Data Bases («Международная конференция по сверхбольшим базам данных»). Большинство исследований проводится под эгидой некоммерческой организации VLDB Endowment (Фонд целевого капитала «VLDB»), которая обеспечивает продвижение научных работ и обмен информацией в области сверхбольших БД и смежных областях.

## 5.9 JDBC - средство общения между Java и базами данных

Базы данных… они повсюду. От них просто нет никакого спасения. Сегодня мало кто не сталкивался с программированием приложений, которые используют базы данных. Начиная от простых (а порой и сложных) текстовых файлов с собственной структурой, заканчивая до боли знакомыми SQL-ориентированными СУБД. Тут собственно разработчики Java не могли остаться в стороне (ну а как иначе?) и написали интерфейс для взаимодействия Java-приложений с базами данных. Тут и далее под базами данных будем понимать СУБД использующие SQL.

Случилось это сравнительно давно. Потому JDBC немного изменился, в лучшую сторону.

Рассматривая данный интерфейс, можно провести параллель с известным DBI/DBD, языка Perl. Сходство весьма заметно. И тот и другой работают по практически идентичной схеме: имея единый интерфейс, подключаем драйвер для работы с определенной СУБД и собственно начинаем писать. Ежели нам нужно сменить СУБД - меняем драйвер. Как это реализовано в Perl, нам пока мало интересно, а вот JDBC рассмотрим поподробнее ;).

Для начала установим себе драйвер, используемой нами базы данных. Пусть это будет PostgreSQL. Любой JDBC драйвер может представлять собой jar-архив. Для того чтобы из своей программы можно было бы его использовать, нужно, чтобы переменная окружения CLASSPATH содержала путь к нему. Делается это примерно так:

export CLASSPATH=$CLASSPATH:/usr/local/lib/postgresql.jar

Это в том случае, если у вас стоит Linux. В любом другом случае можно обратиться к документации J.   
  
Перейдем непосредственно к использованию нашего JDBC по назначению. Любая программа, которая предполагает использовать JDBC должна импортировать пакеты java.sql.\*:

**import** java.sql.\*;

Но при помощи этих пакетов мы не сможем осуществить связь с нашей СУБД. Для этого нам нужен тот самый драйвер, который мы так аккуратно положили в /usr/local/lib/.

Для загрузки/использования нашего драйвера можно пойти двумя путями. Первый – это вписать имя драйвера в код программы. В этом случае мы не сможем изменить СУБД без перекомпиляции. Второй – это подключать драйвер из командной строки Java (JVM).   
  
Следующей строчкой мы загружаем драйвер, и он автоматически зарегистрируется себя для использования вместе с JDBC.

Class.forName(“postgresql.Driver”);

Во втором же случае мы используем ключ –D для подключения драйвера.

java –Djdbc.drivers=postgresql.Driver myClass

Выбирайте любой способ, ориентируясь на то, нужно ли вам будет в дальнейшем переходить на другую СУБД. Если да, то я думаю, второй способ будет предпочтительнее. Особенно для тех, кто пишет программы для широкой публики. А ей свойственно хотеть самим выбирать, чем пользоваться.

Внимание, если вы все сделали одним из этих способов, но при выполнении ваше программы вы получаете ошибку No driver available - это, скорее всего, означает, что вы просто не внесли путь к драйверу в переменную CLASSPATH.

Для соединения с базой данных используют класс Connection:

**Connection** dbh = DriverManager.getConnection(url, user, passwd);

Здесь url - это строка, по которой JDBC определяет куда, где и чем устанавливать соединение. Она имеет следующий формат:

jdbc:postgresql:база\_данных

jdbc:postgresql://сервер/база\_данных

jdbc:postgresql://сервер:порт/база\_данных

Где сервер – это адрес сервера, где расположена СУБД, порт – это tcp-порт, на котором будет устанавливаться соединение и база\_данных – это имя базы данных, которую мы хотим использовать. Следует отметить, что в DBI/DBD эта строчка имела вид:

dbi:postgres:dbname=база\_данных;host=сервер;port=порт;

Как видно они очень схожи. Вместо postgres/postgresql может стоять любой, выбранный вами драйвер, для работы с вашей СУБД, например mySQL.

В результате имеем, например:

**Connection** dbh = DriverManager.getConnection(

"jdbc:postgresql://127.0.0.1/flowers\_db","alex","123456"

);

Установив соединение, мы можем выполнять sql-запросы к нашей базе данных. Для примера, рассмотрим небольшой кусок кода на Java:

**Statement** st = dbh.createStatement();

**ResultSet** rs = st.executeQuery(

"select \* from flowers\_tbl where (flowers\_id in (1,2,3,4,5,6,7,8,9))"

);

**while** (rs.next())

{

System.out.println(rs.getString(1));

}

rs.close();

st.close();

Из этого примера хорошо видно, что для того, чтобы посылать запросы к базе данных, необходимо создать экземпляр класса Statement st. Метод executeQuery этого класса отправляет переданный ему запрос к базе данных и в качестве ответа возвращает результат в виде класса ResultSet.

В общем случае вы можете использовать только один экземпляр класса Statement, если конечно вам по какой-либо серьезной причине не понадобится второй. В случае же работы с потоками, дело обстоит иначе. Здесь обязательно иметь на каждый поток один экземпляр Statement. О потоках речь пойдет в других статьях.

Перед тем как получать данные результата запроса, которые хранятся в нашем случае в переменной rs, необходимо выполнить метод next(). После выполнения этого метода, вы получаете очередную запись виртуально SQL-таблицы ответа на запрос. Метод next() возвращает true, если на запрос были получены данные или еще есть данные в стеке ответа. В противном случаете next() возвращает false.

Ну а метод getString() возвращает строковую переменную, значения поля записи с заданным номером. В нашем случае это 1. После использования результатов переменную rs надо закрыть методом close(), чтобы потом можно было повторно ее использовать для других запросов.

В спецификации JDBC вы можете использовать ResultSet, только один раз последовательно прочитав значения. Это означает, что, взяв однажды значение, вы не можете получить его снова. Для этого лучше использовать дополнительную переменную. Хотя многие драйвера СУБД позволяют снять это ограничения, я все же советую вам использовать вспомогательную переменную, нежели полагаться на драйвер. Это просто накладывает ограничение на переносимость между различными СУБД.

Метод executeQuery() удобен лишь в том случае, когда мы ожидаем ответа от базы данных. Специально для запросов, на которые ответ не обязателен в JDBC был введен еще один метод класса Statement. Это – executeUpdate(). К запросам такого типа относится UPDATE, CREATE, INSERT и пр. С ними можно было познакомиться, прочитав мою статью про SQL в одном из номеров КГ. Приведем пример использования:

st.executeUpdate("create database flowers");

st.executeUpdate("create table flowers\_tbl (id integer, name char(25))");

Выше мы рассмотрели возможности JDBC для выполнения SQL-запросов обычным образом. Однако этим возможности JDBC не ограничиваются. Очень многие из них останутся так и не затронутыми в этой статье, о которых вы можете узнать из официальной спецификации JDBC. А нам еще надо рассмотреть возможность предварительной подготовки SQL запросов с последующей подстановкой в них значений. Для этого вместо Statement нужно создать экземпляр класса PreparedStatement:

**PreparedStatement** ps = dbh.prepareStatement("insert into flowers\_tbl values(?,?)");

После чего методом setString() заменяем знаки '?' на значения:

**int** a = 1;

**String** b = "rose";

ps.setString(1,a);

ps.setString(2,b);

И после чего выполняем метод executeUpdate(). Однако, если бы нам нужно было бы вернуть ответ базы данных, то вместо метода executeUpdate() мы используем executeQuery() и тоже без параметров. Вот как это выглядит в обоих случаях:

ps.executeUpdate()

// или

**ResultSet** rs = ps.exexuteQuery();

Далее вы знаете. Такой механизм в DBI/DBD называется методом курсоров. С другой стороны можно было бы просто сформировать строчку с нужными переменными и выполнить простой Statement, но это куда больше работы, чем с использованием PreparedStatement.

# Создание сайта при помощи html и css

## 6.1 Возможности HTML

**HTML** (от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HyperText Markup Language — «язык разметки [гипертекста](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82)») — стандартный [язык разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](http://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) создаются при помощи языка HTML (или [XHTML](http://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) и отображается в виде документа, в удобной для человека форме.

HTML является приложением («частным случаем») [SGML](http://ru.wikipedia.org/wiki/SGML) (стандартного обобщённого языка разметки) и соответствует международному стандарту [ISO](http://ru.wikipedia.org/wiki/ISO) 8879. [XHTML](http://ru.wikipedia.org/wiki/XHTML) же является приложением [XML](http://ru.wikipedia.org/wiki/XML).

## 6.1.1 Общее представление

Язык HTML был разработан [британским](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D1%8F) учёным [Тимом Бернерсом-Ли](http://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%BD%D0%B5%D1%80%D1%81-%D0%9B%D0%B8,_%D0%A2%D0%B8%D0%BC) приблизительно в [1989](http://ru.wikipedia.org/wiki/1989)—[1991 годах](http://ru.wikipedia.org/wiki/1991_%D0%B3%D0%BE%D0%B4) в стенах [Европейского совета по ядерным исследованиям](http://ru.wikipedia.org/wiki/CERN) в [Женеве](http://ru.wikipedia.org/wiki/%D0%96%D0%B5%D0%BD%D0%B5%D0%B2%D0%B0) ([Швейцария](http://ru.wikipedia.org/wiki/%D0%A8%D0%B2%D0%B5%D0%B9%D1%86%D0%B0%D1%80%D0%B8%D1%8F)). HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области [вёрстки](http://ru.wikipedia.org/wiki/%D0%92%D1%91%D1%80%D1%81%D1%82%D0%BA%D0%B0). HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и [семантических](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0) элементов — [дескрипторов](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%B3_%28%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8%29). Дескрипторы также часто называют «[тегами](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%B3_%28%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8%29)». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. [Мультимедийные](http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0) возможности были добавлены позже.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <TABLE>, несколько раз использованный для форматирования страницы, которую вы на даный момент читаете, предназначен для создания в документах самых обычных таблиц, но, как можно убедиться, здесь нет ни одной таблицы. С течением времени, основная идея платформонезависимости языка HTML была отдана в своеобразную жертву современным потребностям в мультимедийном и графическом оформлении.

## 6.1.2 Браузеры

Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют [расширение](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8%D0%BC%D0%B5%D0%BD%D0%B8_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0) .html или .htm), обрабатываются специальными [приложениями](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), которые отображают документ в его форматированном виде. Такие приложения, называемые «браузерами» или «интернет-обозревателями», обычно предоставляют пользователю удобный [интерфейс](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) для запроса [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0), их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами являются [Internet Explorer](http://ru.wikipedia.org/wiki/Internet_Explorer), [Mozilla Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox), [Apple Safari](http://ru.wikipedia.org/wiki/Apple_Safari), [Google Chrome](http://ru.wikipedia.org/wiki/Google_Chrome) и [Opera](http://ru.wikipedia.org/wiki/Opera) (см.: [Браузер#Рыночные доли](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80#.D0.A0.D1.8B.D0.BD.D0.BE.D1.87.D0.BD.D1.8B.D0.B5_.D0.B4.D0.BE.D0.BB.D0.B8)).

## 6.1.3 Версии

* [RFC 1866](http://tools.ietf.org/html/rfc1866) — HTML 2.0, одобренный как стандарт [22 сентября](http://ru.wikipedia.org/wiki/22_%D1%81%D0%B5%D0%BD%D1%82%D1%8F%D0%B1%D1%80%D1%8F) [1995 года](http://ru.wikipedia.org/wiki/1995_%D0%B3%D0%BE%D0%B4);
* HTML 3.2 — [14 января](http://ru.wikipedia.org/wiki/14_%D1%8F%D0%BD%D0%B2%D0%B0%D1%80%D1%8F) [1997 года](http://ru.wikipedia.org/wiki/1997_%D0%B3%D0%BE%D0%B4);
* HTML 4.0 — [18 декабря](http://ru.wikipedia.org/wiki/18_%D0%B4%D0%B5%D0%BA%D0%B0%D0%B1%D1%80%D1%8F) 1997 года;
* HTML 4.01 (изменения, причём более значительные, чем кажется на первый взгляд) — [24 декабря](http://ru.wikipedia.org/wiki/24_%D0%B4%D0%B5%D0%BA%D0%B0%D0%B1%D1%80%D1%8F) [1999 года](http://ru.wikipedia.org/wiki/1999_%D0%B3%D0%BE%D0%B4);
* ISO/IEC 15445:2000[[4]](http://ru.wikipedia.org/wiki/HTML#cite_note-3) (так называемый ISO HTML, основан на HTML 4.01 Strict) — [15 мая](http://ru.wikipedia.org/wiki/15_%D0%BC%D0%B0%D1%8F) [2000 года](http://ru.wikipedia.org/wiki/2000_%D0%B3%D0%BE%D0%B4).
* [HTML 5](http://ru.wikipedia.org/wiki/HTML_5) — в разработке. Конец разработки запланирован на 2014 год.

Официальной спецификации HTML 1.0 не существует. До 1995 года существовало множество неофициальных стандартов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.

Версия 3 была предложена [Консорциумом всемирной паутины](http://ru.wikipedia.org/wiki/W3C) (W3C) в марте 1995 года и обеспечивала много новых возможностей, таких как создание таблиц, «обтекание» изображений текстом и отображение сложных [математических формул](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D0%B0). Даже при том, что этот стандарт был совместим со второй версией, реализация его была сложна для браузеров того времени. Версия 3.1 официально никогда не предлагалась, и следующей версией стандарта HTML стала 3.2, в которой были опущены многие нововведения версии 3.0, но добавлены нестандартные элементы, поддерживаемые браузерами [Netscape Navigator](http://ru.wikipedia.org/wiki/Netscape_Navigator) и [Mosaic](http://ru.wikipedia.org/wiki/Mosaic).

В версии HTML 4.0 произошла некоторая «очистка» стандарта. Многие элементы были отмечены как устаревшие и нерекомендованные ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) deprecated). В частности, элемент font, используемый для изменения свойств [шрифта](http://ru.wikipedia.org/wiki/%D0%A8%D1%80%D0%B8%D1%84%D1%82), был помечен как устаревший (вместо него рекомендуется использовать таблицы стилей [CSS](http://ru.wikipedia.org/wiki/CSS)).

В [1998 году](http://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4) [консорциум Всемирной паутины](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D1%80%D1%86%D0%B8%D1%83%D0%BC_%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D1%8B) начал работу над новым языком разметки, основанном на HTML 4, но соответствующим синтаксису XML. Впоследствии новый язык получил название [XHTML](http://ru.wikipedia.org/wiki/XHTML). Первая версия XHTML 1.0 одобрена в качестве Рекомендации консорциума Всемирной паутины [26 января](http://ru.wikipedia.org/wiki/26_%D1%8F%D0%BD%D0%B2%D0%B0%D1%80%D1%8F) [2000 года](http://ru.wikipedia.org/wiki/2000_%D0%B3%D0%BE%D0%B4).

Планируемая версия XHTML 2.0 должна была разорвать совместимость со старыми версиями HTML и XHTML, но 2 июля [2009 года](http://ru.wikipedia.org/wiki/2009_%D0%B3%D0%BE%D0%B4) [консорциум Всемирной паутины](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D1%80%D1%86%D0%B8%D1%83%D0%BC_%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D1%8B) объявил, что полномочия рабочей группы XHTML2 истекают в конце [2009 года](http://ru.wikipedia.org/wiki/2009_%D0%B3%D0%BE%D0%B4). Таким образом, была приостановлена вся дальнейшая разработка стандарта XHTML 2.0.

## 6.1.4 Перспективы

В настоящее время [Консорциум всемирной паутины](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D1%80%D1%86%D0%B8%D1%83%D0%BC_%D0%B2%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D1%8B) разрабатывает [HTML версии 5](http://ru.wikipedia.org/wiki/HTML5). Черновой вариант спецификации языка появился в Интернете [20 ноября](http://ru.wikipedia.org/wiki/20_%D0%BD%D0%BE%D1%8F%D0%B1%D1%80%D1%8F) [2007 года](http://ru.wikipedia.org/wiki/2007_%D0%B3%D0%BE%D0%B4).

Сообществом [WHATWG](http://ru.wikipedia.org/wiki/WHATWG) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Web Hypertext Application Technology Working Group), начиная с [2004 года](http://ru.wikipedia.org/wiki/2004_%D0%B3%D0%BE%D0%B4), разрабатывается спецификация [Web Applications](http://ru.wikipedia.org/w/index.php?title=Web_Applications&action=edit&redlink=1) 1.0, часто неофициально называемая «HTML 5», которая расширяет HTML (впрочем, имея и совместимый с XHTML 1.0 XML-синтаксис) для лучшего представления семантики различных типичных страниц, например [форумов](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%84%D0%BE%D1%80%D1%83%D0%BC), сайтов [аукционов](http://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%BA%D1%86%D0%B8%D0%BE%D0%BD), [поисковых систем](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), онлайн-магазинов и т. д., которые не очень удачно вписываются в модель XHTML 2.

## 6.1.5 Структура HTML-документа

HTML — теговый язык разметки [документов](http://ru.wikipedia.org/wiki/HTML-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82). Любой документ на языке HTML представляет собой набор [элементов](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_HTML), причём начало и конец каждого элемента обозначается специальными пометками — [тегами](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%B3_%28%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8%29). Элементы могут быть *пустыми*, то есть не содержащими никакого текста и других данных (например, тег перевода строки <br>). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь *атрибуты*, определяющие какие-либо их свойства (например, размер шрифта для элемента font). Атрибуты указываются в открывающем теге. Вот примеры фрагментов HTML-документа:

* <strong>Текст между двумя тегами — открывающим и закрывающим.</strong>
* <a href="http://www.example.com">Здесь элемент содержит атрибут href.</a>
* А вот пример пустого элемента: <br>

Регистр, в котором набрано имя элемента и имена атрибутов, в HTML значения не имеет (в отличие от XHTML). Элементы могут быть вложенными. Например, следующий код:

<b>

Этот текст будет полужирным,

<i>а этот - ещё и курсивным</i>

</b>

даст такой результат:

**Этот текст будет полужирным, *а этот — ещё и курсивным***

Кроме элементов, в HTML-документах есть и сущности ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) entities) — [«специальные символы»](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%B5%D0%BC%D0%BE%D0%BD%D0%B8%D0%BA%D0%B8_%D0%B2_HTML). Сущности начинаются с символа [амперсанда](http://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%BF%D0%B5%D1%80%D1%81%D0%B0%D0%BD%D0%B4) и имеют вид &имя; или &#NNNN;, где NNNN — код символа в [Юникоде](http://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4) в десятичной системе счисления.

Например, © — знак [авторского права](http://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE) (©). Как правило, сущности используются для представления символов, отсутствующих в кодировке документа, или же для представления «специальных» символов: &amp; — амперсанда (&), &lt; — символа «меньше» (<) и &gt; — символа «больше» (>), которые некорректно записывать «обычным» образом, из-за их особого значения в HTML.

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, должен начинаться со строки объявления версии HTML <!DOCTYPE…>, которая обычно выглядит примерно так:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

Если эта строка не указана, то добиться корректного отображения документа в браузере становится труднее.

Далее обозначается начало и конец документа тегами <html> и </html> соответственно. Внутри этих тегов должны находиться теги заголовка (<head></head>) и тела (<body></body>) документа.

## 6.2 Основные принципы использования CSS

**CSS** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Cascading Style Sheets* — **каскадные таблицы стилей**) — [формальный язык](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) описания внешнего вида документа, написанного с использованием [языка разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8).

Преимущественно используется как средство описания, оформления внешнего вида [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0), написанных с помощью [языков разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) [HTML](http://ru.wikipedia.org/wiki/HTML) и [XHTML](http://ru.wikipedia.org/wiki/XHTML), но может также применяться к любым [XML-документам](http://ru.wikipedia.org/wiki/XML), например, к [SVG](http://ru.wikipedia.org/wiki/SVG) или [XUL](http://ru.wikipedia.org/wiki/XUL).

### 6.2.1 Цель создания CSS

CSS используется создателями [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) для задания [цветов](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82), [шрифтов](http://ru.wikipedia.org/wiki/%D0%A8%D1%80%D0%B8%D1%84%D1%82), расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью [HTML](http://ru.wikipedia.org/wiki/HTML) или других [языков разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8)) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью [формального языка](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или [методах](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4) вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими [шрифт Брайля](http://ru.wikipedia.org/wiki/%D0%A8%D1%80%D0%B8%D1%84%D1%82_%D0%91%D1%80%D0%B0%D0%B9%D0%BB%D1%8F).

### 6.2.2 Способы подключения CSS к документу

Правила CSS пишутся на [формальном языке](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) CSS и располагаются в таблицах стилей, то есть таблицы стилей содержат в себе правила CSS. Эти таблицы стилей могут располагаться как в самом [веб-документе](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82), внешний вид которого они описывают, так и в отдельных [файлах](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), имеющих [формат](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0) CSS. (По сути, формат CSS — это обычный [текстовый файл](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB). В файле .css не содержится ничего, кроме перечня правил CSS и комментариев к ним.)  
То есть, эти таблицы стилей могут быть подключены, внедрены в описываемый ими [веб-документ](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82) четырьмя различными способами:

* когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством [тега](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_HTML) <link>, располагающегося в этом документе между тегами <head> и </head>. (Тег <link> будет иметь атрибут href, имеющий значением адрес этой таблицы стилей). Все правила этой таблицы действуют на протяжении всего документа;

<head>

.....

<link rel="stylesheet" type="text/css" href="style.css">

</head>

* когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством директивы @import, располагающейся в этом документе между тегами <style> и </style> (которые, в свою очередь, располагаются в этом документе между тегами <head> и </head>) сразу после тега <style>, которая также указывает (в своих скобках, после слова url) на адрес этой таблицы стилей. Все правила этой таблицы действуют на протяжении всего документа;

<head>

.....

<style type="text/css" media="all">

@import url(style.css);

</style>

</head>

* когда таблица стилей описана в самом документе, она может располагаться в нём между тегами <style> и </style> (которые, в свою очередь, располагаются в этом документе между тегами <head> и </head>). Все правила этой таблицы действуют на протяжении всего документа;

<head>

.....

<style type="text/css">

body {

color: red;

}

</style>

</head>

* когда таблица стилей описана в самом документе, она может располагаться в нём в теле какого-то отдельного тега (посредством его атрибута style) этого документа. Все правила этой таблицы действуют только на содержимое этого тега.

<p style="font-size: 21px; color: green;">Рассказ о том, как вредно красить батареи</p>

В первых двух случаях говорят, что к документу применены *внешние таблицы стилей*, а во вторых двух случаях — *внутренние таблицы стилей*.

Для добавления CSS к [XML-документу](http://ru.wikipedia.org/wiki/XML), последний должен содержать специальную ссылку на таблицу стилей. Например:

<?xml-stylesheet type="text/css" href="style.css"?>

### 6.2.3 Иерархия элементов внутри документа

Как известно, [HTML-документы](http://ru.wikipedia.org/wiki/HTML) строятся на основании иерархии элементов, которая может быть наглядно представлена в древовидной форме. Элементы HTML друг для друга могут быть родительскими, дочерними, элементами-предками, элементами-потомками, сестринскими.

Элемент является родителем другого элемента, если в иерархической структуре документа он находится сразу, непосредственно над этим элементом. Элемент является предком другого элемента, если в иерархической структуре документа он находится где-то выше этого элемента.  
Пускай, например, в документе присутствуют два абзаца p, включающие в себя шрифт с полужирным начертанием b. Тогда элементы b будут дочерними элементами своих родительских элементов p, и потомками своих предков body. В свою очередь, для элементов p элемент body будет являться только родителем. И кроме того, эти два элемента p будут являться сестринскими элементами, как имеющими одного и того же родителя — body.

В CSS могут задаваться, при помощи [селекторов](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D0%BE%D1%80), не только одиночные элементы, но и элементы, являющиеся потомками, дочерними или сестринскими элементами других элементов (см. подраздел «виды селекторов»).

### 6.2.4Построение правила CSS

В первых трёх случаях подключения таблицы CSS к документу (см. выше) каждое правило CSS из таблицы стилей имеет две основные части — *селектор* и *блок объявлений*. *Селектор*, расположенный в левой части правила, определяет, на какие части документа распространяется правило. *Блок объявлений* располагается в правой части правила. Он помещается в фигурные скобки, и, в свою очередь, состоит из одного или более *объявлений*, разделённых знаком «;». Каждое *объявление* представляет собой сочетание *свойства CSS* и *значения*, разделённых знаком «:». Селекторы могут группироваться в одной строке через запятую. В таком случае свойство применяется к каждому из них.

селектор, селектор {

свойство: значение;

свойство: значение;

свойство: значение;

}

В четвёртом случае подключения таблицы CSS к документу (см. список) правило CSS (являющееся значением атрибута style тега, на который оно действует) представляет собой перечень *объявлений* («*свойство CSS* : *значение*»), разделённых знаком «;».

#### Виды селекторов

Селекторы правила CSS могут быть

* селекторами элементов;

p {font-family: Garamond, serif;}

* селекторами классов;

.note {color: red; background: yellow; font-weight: bold;}

* селекторами идентификаторов;

#paragraph1 {margin: 0;}

* селекторами атрибутов;

a[href="http://www.somesite.com"]{font-weight:bold;}

* селекторами потомков (контекстными селекторами);

div#paragraph1 p.note {color: red;}

* селекторами дочерних элементов;

p.note > b {color: green;}

* селекторами сестринских элементов;

h1 + p {font-size: 24pt;}

* селекторами псевдоклассов;

a:active {color:yellow;}

* селекторами псевдоэлементов.

p:first-letter {font-size: 32px;}

* Также в CSS существует так называемый *универсальный селектор*, обозначающий любой элемент, встречающийся в документе. Например, \* {color:red;}. Перед любым селектором, задающим класс или идентификатор, можно поставить знак универсального селектора, в результате получится эквивалентное выражение, например, .first {...} и \*.first {...} имеют один и тот же смысл.

### 6.2.5 Классы элементов. Идентификаторы элементов.

Класс или идентификатор может быть присвоен какому-нибудь элементу (тегу) HTML посредством атрибутов class или id этого элемента (тега):

<div id="first"> ... </div>

<p class="big"> ... </p>

Основное отличие между классами элементов и идентификаторами элементов в том, что в документе какой-нибудь класс может быть присвоен сразу нескольким элементам, а идентификатор - только одному. Также отличие в том, что могут существовать множественные классы (когда класс элемента состоит из нескольких слов, разделённых пробелами). Для идентификаторов такое невозможно.

Важно отметить следующее отличие идентификатора от класса: идентификаторы широко используются в JavaScript для нахождения уникального элемента в документе.

Имена классов и идентификаторов, в отличие от названий тегов и их атрибутов, чувствительны к регистру ввода букв.

Свойства классов и идентификаторов задаются с помощью соответствующих селекторов. Причём может быть задано как свойство класса в целом (в таком случае селектор начинается с «.»), или свойство идентификатора самого по себе (в таком случае селектор начинается с «#»), так и свойство какого-нибудь элемента этого класса или с этим идентификатором.

В CSS помимо классов, задаваемых автором страницы, существует также ограниченный набор так называемых *псевдоклассов*, описывающих вид гиперссылок с определённым состоянием в документе, вид элемента, на котором находится фокус ввода, а также вид элементов, являющихся первыми дочерними элементами других элементов. Также в CSS существует четыре так называемых *псевдоэлемента*: первая буква, первая строка, применение специальных стилей до и после элемента.

### 6.2.6 Наследование. Каскадирование. Приоритеты стилей CSS.

Применение CSS к документам HTML основано на принципах *наследования* и *каскадирования*. Принцип *наследования* заключается в том, что свойства CSS, объявленные для элементов-предков, наследуются элементами потомками. Но, естественно, не все свойства CSS наследуются — например, если для тега параграфа p средствами CSS задана рамка, то она не будет наследоваться ни одним тегом, содержащимся в данном теге p, а вот если для параграфа p средствами CSS задан цвет шрифта (например, color:green;), то это свойство будет унаследовано каждым элементом-тегом, находящимся в параграфе.

Принцип *каскадирования* применяется в случае, когда какому-то элементу HTML одновременно поставлено в соответствие более одного правила CSS, то есть, когда происходит конфликт значений этих правил. Чтобы разрешить такие конфликты вводятся правила приоритета.

* Наиболее низким приоритетом обладает стиль браузера;
* Следующим по значимости является стиль, заданный пользователем браузера в его настройках;
* И наиболее высоким приоритетом обладает стиль, заданный непосредственно автором страницы. И далее, уже в этом авторском стиле приоритеты расставляются следующим образом:
  + Самым низким приоритетом обладают стили, наследуемые в документе элементом от своих предков;
  + Более высоким приоритетом обладают стили, заданные во внешних таблицах стилей, подключённых к документу;
  + Ещё более высоким приоритетом обладают стили, заданные непосредственно селекторами всех десяти видов (см. подраздел «виды селекторов»), содержащимися в контейнерах style данного документа. Нередки случаи, когда к какому-нибудь элементу имеют отношение, задают его вид, несколько таких селекторов. Такие конфликты между ними разрешаются с помощью расчёта специфичности каждого такого селектора и применения этих селекторов к данному элементу в порядке убывания их специфичностей. При расчёте специфичности селектора принимается во внимание:
    - количество идентификаторов (#id) в селекторе —- ((1,0,0) за каждый объявленный идентификатор в селекторе правила CSS);
    - количество классов (.class) и псевдоклассов (:pseudoclass) в селекторе —- ((0,1,0) за каждый объявленный класс и псевдокласс в селекторе правила CSS );
    - количество тегов в селекторе —- ((0,0,1) за каждый объявленный тег в селекторе правила CSS). *(Принцип расчёта таков, что, например, (1,0,0) будет иметь большую специфичность, соответственно — бо́льший приоритет, чем даже (0,10,0), а (0,1,0) будет иметь большую специфичность, больший приоритет, чем (0,0,10). Если же рассчитанные таким образом специфичности окажутся одинаковыми, то к элементу будет применено правило, описанное селектором, расположенным в документе ниже других.)*
  + Ещё более высоким приоритетом обладают стили, объявленные непосредственно в теге данного элемента посредством атрибута style этого тега;
  + И наконец самым высоким приоритетом обладают стили, объявленные автором страницы *или пользователем*, с помощью сопроводительного слова !important. Если таких свойств несколько, то предпочтение отдаётся в первую очередь стилям, *заданным пользователем*, а для остальных свойств (которые будут являться задаваемыми автором страницы) потребуется определить их специфичности по принципам, описанным выше, и применять эти свойства в порядке убывания этих их специфичностей.

### 6.2.7 Пример таблицы стилей

Пример таблицы стилей (в таком виде она может быть либо размещена в отдельном файле .css, либо же обрамлена тегами <style> и размещена в «шапке» той самой веб-страницы, на которую она действует):

p {

font-family: Garamond, serif;

}

h2 {

font-size: 110 %;

color: red;

background: white;

}

.note {

color: red;

background: yellow;

font-weight: bold;

}

p#paragraph1 {

margin: 0;

}

a:hover {

text-decoration: none;

}

#news p {

color: blue;

}

Здесь приведено шесть правил CSS с селекторами p, h2, .note, p#paragraph1, a:hover и #news p.

* Первое правило присвоено HTML-элементу p ([абзацу](http://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B7%D0%B0%D1%86)) — назначен стиль. Абзацы будут отображаться шрифтом [Garamond](http://ru.wikipedia.org/wiki/Garamond), или, если такой шрифт недоступен, каким-либо другим шрифтом с засечками («[serif](http://ru.wikipedia.org/wiki/Serif)»).
* Второе правило присвоено HTML-элементу h2 (заголовку второго уровня). Заголовок второго уровня будет отображаться красным на белом фоне с увеличенным [кеглем](http://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D0%B3%D0%BB%D1%8C).
* Третье правило будет применено к ЛЮБОМУ элементу, атрибут class которого равен 'note'. Например, к параграфу: <p class="note">Этот абзац будет выведен полужирным шрифтом красного цвета на желтом фоне.</p>
* Четвёртое правило будет применяться только к элементам p, атрибут id которых равен paragraph1. Такие элементы не будут иметь внешних отступов (margin).
* Пятое правило определяет стиль hover для элементов a — [гиперссылок](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%81%D1%81%D1%8B%D0%BB%D0%BA%D0%B0). [По умолчанию](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE_%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E), в большинстве браузеров текст элементов a подчеркивается. Это правило уберёт подчеркивание, когда указатель мыши находится над этими элементами.

1. Последнее, шестое правило, применяется для элементов **p**, которые находятся внутри КАКОГО-ЛИБО элемента с атрибутом id, равным «news» (#news p — это типичный случай селектора потомков, см. 5-й пункт списка выше).

## 6.2.8 CSS-вёрстка

До появления CSS оформление веб-страниц осуществлялось исключительно средствами [HTML](http://ru.wikipedia.org/wiki/HTML), непосредственно внутри содержимого документа. Однако с появлением CSS стало возможным принципиальное разделение содержания и представления документа. За счёт этого нововведения стало возможным лёгкое применение единого стиля оформления для массы схожих документов, а также быстрое изменение этого оформления.

Преимущества:

* Несколько дизайнов страницы для разных устройств просмотра. Например, на экране дизайн будет рассчитан на большую ширину, во время печати меню не будет выводиться, а на [КПК](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и сотовом телефоне меню будет следовать за содержимым.
* Уменьшение времени загрузки страниц сайта за счет переноса правил представления данных в отдельный CSS-файл. В этом случае браузер загружает только структуру документа и данные, хранимые на странице, а представление этих данных загружается браузером только один раз и могут быть [закешированы](http://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).
* Простота последующего изменения дизайна. Не нужно править каждую страницу, а лишь изменить CSS-файл.
* Дополнительные возможности оформления. Например, с помощью CSS-вёрстки можно сделать блок текста, который остальной текст будет обтекать (например для меню) или сделать так, чтобы меню было всегда видно при прокрутке страницы.

Недостатки:

* Различное отображение вёрстки в различных браузерах (особенно устаревших), которые по разному интерпретируют одни и те же данные CSS.
* Часто встречающаяся необходимость на практике исправлять не только один CSS-файл, но и теги HTML, которые сложным и ненаглядным способом связаны с селекторами CSS, что иногда сводит на нет простоту применения единых файлов стилей и значительно удлиняет время редактирования и тестирования.

## 6.2.9 История создания и развития CSS

CSS — одна из широкого спектра технологий, одобренных консорциумом [W3C](http://ru.wikipedia.org/wiki/W3C) и получивших общее название «стандарты Web». В 1990-х годах стала ясна необходимость стандартизировать Web, создать какие-то единые правила, по которым программисты и [веб-дизайнеры](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD%D0%B5%D1%80) проектировали бы сайты. Так появились языки [HTML](http://ru.wikipedia.org/wiki/HTML) 4.01 и [XHTML](http://ru.wikipedia.org/wiki/XHTML) и стандарт CSS.

В начале 1990-х различные браузеры имели свои стили для отображения веб страниц. HTML развивался очень быстро и был способен удовлетворить все существовавшие на тот момент потребности по оформлению информации, поэтому CSS не получил тогда широкого признания.

Термин «каскадные таблицы стилей» был предложен [Хокон Виум Ли](http://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D0%BA%D0%BE%D0%BD_%D0%92%D0%B8%D1%83%D0%BC_%D0%9B%D0%B8) в 1994 году. Совместно с Бертом Босом он стал развивать CSS.

В отличие от многих существовавших на тот момент языков стиля, CSS использует наследование от родителя к потомку, поэтому разработчик может определить разные стили, основываясь на уже определенных ранее стилях.

В середине 1990-х Консорциум Всемирной паутины ([W3C](http://ru.wikipedia.org/wiki/W3C)) стал проявлять интерес к CSS, и в декабре 1996 года была издана рекомендация CSS1.

### 6.2.10 Уровень 1 (CSS1)

Рекомендация W3C, принята [17 декабря](http://ru.wikipedia.org/wiki/17_%D0%B4%D0%B5%D0%BA%D0%B0%D0%B1%D1%80%D1%8F) [1996 года](http://ru.wikipedia.org/wiki/1996_%D0%B3%D0%BE%D0%B4), откорректирована [11 января](http://ru.wikipedia.org/wiki/11_%D1%8F%D0%BD%D0%B2%D0%B0%D1%80%D1%8F) [1999 года](http://ru.wikipedia.org/wiki/1999_%D0%B3%D0%BE%D0%B4). Среди возможностей, предоставляемых этой рекомендацией:

* Параметры шрифтов. Возможности по заданию гарнитуры и размера шрифта, а также его стиля — обычного, курсивного или полужирного.
* Цвета. Спецификация позволяет определять цвета текста, фона, рамок и других элементов страницы.
* Атрибуты текста. Возможность задавать межсимвольный интервал, расстояние между словами и высоту строки (то есть межстрочные отступы)
* Выравнивание для текста, изображений, таблиц и других элементов.
* Свойства блоков, такие как высота, ширина, внутренние (padding) и внешние (margin) отступы и рамки. Так же в спецификацию входили ограниченные средства по позиционированию элементов, такие как float и clear.

### 6.2.11 Уровень 2 (CSS2)

Рекомендация W3C, принята [12 мая](http://ru.wikipedia.org/wiki/12_%D0%BC%D0%B0%D1%8F) [1998 года](http://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4). Основана на CSS1 с сохранением обратной совместимости за несколькими исключениями. Добавление к функциональности:

* Блочная вёрстка. Появились относительное, абсолютное и фиксированное позиционирование. Позволяет управлять размещением элементов по странице без [табличной вёрстки](http://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%B2%D1%91%D1%80%D1%81%D1%82%D0%BA%D0%B0).
* Типы носителей. Позволяет устанавливать разные стили для разных носителей (например [монитор](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80_%28%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE%29), [принтер](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%82%D0%B5%D1%80), [КПК](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)).
* Звуковые таблицы стилей. Определяет голос, громкость и т. д. для звуковых носителей (например для слепых посетителей сайта).
* Страничные носители. Позволяет, например, установить разные стили для элементов на чётных и нечётных страницах при печати.
* Расширенный механизм селекторов.
* Указатели.
* Генерируемое содержимое. Позволяет добавлять содержимое, которого нет в исходном документе, до или после нужного элемента.

В настоящее время W3C больше не поддерживает CSS2 и рекомендует использовать CSS2.1

#### Уровень 2, ревизия 1 (CSS2.1)

Рекомендация W3C, принята [7 июня](http://ru.wikipedia.org/wiki/7_%D0%B8%D1%8E%D0%BD%D1%8F) [2011 года](http://ru.wikipedia.org/wiki/2011_%D0%B3%D0%BE%D0%B4).

CSS2.1 основана на CSS2. Кроме исправления ошибок, в новой ревизии изменены некоторые части спецификации, а некоторые и вовсе удалены. Удаленные части могут в будущем быть добавлены в CSS3.

### 6.2.12 Уровень 3 (CSS3)

Разрабатываемая версия.

Сильно расширена по сравнению с предыдущими версиями. Нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией ([анимацией](http://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%28%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%29)) и, возможно, введением [переменных](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F).

## 6.2.13 Поддержка CSS браузерами

Наиболее полно поддерживающими стандарт CSS являются браузеры, работающие на [движках](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA) [Gecko](http://ru.wikipedia.org/wiki/Gecko) ([Mozilla Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox) и др.), [WebKit](http://ru.wikipedia.org/wiki/WebKit) ([Safari](http://ru.wikipedia.org/wiki/Safari), [Arora](http://ru.wikipedia.org/wiki/Arora), [Google Chrome](http://ru.wikipedia.org/wiki/Google_Chrome)) и [Presto (Opera)](http://ru.wikipedia.org/wiki/Presto_%28Opera%29).

Бывший когда-то самым распространённым браузером [Internet Explorer](http://ru.wikipedia.org/wiki/Internet_Explorer) 6 поддерживает CSS далеко не полностью.

Вышедший спустя 7 лет после своего предшественника Internet Explorer 7 хотя и значительно улучшил уровень поддержки CSS, но всё ещё содержит значительное количество ошибок.

В Internet Explorer 8 используется новый движок, который полностью поддерживает CSS 2.1 и частично — CSS 3.

Для проверки поддержки браузером веб-стандартов (в том числе и различных частей стандарта CSS) был разработан тест [Acid](http://ru.wikipedia.org/wiki/Acid1). Его вторая версия называется [Acid2](http://ru.wikipedia.org/wiki/Acid2), а третья, соответственно, [Acid3](http://ru.wikipedia.org/wiki/Acid3).

### 6.2.14 Различные блоковые модели

В стандартах CSS от Консорциума [W3C](http://ru.wikipedia.org/wiki/W3C) используется модель, в которой свойство width определяет ширину содержимого блока, не включая в нее отступы и рамки. Ранние версии [Internet Explorer](http://ru.wikipedia.org/wiki/Internet_Explorer) (4 и 5), реализовали собственную модель, в которой width определяет расстояние между рамками блока, включая отступы (padding) и рамки (border). Кроме Internet Explorer 5 эту модель так же понимают браузеры [Netscape](http://ru.wikipedia.org/wiki/Netscape_Navigator) 4 и [Opera](http://ru.wikipedia.org/wiki/Opera) 7. Поддержка стандартной модели W3C появилась в IE только в шестой версии.

В разрабатываемом стандарте CSS3 для решения этой проблемы введено свойство box-sizing, со значениями content-box для указания на использование стандартной модели W3C и border-box для использования модели IE 5.

В браузере [Mozilla](http://ru.wikipedia.org/wiki/Mozilla), при поддержке этого свойства, под собственным «рабочим» названием -moz-box-sizing, ввели еще одно значение — padding-box, таким образом создав третью блочную модель, в которой width это размер содержимого и отступов блока, не включая рамки.

### 6.2.15 CSS-фильтры

Различия в реализации CSS различными браузерами заставляют веб-разработчиков искать решения, как заставить все браузеры отображать страницу одинаково. CSS-фильтры (также часто называемые CSS-хаками) позволяют выборочно применять (или не применять) стили к различным элементам. Например, известно, что [Internet Explorer](http://ru.wikipedia.org/wiki/Internet_Explorer) 6 применяет правила, использующие селекторы вида \* html селектор (фильтр, известный как «[star html bug](http://ru.wikipedia.org/w/index.php?title=Star_html_bug&action=edit&redlink=1)»). Тогда, чтобы заставить и браузеры, использующие блоковую модель W3C и IE, работающего в [Quirks mode](http://ru.wikipedia.org/wiki/Quirks_mode) со своей блоковой моделью, отображать блок #someblock шириной в 100 пикселей и внутренними отступами в 10 пикселей можно написать такой код:

/\* Модель W3C - 80px ширина содержимого и 10px отступы с каждой стороны \*/

#someblock { width: 80px; padding: 10px; }

/\* Следующее правило применит только IE6. \*/

\* html #someblock { width: 100px; padding: 10px; }

Ещё одним способом выборочного применения правил для Internet Explorer являются [условные комментарии](http://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BA%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8).

### 6.2.16 Безопасность

Все поддерживаемые версии Internet Explorer, на конец 2010 года были уязвимы: при обработке браузером каскадных таблиц стилей (CSS) может возникнуть неинициализированная память, используемая затем для удаленного запуска на компьютере пользователя вредоносного кода.

## 6.2.17 CSS Framework

**CSS Framework** (также **Web design framework**) — это заранее подготовленная css-библиотека, созданная для упрощения работы верстальщика, быстроты разработки и исключения максимально возможного числа ошибок вёрстки (проблемы совместимости различных версий браузеров и т. д.). Так же как и библиотеки скриптовых языков программирования, CSS-фреймвёрки, обычно имеющие вид внешнего .css-файла, «подключаются» к проекту (добавляются в заголовок веб-страницы), позволяя не искушенному в тонкостях вёрстки программисту или дизайнеру правильно создать xhtml-макет.

## 6.2.18 Расширения CSS

Часто при вёрстке страниц нужно использовать одно и то же значение много раз: один и тот же цвет, один и тот же шрифт. И если это значение нужно будет изменить, то придётся менять во многих местах. В стандартном CSS нет возможностей наследования стилей, вычисляемых значений и прочих зависимостей.

Для решения этих вопросов и ускорения разработки существует несколько расширений языка CSS. Расширений в том смысле, что код CSS является валидным кодом для расширения, но не наоборот. Чтобы из кода "расширенного CSS" получился обычный CSS-файл, воспринимаемый браузером, необходимо выполнить компиляцию. Компиляция может быть нескольких типов:

* во время запуска страницы на стороне клиента (средствами JavaScript)
* во время запуска страницы на стороне сервера
* во время вёрстки сайта средствами специального компилятора

Примеры расширений CSS:

* [Sass](http://ru.wikipedia.org/wiki/Sass)
* [LESS](http://lesscss.org/)

# Создание программного комплекса, используя возможности выбранных языков программирования

Ранее были рассмотрены основные задачи дипломной работы:

Основные:

* Реализация поиска аналогов для сравнительного подхода
* Реализация автоматического составления таблиц для доходного и затратного подходов

Второстепенные:

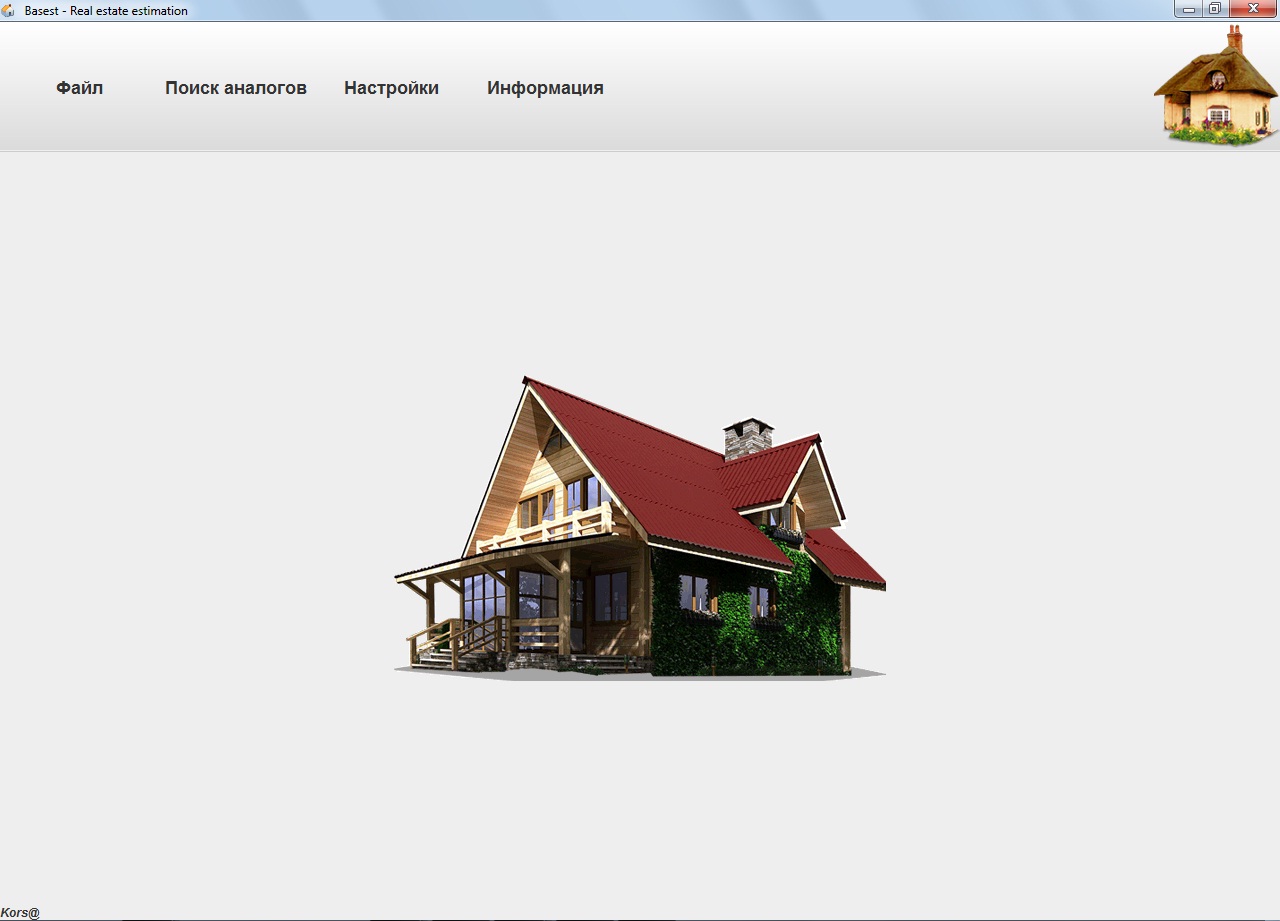
* Предоставление информации необходимой в процессе оценки
* Введение дополнительных функций программы для удобства ее использования

Также мы уже рассмотрели язык программирования на котором будет построен наш программный комплекс.

## 7.1 Реализация интерфейса пользовательского приложения

Для реализации пользовательского интерфейса была использована библиотека Swing.

Swing — [библиотека](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) для создания [графического интерфейса](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) на языке [Java](http://ru.wikipedia.org/wiki/Java). Swing был разработан компанией [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems). Он содержит ряд графических [компонентов](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Swing widgets), таких как кнопки, поля ввода, таблицы и т. д. Swing относится к библиотеке классов [JFC](http://ru.wikipedia.org/wiki/Java_Foundation_Classes), которая представляет собой набор библиотек для разработки графических оболочек. К этим библиотекам относятся [Java 2D](http://ru.wikipedia.org/w/index.php?title=Java_2D&action=edit&redlink=1), [Accessibility-API](http://ru.wikipedia.org/w/index.php?title=Accessibility-API&action=edit&redlink=1), [Drag & Drop](http://ru.wikipedia.org/wiki/Drag_%26_Drop)-API и [AWT](http://ru.wikipedia.org/wiki/Abstract_Window_Toolkit).

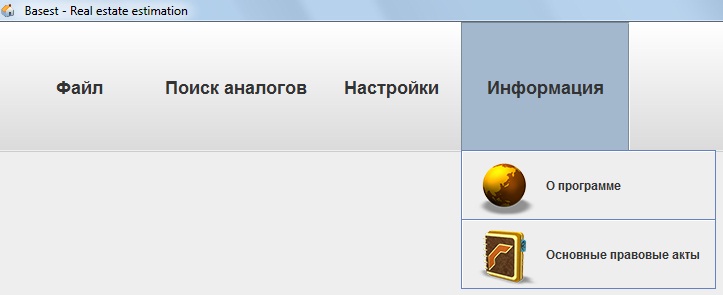


#### План интерфейса:

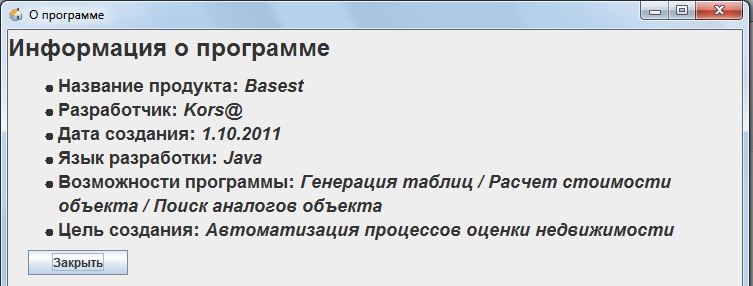
## 7.2 Реализация информационного блока программы:

Составляющие информационного блока:

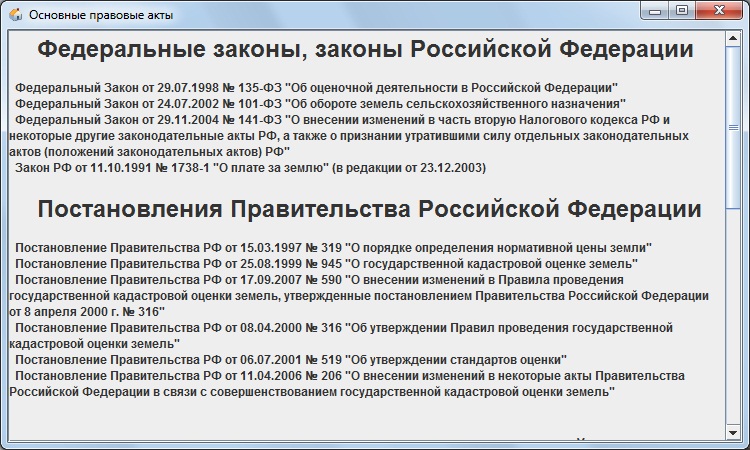
* Информация о программе
* Информация для оценки



- Информация о программе:



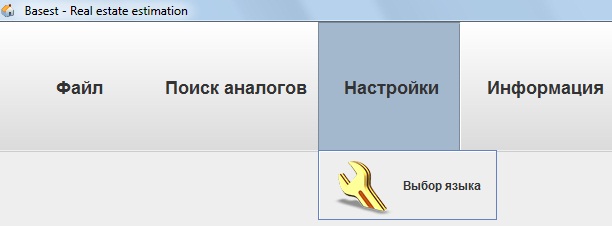
- Информация для оценки



## 7.3 Реализация меню «Настройки»

Составляющие меню «Настройки»

* Выбор языка



#### При написании данной возможности программы использовались возможности Java. Для чтения из текстового файла, и передачи названий кнопок на разных языках, использовались возможности библиотек java.io.BufferedReader и java.io.FileReader.

#### D:\Треклятый универ\язык.jpg

Примечание:

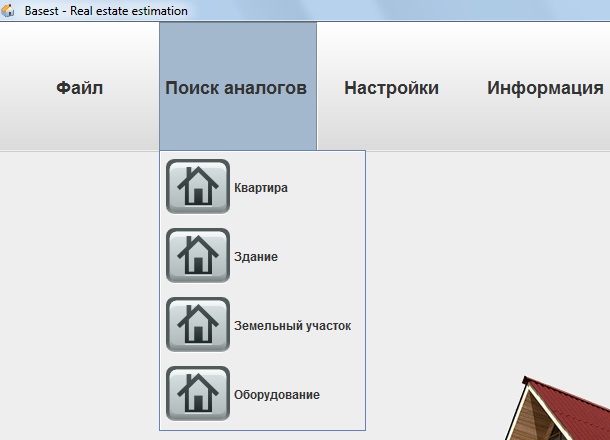
Дополнительно реализован запуск выбора языка

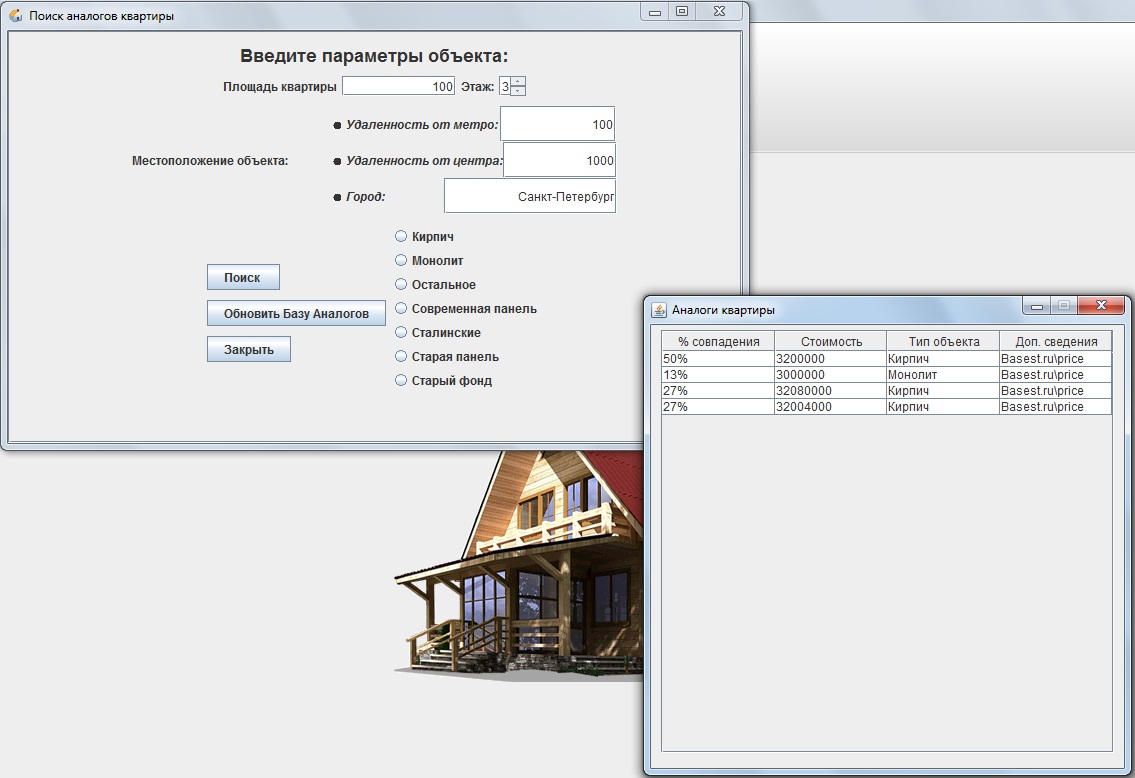
при первом открытии программы

## 7.4 Реализация меню «аналоги»

Составляющие меню «аналоги»

* Квартира
* Здание
* Земельный участок
* Оборудование

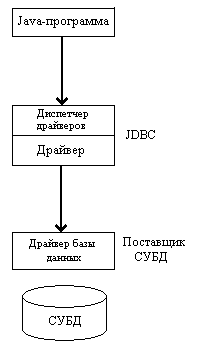




#### Реализация связи пользовательского интерфейса с данными БД

Изначально планировалась связь программы с БД напрямую при помощи библиотеки JDBC, но при детальном рассмотрении принципа работы этой библиотеки был выбран более быстро действенный способ реализации, используя возможности java.io.BufferedReader и java.io.FileReader.

**Принцип работы JDBC**



Другими словами, для использования данных хранимых в используемой БД надо их все загрузить, присвоить все разным переменным и только затем есть возможность их дальнейшей программной обработки.

В случае с чтением из текстового файла, количество переменных существенно сокращается, за счет выборки только необходимых параметров в процессе чтения.

При получении списков аналогов квартиры мы получаем четыре поля с параметрами аналогов:

- % совпадения

- Стоимость

- Тип объекта

- Доп. Сведения

% совпадения - это процентное совпадение объекта аналога, которое рассчитывается путем процентного сравнения с данными аналогов в БД.

Стоимость, тип и доп. сведения – представляют собой данные взятые из БД.

Стоимость – стоимость объекта аналога

Тип объекта - тип здания или постройки объекта аналога

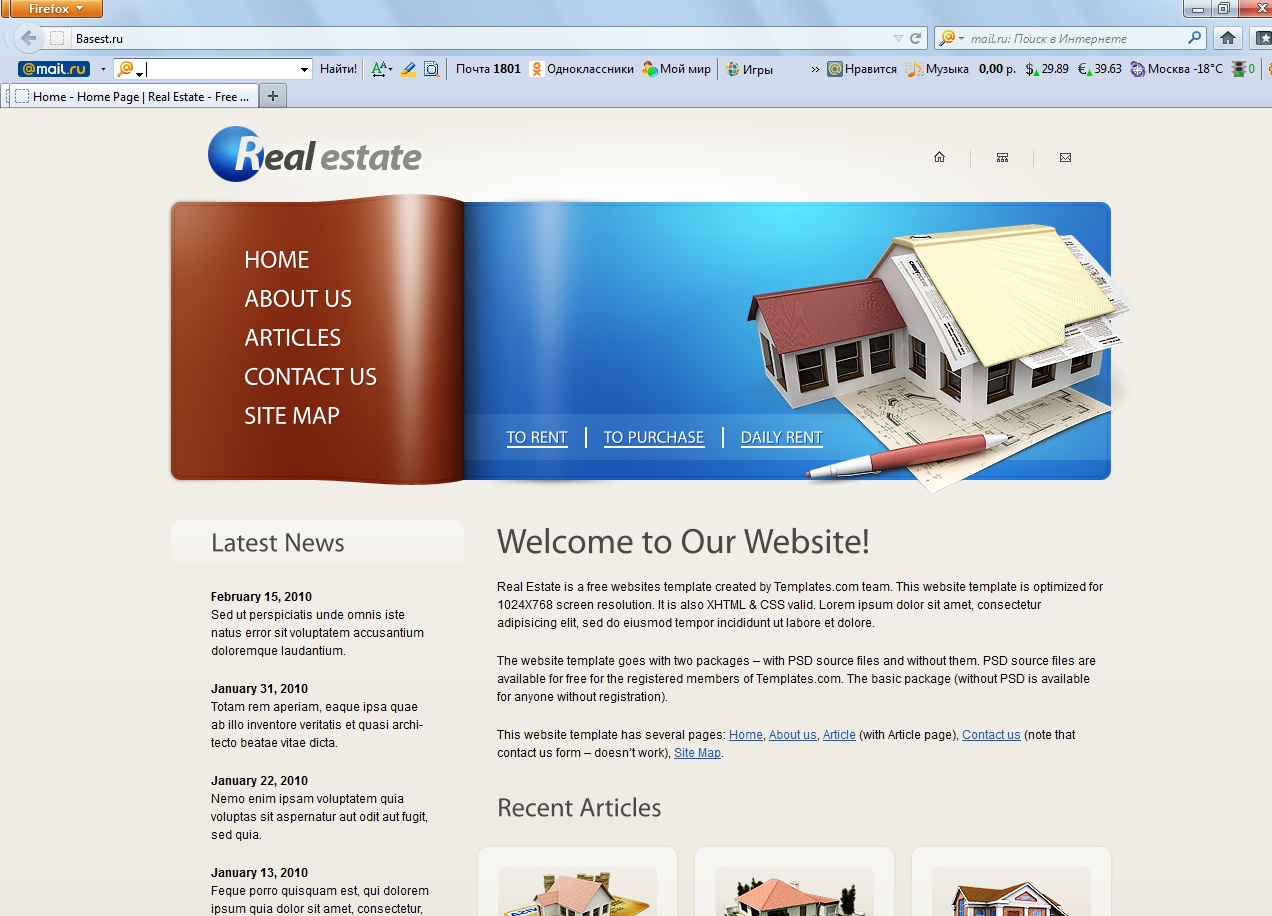
Доп. сведения – ссылка на источник данных

#### Создание сайта

Для того чтобы пользоваться данными из БД, необходимо организовать доставку этих данных пользователю. Для решения данной проблемы был создан сайт, для быстрой передачи файлов пользователю. Требования к сайту не подразумевают особых технологических разработок и эксклюзивного контента, потому можно использовать для его создания только html и css.

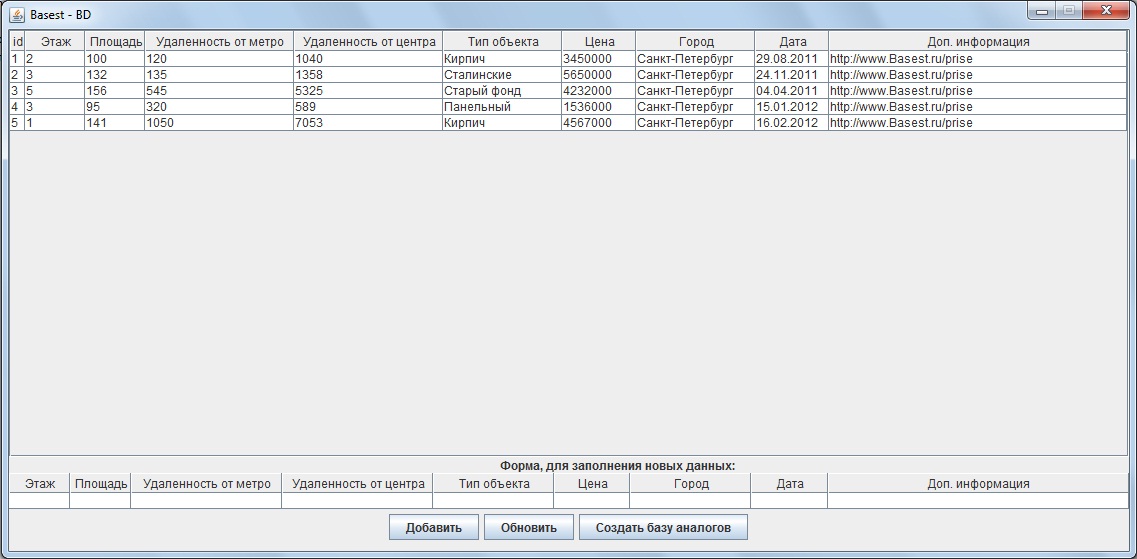
HTML (от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HyperText Markup Language — «язык разметки [гипертекста](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82)») — стандартный [язык разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](http://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) создаются при помощи языка HTML (или [XHTML](http://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) и отображается в виде документа, в удобной для человека форме.

CSS ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Cascading Style Sheets — каскадные таблицы стилей) — [формальный язык](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) описания внешнего вида документа, написанного с использованием [языка разметки](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8).

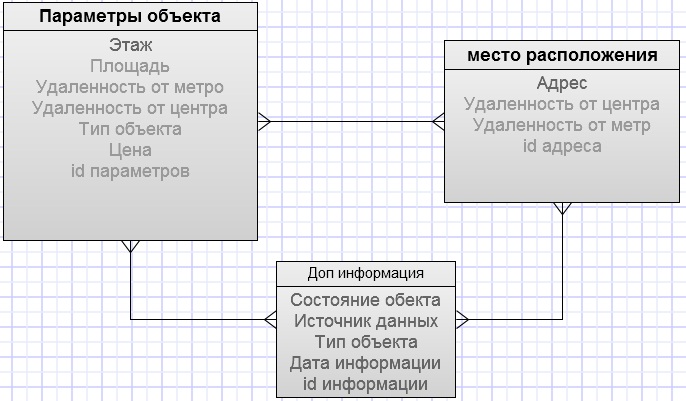


#### Реализация программы для заполнения БД информацией

Также для удобства нам понадобится мини-приложение для быстрого заполнения БД информацией.



ER- диаграмма

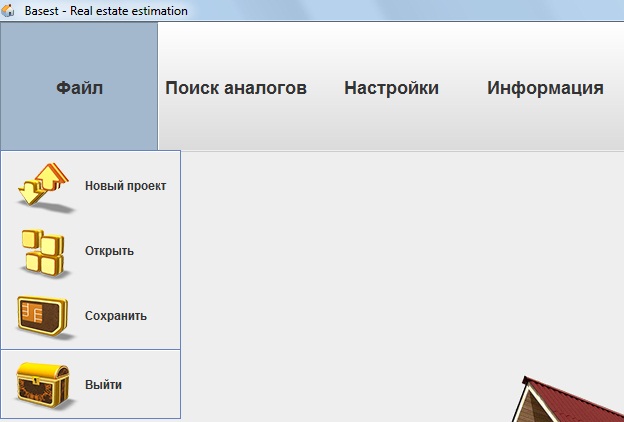


Принцип функционирования данных от исходной БД пользователю

## 7.5 Реализация меню «Фаил»

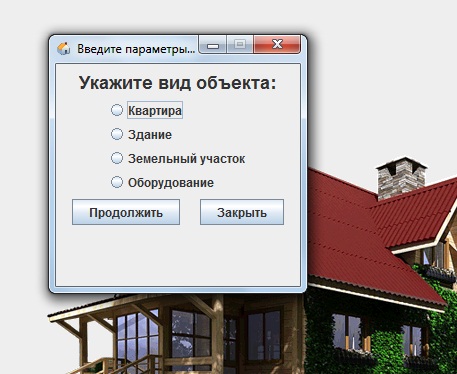
Составляющие меню «Фаил» :

* Новый проект
* Открыть
* Сохранить
* Выйти

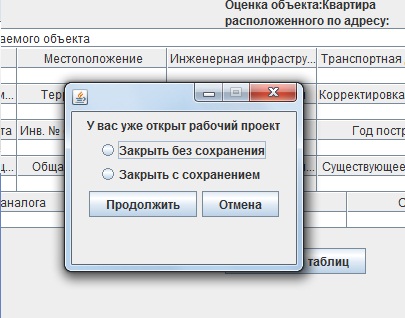


#### Реализация «Нового проекта»

Если нет открытого проекта:

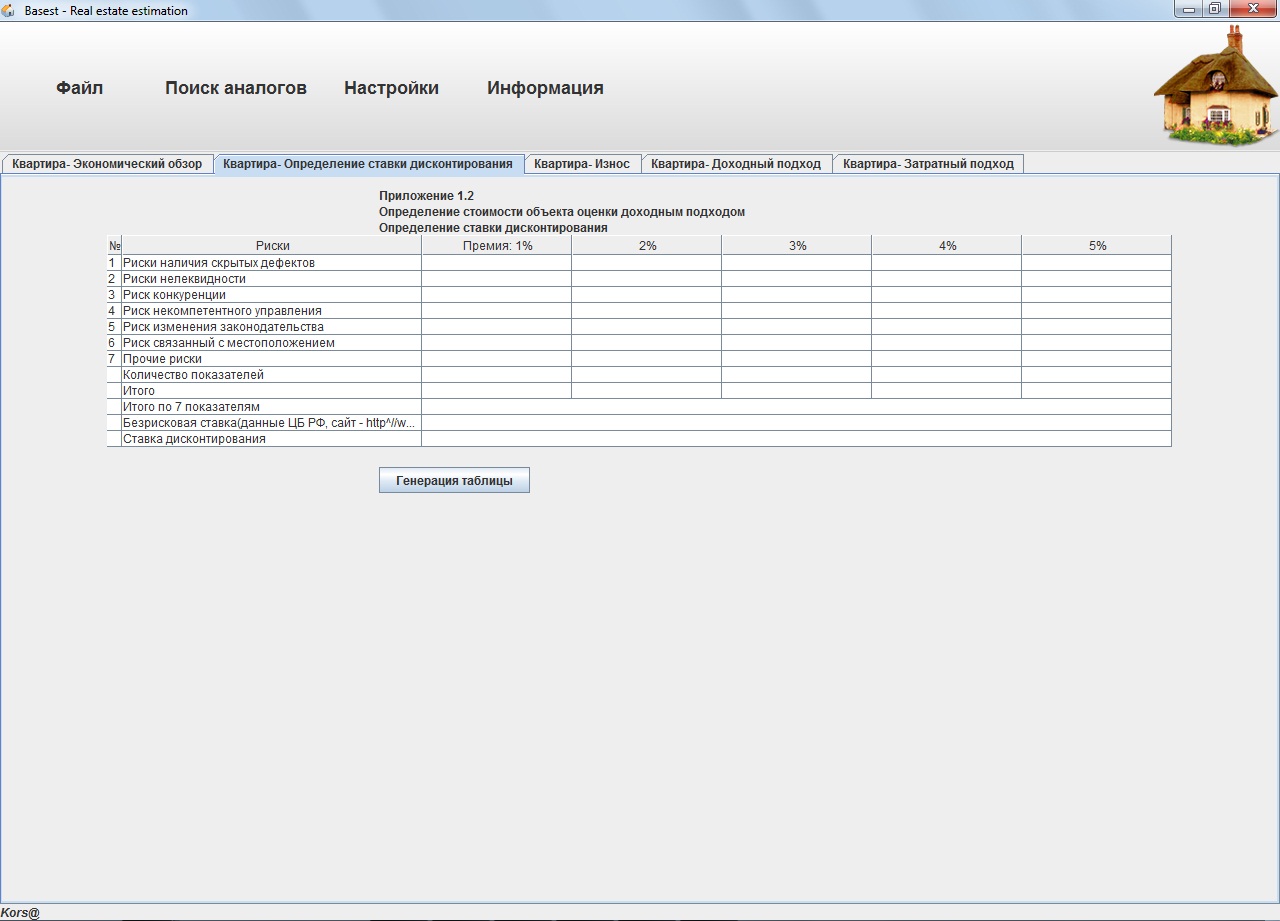


Если есть открытый проект:



При создании данного пункта меню был написан алгоритм, способный распознавать открыт ли уже какой-либо проект, и функционировать в зависимости от этого фактора

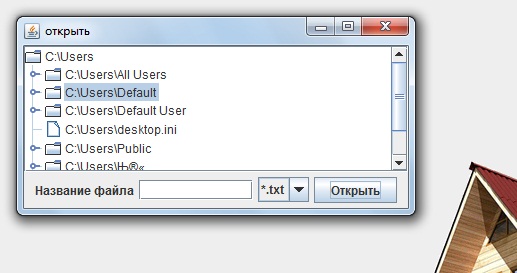
«Новый проект»



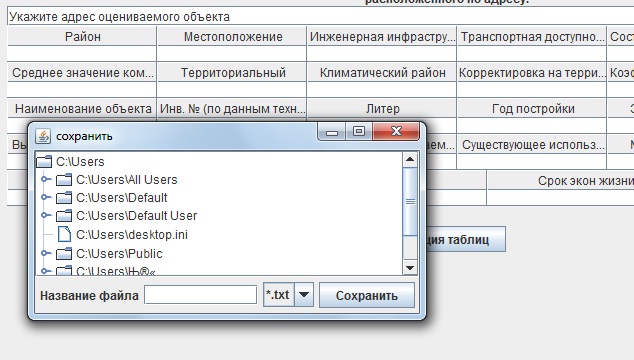
#### Реализация пунктов «Сохранить» и «Открыть»

При реализации данных пунктов меню была использована библиотека JTree, для создания дерева файлов, что необходимо при сохранении и открытия проектов.

Открыть

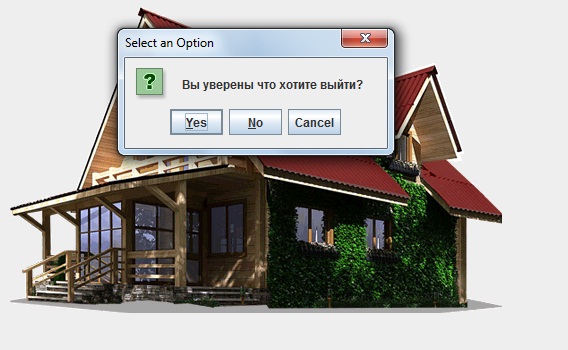


Сохранить



#### Реализация пункта «Выход»

«Выход»



При реализации пункта меню «Выход» мы пользовались возможностями Java «Окно вопроса», что позволило не прописывать вручную все нюансы этого пункта. В этом окне несколько кнопок, одну из которых пользователь должен нажать. В программу, вызывающую это диалоговое окно, возвращается информация о выборе пользователя, на основе которой и строится дальнейший ход работы программы.

# Технико-экономическое обоснование проекта

## Концепция экономического обоснования научно-технической разработки

В настоящее время, особенно в крупных мегаполисах, постоянно совершаются операции над недвижимостью, но чтобы продать, обменять, сдать в аренду и даже разрешить спор в судебном порядке, необходимо конкретно выяснить стоимость объекта. На оценку влияет множество факторов, начиная от простых размеров объекта, удаленности от центра и заканчивая экономической ситуацией в целом. Работа оценщика подразумевает огромное количество вычислительной работы, поиск аналогов объекта и т.д..

Целью создания дипломного проекта является программа, предназначенная для определения рыночной стоимости объектов недвижимости и земельных участков с последующим составлением отчета. Использование современных технологий, Java, SQL обеспечит быстрое и надежное функционирование объекта, а главное обеспечит легкость интегрирования в различные системы, за счет гибкой системы настройки под критерии пользователя, доступности и распространенности.

В рамках настоящего дипломного проекта разработана программа для автоматизации основных аспектов оценки недвижимости. Данная программа обеспечивает удобное управление расчетами, в том числе калькуляциями.

Описание работ по разработке программы автоматизации оценки недвижимости:

* Формулирование целей и требований к проекту.   
  На данном этапе идет анализ тех частей работы оценщика, которые подразумевают монотонную работу, расчеты и т.д.
* Введение в предметную область.

На данном этапе идет изучение предметной области, ознакомление с материалами (документами, экономическими расчетами), при необходимости консультации специалиста данной предметной области.

* Изучение литературы.

На данном этапе идет изучение функциональных возможностей различного рода программных сред, их взаимодействие с различного вида системами.

* Анализ требований к программному комплексу.

На данном этапе производится анализ основных, функциональных, системных и других требований к программному комплексу.

* Разработка концептуальных моделей.

На данном этапе идет разработка архитектуры системы, подсистем, а так же принципы их взаимодействия.

* Разработка графического интерфейса.

На данном этапе идет разработка графического интерфейса.

* Тестирование программного продукта.

На данном этапе проводится тестирование работы отдельных модулей и системы в целом.

* Оформление проекта, разработка документации, отладка

На данном этапе идет оформление проекта, разработка технической документации, руководства пользователя и отладка.

Для проведения экономического обоснования работы рассчитывается себестоимость научно-технической разработки на основании трудоемкости работ и сметы затрат по соответствующим статьям калькуляции. Этапы проведения экономического обоснования проекта:

* определение потребительских свойств продукта, связанных с совершенствованием продукта;
* анализ рынка сбыта проектируемой продукции и разработка плана маркетинга;
* описание процесса производства продукции и оценка ресурсов, необходимых для усовершенствования разрабатываемого продукта;
* описание конкретных работ и организационно-технических мероприятий, направленных на практическую реализацию проекта;
* оценка интегрального экономического эффекта, получаемого от модернизации программного продукта.

## Потребительские свойства продукта, связанного с совершенствованием продукта

Таблица 3.1

Технико-экономические параметры проектируемой продукции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование технико-эксплуатационного  параметра продукции | Ед.  Мерения | Проектируе-мая  продукция | Наименование (марка) существующей  Продукции | |
| 1 | 2 |
| Занимаемый объем памяти на HD | мб | 2 | 1 | 1 |
| Необходимость в дополнительном ПО | шт. | 0 | 1 | 2 |
| Скорость составления отчета | сек. | 1 | 2 | - |
| Потребность в дополнительных комплектующих компьютера | шт. | 1 | 0 | 0 |
| Удобство использования |  | Лучше | Хуже | Хуже |
| Понятность интерфейса |  | Лучше | Хуже | Хуже |
| Необходимость обновления |  | Да | Да | Да |
| Необходимость установки |  | Нет | Да | Да |

Таблица 8.2

Бальное сравнение новой системы и существующих программ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Показатель** | **Балльная оценка** | | **Коэффициент значимости, µi** |
| **Старая система, Ббаз** | **Новая система, Бi** |
| 1 | Обновление баз данных | 3 | 10 | 0,3 |
| 2 | Удобство эксплуатации | 7 | 9 | 0,2 |
| 3 | Обновление версии программы | 2 | 8 | 0,1 |
| 4 | Возможность построения отчетов | 8 | 8 | 0,2 |
| 5 | Возможность контроля | 8 | 9 | 0,2 |

Уровень качества определяется по формуле: **Ккач=∑ µi\*(Бi/ Ббаз)** = 0,3\*10/3 + 0,2\*9/7 + 0,1\*8/2 + 0,2\*8/8 + 0,2\*9/8 = 2.1

При вышеперечисленных экспертных оценках коэффициент качества составляет 2,1. Т.о. уровень качества новой системы превосходит уровень старой более чем в два раза.

## Рынок и план маркетинга

#### 8.3.1 Сегментирование рынка

Сегментирование рынка состоит в выделении сегментов анализируемого рынка и оценки спроса на продукцию в каждом сегменте рынка. Процедура сегментирования включает:

* Описание рынка, оценка его размеров (объемов продаж), стадию развития (зарождающийся, растущий, зрелый или падающий), краткая характеристика продукции, реализуемой на данном рынке с указанием периода жизненного цикла продукции на рынке;

Основными потребителями данной продукции являются оценочные компании. На данной стадии развития рынок потребления является растущим, а значит в потребности данной программы будут нуждаться все большее количество потребителей. Так как программа обладает функцией самообновления, то жизненный цикл данной программы будет таким же, сколько существуют оценочные компании.

* Анализ требований различных групп потенциальных покупателей к продукции, реализуемой на рынке. Среди общих требований, предъявляемых покупателями к продукции (новизна, технический уровень, качество и надежность в эксплуатации, уровень послепродажного обслуживания, цена продукции и др.), следует выделить те, которые являются наиболее существенными для определенных групп потребителей;

Помимо оценочных компаний потребителями данной программы могут быть студенты вузов, занимающихся подготовкой специалистов по данной специальности. Для данных двух групп потребителей необходимы такие критерии как технический уровень, надежность в эксплуатации, цена покупки и эксплуатации, уровень послепродажного обслуживания.

* Группировка потенциальных покупателей в рыночные сегменты, внутри которых имеет место схожесть требований, предъявляемых потенциальными покупателями к продукции и готовых ее покупать;

При разделении на рыночные сегменты можно выделить такие как:

- Оценочные компании

- Студенты экономических направлений

Внутри сегментов рынка можно выделить такие схожие признаки требований как:

- технический уровень

- надежность в эксплуатации

- цена покупки

* Оценка спроса на продукцию в каждом сегменте рынка с учетом численности потенциальных покупателей, частоты производимых покупок, а также тенденций изменения спроса.

Разрабатываемый продукт полностью удовлетворяет потребностям целевых сегментов рынка. Но среди сегмента рынка направленного на студенческую аудиторию будет наблюдаться тенденция на приобретение продукта для ознакомления, без приобретения лицензии на дальнейшее обслуживание продукта.

#### 8.3.2 Выбор целевого сегмента рынка

Выбор целевого сегмента рынка производится на основе анализа конкурентных позиций новой продукции в различных сегментах рынка, оценки благоприятных возможностей и угроз продвижения этой продукции в каждом сегменте рынка. Для выбора целевого сегмента следует произвести следующее:

А) Анализ конкурентных позиций разрабатываемой продукции в сравнении с продукцией, реализуемой в каждом сегменте рынка, с указанием улучшенных параметров или недостатков разрабатываемой продукции, а также влияния этих параметров на потребительский выбор;

Таблица 8.3(1)

Для оценочных компаний:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Показатель | Балльная оценка | | Коэффициент значимости, µi |
| Старая система, Ббаз | Новая система, Бi |
| 1 | Обновление баз данных | 3 | 10 | 0,3 |
| 2 | Удобство эксплуатации | 7 | 9 | 0,2 |
| 3 | Обновление версии программы | 2 | 8 | 0,1 |
| 4 | Возможность построения отчетов | 8 | 8 | 0,2 |
| 5 | Возможность контроля | 8 | 9 | 0,2 |

Уровень качества определяется по формуле:

Ккач=∑ µi\*(Бi/ Ббаз) = 0,3\*10/3 + 0,2\*9/7 + 0,1\*8/2 + 0,2\*8/8 + 0,2\*9/8 = 2.1

Уровень качества новой системы превосходит уровень старой более чем в два раза

Таблица 3.3(2)

Для студенческой аудитории:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Показатель | Балльная оценка | | Коэффициент значимости, µi |
| Старая система, Ббаз | Новая система, Бi |
| 1 | Обновление баз данных | 3 | 10 | 0,1 |
| 2 | Удобство эксплуатации | 7 | 9 | 0,3 |
| 3 | Обновление версии программы | 2 | 8 | 0,1 |
| 4 | Возможность построения отчетов | 8 | 8 | 0,3 |
| 5 | Возможность контроля | 8 | 9 | 0,2 |

Уровень качества определяется по формуле:

Ккач=∑ µi\*(Бi/ Ббаз) = 0,1\*10/3 + 0,3\*9/7 + 0,1\*8/2 + 0,3\*8/8 + 0,2\*9/8 = 1,7

Уровень качества новой системы превосходит уровень старой почти в два раза

Уровень качества новой системы в обоих сегментах целевого рынка превосходит возможности старой системы и может занять высокую конкурентную позицию на рынке.

Б) Анализ потенциальных барьеров при входе на конкретный сегмент рынка, обратив внимание на возможности и способы преодоления этих барьеров;

Основным барьером при входе на рынок сбыта продукции является узнаваемость новой системы для целевой аудитории. Для решения данной проблемы возможны несколько решений:

- Организация бесплатной ознакомительной версии продукта

- При приобретении продукта бесплатное пользование обновляемыми базами данных в течении некоторого времени.

- Размещение рекламы на ресурсах наиболее посещаемых потребителями данной продукции.

С) Выбор наиболее привлекательного сегмента рынка (или несколькольких сегментов) с учетом таких факторов, как привлекательность сегмента с точки зрения его прибыльности, стабильности, с одной стороны, а с другой - высоких конкурентных позиций, которые может занять новая продукция на рынке.

Наиболее привлекательный сегмент рынка, для сбыта продукции являются оценочные компании. На данном сегменте рынка продукт может занять лидирующие конкурентные позиции и приносить стабильный доход.

## Производство продукта

Таблица 8.4

Технологический процесс производства продукции

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование технологической операции | Норма времени, час/шт | Технологическое оборудование | | | Производственные площади, кв.м. |
| Наименование (тип, марка) | Стоимость, руб. | Способ приобретения |  |
| Разработка | 150 | Pentiun Core 2 Duo 2шт. | 60 000 | Покупка | 12 |
|
|
|
| Подключение к интернету 15 Мбит/с, 5 мес. | 2250 | Аренда |
| Хостинг, 5 мес. | 1000 |
| Представление документации | 20 | Samsung SCX-3200 | 6000 | Покупка | 12 |

Число рабочих мест на каждой операции () рассчитывается с учетом прогнозируемого объема производства:

, 

где  - норма времени выполнения операции *i*, час/шт.;  - максимальный объем производства в планируемом периоде (шт). В качестве планируемого периода принимается неделя; *F* – фонд времени работы рабочего места, час.

Затраты на содержание и эксплуатацию оборудования определяются из расчета на 1 час работы оборудования с учетом стоимости и производительности оборудования:

,

где  - расчетная себестоимость одного машино-часа работы оборудования на *i*-й технологической операции (для 1 ПК и принтера стоимость электроэнергии на час работы усредним до 1р.), р/м-ч;  - количество машино-часов, затрачиваемых на выполнение *i*-й технологической операции, м-ч (возьмем верхнюю границу времени 5 месяцев по 8 часов).



• Трудоемкость выполнения работ Tи в месяц = 150 чел.-дн.,

• Дневная ставка инженера Ди = 900 руб.

• Процент дополнительной заработной платы 12%.

• Страховые взносы (замена ЕСН с 2010 года, в т.ч.: ПФР – 26%, ФСС – 2,9%, ФФОМС – 2,1%, ТФОМС – 3%) = 34%.

• Прочие прямые расходы 5%.

• Процент накладных расходов НР = 20%.

В статье “Основная заработная плата” показывается сумма заработной платы сотрудников, непосредственно занятых разработкой проекта, а также заработная плата работников нештатного состава, привлеченных к ее выполнению. Основная заработная плата исполнителей рассчитывается по формуле:

Сзо = Tи\* Ди = 135000 руб.

Статья “Дополнительная заработная плата” включает выплаты, которые в соответствии с законодательством о труде или коллективным договором, осуществляется за неотработанное время. К таким выплатам относятся оплата отпусков, время выполнения общественных обязанностей и другие. Дополнительная заработная плата рассчитывается по формуле:

Сдз = Сзо\*0,12 = 16200 руб.

Статья “Страховые взносы” включает сумму отчислений от основной и дополнительной заработной платы, т.е. отчисления в фонды социального страхования, медицинского страхования, фонд занятости и пенсионный фонд. Отчисления осуществляются по законодательно установленной норме. Отчисления на социальные нужды рассчитываются по формуле:

Ссн = (Сзо+ Сдз)\*0,34 = 51408 руб.

Статья “Прочие прямые расходы” включает расходы на получение специальной научно-технической информации, платежи за использование средств связи, коммуникации и др. Прочие прямые расходы рассчитываются по формуле:

Сп.пр = (См+Сзо+ Сдз+ Ссн+ Спл)\*0,05 = 10237 руб.

Статья “Накладные расходы” включает расходы на управление и хозяйственное обслуживание, которые в равной степени относятся ко всем выполняемым работам. Накладные расходы рассчитываются по формуле:

Снр = (Сзо+ Сдз)\*НР = 30240 руб.

На основании полученных данных по отдельным статьям затрат составляется калькуляция себестоимости разработки в целом (таблица 8.5).

Таблица 8.5

|  |  |  |
| --- | --- | --- |
| **№** | **Статья затрат** | **Сумма, руб.** |
| 1 | Материалы | *72 020* |
| 2 | Специальное оборудование для научных и экспериментальных работ | *0* |
| 3 | Аренда помещения | *10 000* |
| 4 | Основная заработная плата | *135 000* |
| 5 | Дополнительная заработная плата | *16 200* |
| 6 | Отчисления на социальные нужды | *51 408* |
| 7 | Прочие прямые расходы | *10 237* |
| 8 | Накладные расходы | *30 240* |
| 9 | Расходы на электроэнергию | *3600* |
| Итого себестоимость | | *328 705* |

Вывод: данные, представленные в таблице 8.5, показывают, что итоговая себестоимость разработки равна *328 705 руб*.

## 8.5 Организационный план проекта

Таблица 8.6

Организационный план работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование работ | Ответственный исполнитель | Ожидаемые сроки выполнения | Потребность в ресурсах |
| п/п | (мероприятий) |
| 1 | Формулирование целей и требований к проекту |  | 1 |  |
| 2 | Введение в предметную область |  | 15 |  |
| 3 | Изучение литературы |  | 20 |  |
| 4 | Изучение концептуальной, логической и физической моделей сущестующих систем |  | 5 |  |
| 5 |
| 6 | Анализ требований к программному комплексу |  | 5 |  |
| 7 | Разработка технического задания |  | 1 |  |
| 8 | Разработка концептуальных моделей |  | 15 |  |
| 9 | Разработка системы связей с внешними источниками |  | 30 |  |
| 10 | Разработка графического интерфейса |  | 10 |  |
| 11 | Тестирование программного продукта |  | 2 |  |
| 12 | Разработка документации |  | 5 |  |
|  | Всего |  | 110 |  |

## Прогноз финансовых показателей проекта

Капитальные вложения в научно-техническую продукцию определяются прямым счетом на основе планируемых (фактических) затрат на их создание. Величина этих затрат принимается равной себестоимости разработки.

Капитальные вложения в основные средства определяются единовременными затратами на приобретение дополнительных производственных и административных помещений, технологического оборудования, транспортных средств и иных основных средств, необходимых для организации производства и сбыта новой продукции.

Капитальные вложения в производственные и административные помещения определяются по формуле

,

где *Ч* – число дополнительных рабочих мест, создаваемых для организации производства новой продукции, определяемое как численность основных, вспомогательных рабочих, специалистов и административно-управленческого персонала, задействованного в проекте; *П* – усредненная площадь одного рабочего места с учетом потребности в производственных площадях, занимаемых оборудованием, кв.м; *Ц* – рыночная стоимость одного квадратного метра производственных (офисных) помещений, р/кв.м..

Согласно требованиям санитарных норм рекомендуемая площадь одного рабочего места составляет от 4 до 6 квадратных метров. Возьмем 5 кв.м.



Капитальные вложения в технологическое оборудование, транспортные средства, передаточные устройства определяются по формуле

,

где  - количество единиц оборудования *i*-го вида, необходимых для организации производства новой продукции в планируемых объемах, шт.;  - цена единицы оборудования *i*-го вида, тыс. р; *m* – количество видов оборудования, транспортных средств, передаточных устройств и т.п.; *H* – доля затрат на транспортировку, монтаж и наладку от стоимости оборудования, передаточных устройств примем равной 4%.



Основные результаты расчетов капитальных вложений во внеоборотные активы сводятся в табл. 8.7, в которой указываются также нормативные сроки использования каждого из активов.

Таблица 8.7

Капитальные вложения во внеоборотные активы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п  п | Наименование актива | Затраты на приобретение, тыс.р. | Нормативный срок использования, лет | Годовая амортизация, тыс.р. |
| 1. | Нематериальные активы | 247 | 40 | 6,125 |
| в том числе: |
| 1.1. | Научно-техническая продукция | 247 | 40 | 6,125 |
| 2. | Основные средства | 82 | - | 7,125 |
| в том числе: |
| 2.1. | Производственные и служебные помещения | 10 | 40 | 0,25 |
| 2.2. | Оборудование | 72 | 10 | 7,2 |
| ВСЕГО | | 329 | - | 13,375 |

При расчете величины текущих производственно-сбытовых издержек, связанных с реализацией проекта, в их составе целесообразно выделить:

* переменные издержки, величина которых будет меняться с изменением объемов производства и продаж. В состав переменных издержек включаются затраты на расходные материалы, заработная плата персонала с начислениями. Переменные издержки рассчитываются прямым счетом на единицу продукции, а затем пересчитываются на установленные объемы производства в планируемом периоде;
* постоянные издержки, величина которых является неизменной при изменении объемов производства и продаж. Величина этих издержек рассчитывается на месяц . В состав этих издержек включаются амортизация внеоборотных активов, вовлеченных в проект (см. табл. 8.8), затраты на оплату арендуемых активов (арендная плата за помещения, оборудование и т.п.), административные и прочие общие расходы предприятия, относимые на проект.

Таблица 8.8.

Производственно-сбытовые издержки

Планируемый период – месяц.

Объем производства и продаж в планируемом периоде – 1 шт.

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование показателей | Значение показателя на единицу продукции, р. |
| 1. | Переменные издержки | 205438 |
| 1.1. | Сырье и материалы | 1750 |
| 1.2. | Заработная плата производственных рабочих | 151200 |
| 1.3. | Отчисления на социальные нужды | 51408 |
| 1.4. | Затраты на содержание и эксплуатацию производственного оборудования | 1080 |
| 2. | Постоянные издержки | 3300 |
| 2.1. | Амортизация внеоборотных активов | 2650 |
| 2.2. | Арендная плата | 650 |
| Производственно-сбытовые издержки,  Всего | | 208738 |

Величина начальных инвестиций в оборотные средства определяется на основе расчета текущих производственно-сбытовых издержек, связанных с производством и сбытом новой продукции (см. табл. 8.8), планируемой длительности производственно-сбытового цикла продукции, а также ритмичности производства новой продукции.

Расчет величины начальных инвестиций в оборотные средства проведем по следующей формуле:

,

где  - переменные издержки на производство и реализацию единицы продукции, тыс.р/шт.;  - планируемый объем производства продукции, шт./мес.;  - величина постоянных издержек при производстве и реализации продукции в расчете на месяц, тыс. р;  - длительность производственно-сбытового цикла, 5 мес. Показатели  и  берутся из данных табл. 8.7.



Результаты расчета потребности в начальных инвестициях представляются в табличной форме (табл. 8.9).

Потребность в начальных инвестициях равна сумме капитальных вложениям во внеоборотные и оборотные активы. Результаты расчета потребности в начальных инвестициях представляются в табличной форме (табл. 8.9).

Таблица 8.9

Потребность в начальных инвестициях

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование показателей | Сумма, тыс. руб. |
| 1.1 | Нематериальные активы  в том числе: | 247 |
| 1.1.1 | * научно-техническая продукция | 247 |
| 1.1.2 | * прочие нематериальные активы | - |
| 1.2 | Основные средства  в том числе: | 82 |
| 1.2.1 | * здания и сооружения | 10 |
| 1.2.2 | * оборудование | 72 |
| 1.2.3 | * прочие основные средства | - |
| 1.3 | Прочие внеоборотные активы | - |
| 2 | Оборотные активы | 1050 |
| 2.1 | Запасы | - |
| 2.2 | Прочие оборотные активы | - |
|  | Потребность в начальных инвестициях, всего | 1379 |

## Экономическая эффективность

**Основные показатели**

1. Себестоимость = *329 тыс. руб.*
2. Амортизация

Для расчета амортизации можно применить способ списания стоимости пропорционально к объему произведенной продукции. При таком методе вначале следует вычислить амортизацию на единицу продукции при помощи формулы:

А = С / В,

где А – это сумма амортизации, применяемая на единицу продукции, С – это первоначальная стоимость объекта основных средств, а В – планируемый объем производства продукции.

1. A = *329 000* / *100* = *3290 руб*.

Тогда общая амортизация вычисляется по формуле:

1. Aобщ = A \* 25,
2. Aобщ = *3290 \* 25* = *82250* руб.

Период окупаемости вычисляется по формуле:

PBP = I / (π + Aобщ),

где I – инвестиции (эквивалентны себестоимости), π – доход за один период,

1. Aобщ – амортизация.

Примем стоимость одного экземпляра *3500 руб*., планируемое количество проданных копий – *50* в год:

1. PBP = *329 000 / (175 000 + 82 250) = 1,28 г. (470 дней)*

Показатель ROI вычисляется по формуле:

1. ROI = *(1 / 329 000) \* (175 000 \* 5) / 5 = 0,53*

Показатель NPV вычисляется по формуле:

1. NPV = *37 901 руб.*

## Вывод

Таблица 3.10

|  |  |
| --- | --- |
| Параметр | Значение |
| Себестоимость (руб.) | *329 000* |
| Срок окупаемости (дней) | *470* |
| Коэффициент качества | *2,1* |
| ROI (%) | *53 %* |
| NPV (руб.) | *37 901* |

Исходя из полученных результатов, можно сделать вывод, что разработка проекта является рентабельной.

Календарный план НИР

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Этапы и работы** | **Исполнитель** | **Трудоемкость, чел.-дн.** | **Длительность, дн.** | **Продолжительность работы (пятидневка)** | | | | | | | | | | | | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **1** | Формулирование целей и требований к проекту | Исп. | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** | Введение в предметную область | Исп. | 15 | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | Изучение литературы | Исп. | 20 | 20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4** | Изучение концептуальной, логической и физической моделей сущестующих систем | Исп. | 5 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5** | Анализ требований к программному комплексу | Исп. | 5 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **6** | Разработка технического задания | Исп. | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **7** | Разработка концептуальных моделей | Исп. | 15 | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **8** | Разработка системы связей с внешними источниками | Исп. | 30 | 30 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **9** | Разработка графического интерфейса | Исп. | 10 | 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** | Тестирование программного продукта | Исп. | 2 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **11** | Разработка документации | Исп. | 5 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Охрана интеллектуальной собственности

В этом разделе рассматриваются особенности программ для ЭВМ как объектов правовой охраны, а также вопросы защиты прав на них в рамках действующего законодательства. Раздел посвящен практическим вопросам, связанным с официальной регистрацией программ для ЭВМ в Роспатенте и особенностям их коммерческой реализации.

## 9.1 Программа для ЭВМ как объект правовой охраны

Законодательно регулируются правоотношения, которые возникают в результате создания РИД. В процессе дипломного проекта мною, Корсуковым Дмитрием, разработана программа для ЭВМ “Basest-real estate estimation”, которая относится к РИД.

Под программой для ЭВМ понимается "... представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определенного результата". Кроме того, в понятие программы для ЭВМ входят "...подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения".

С точки зрения программистов и пользователей программа для ЭВМ представляет собой детализацию алгоритма решения какой-либо задачи и выражена в форме определенной последовательности предписаний, обеспечивающих выполнение компьютером преобразования исходных данных в искомый результат.

Можно выделить следующие объективные формы представления программы для ЭВМ:

1. исходная программа (или исходный текст) - последовательность предписаний на алгоритмическом (понятном человеку) языке высокого уровня, предназначенных для автоматизированного перевода этих предписаний в последовательность команд в объектном коде;
2. рабочая программа (или объектный код) - последовательность машинных команд, т. е. команд, представленных на языке, понятном ЭВМ;
3. программа, временно введенная в память ЭВМ, - совокупность физических состояний элементов памяти запоминающего устройства ЭВМ (ОЗУ), сохраняющихся до прекращения подачи электропитания к ЭВМ;
4. программа, постоянно хранимая в памяти ЭВМ, - представленная на языке машины команда (или серия команд), выполненная в виде физических особенностей участка интегральной схемы, сохраняющихся независимо от подачи электропитания.

Исходная и рабочая программы, как правило, представляются в виде записи на том или ином языке, выполненной на бумаге или машиночитаемом носителе данных: магнитном или оптическом диске, магнитной ленте и т. п.

Предоставляемая законодательством правовая охрана распространяется "... на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код …". Так как преобразование исходного текста программы для ЭВМ в объектный (машинный) код с помощью специальных программ-трансляторов не меняет сущности данной программы как произведения, то если охраняется исходный текст программы, значит, охране подлежит и соответствующий ей объектный код. Обратное тоже справедливо.

Правовая охрана программ для ЭВМ распространяется только в отношении формы их выражения и «… не распространяется на идеи, концепции, принципы, методы, процессы, системы, способы, решения технических, организационных или иных задач, открытия, факты, языки программирования».

## 9.2 Официальная регистрация программ для ЭВМ

В ГК РФ закреплено право автора или иного правообладателя на государственную регистрацию программы для ЭВМ. Исключение составляют программы для ЭВМ, в которых содержатся сведения, составляющие государственную тайну.

Процедура официальной регистрации программ для ЭВМ включает подачу заявки в федеральный орган исполнительной власти по интеллектуальной собственности (Роспатент), проверку поданных документов и собственно регистрацию.

Заявка должна содержать следующие документы:

1. заявление о государственной регистрации;
2. депонируемые материалы, идентифицирующие программу для ЭВМ,

включая реферат;

1. документ, подтверждающий уплату государственной пошлины в

установленном размере или основание для освобождения от уплаты

государственной пошлины или уменьшения его размера.

Ниже представлен заполненный двухсторонний бланк заявления на официальную регистрацию программы для ЭВМ, созданной в рамках данного дипломного проекта. Также представлены депонируемые материалы, идентифицирующие программу для ЭВМ, реферат на регистрируемую программу. Для заполнения приводимых в этом разделе документов программа была названа «Basest-real estate estimation».

**ПРОГРАММА ДЛЯ ЭВМ**

**Basest – real estate estimation**

Исходный текст программы

Всего 39 листов

Правообладатель: Корсуков Дмитрий Владимирович

Автор: \_Корсуков Дмитрий Владимирович

(ф., и., о.)

(C) Корсуков Д. В., 2012

Санкт-Петербург

2012

**Р Е Ф Е Р А Т**

Автор: Корсуков Дмитрий Владимирович

Правообладатель: Корсуков Дмитрий Владимирович

Программа для ЭВМ: Basest – real estate estimation

Аннотация: Программный комплекс предназначен для автоматизации работы оценщика, предоставляет среду, где пользователь может производить вычисления, создавать рабочие отчеты, искать необходимые аналоги как оборудования так и строительных объектов, используя базы данных.

Тип ЭВМ: IBM PC/AT и совместимые с ней

ОС: Windows 95/98/МЕ/NT/2000/Vista/7

Язык программирования: Java

Объем: 2.2 Мбайт

**СОСТАВ ПРОГРАММЫ**

|  |  |  |
| --- | --- | --- |
| Basest.jar | - | Пользовательский клиент |
| BasestBD.jar | - | Клиент для заполнения БД |
| Basest.html | - | Главная страница сайта |
| style.css | - | Файл для описания стилей, необходимых для сайта |

## 9.3 Коммерческая реализация программ для ЭВМ

Коммерческая реализация программного продукта - программы для ЭВМ связана с понятием использования программы для ЭВМ третьими лицами и осуществляется на основании лицензионного договора с правообладателем.

Одним из типов лицензионного договора на программу для ЭВМ или базу данных является традиционный двухсторонний договор правообладателя – лицензиара, с покупателем (пользователем) - лицензиатом, в котором определяется способы, сроки, территория использования программы или базы данных. Такие договоры составляются, как правило, при:

▫ единичных продажах программного продукта, предназначенного для

решения достаточно узких прикладных задач;

▫ при продажах программного продукта, требующего регулярного

обновления и дополнения;

▫ передаче прав на тиражирование и распространение программного

продукта.

Ниже приведён лицензионный договор, на основании которого я, Корсуков Дмитрий, могу продать программный продукт “ Basest-real estate estimation ” СПбГЭТУ “ЛЭТИ”.

**ЛИЦЕНЗИОННЫЙ ДОГОВОР О ПРЕДОСТАВЛЕНИИ ПРАВА**

**ИСПОЛЬЗОВАНИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

**Стороны в Договоре:**

Гражданин Корсуков Дмитрий Владимирович, проживающий по адресу: 197342, г. Санкт-Петербург, ул. Торжковская, д. 6, кв. 9, именуемая в дальнейшем "ЛИЦЕНЗИАР", с одной стороны, и

Государственное образовательное учреждение высшего профессионального образования "Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" им. В.И.Ульянова (Ленина)", именуемый в дальнейшем "ЛИЦЕНЗИАТ", в лице проректора по научно-образовательной деятельности Шестопалова М. Ю., действующего на основании Устава, с другой стороны,

**принимая во внимание:**

1. что Лицензиар является автором и правообладателем программы для ЭВМ " Basest ";
2. Лицензиат желает получить на условиях настоящего Договора лицензию на использование упомянутой программы для ЭВМ с целью проведения научных исследований в области медицинского приборостроения;
3. Лицензиар готов предоставить Лицензиату такую лицензию,

**договорились о следующем.**

**1. Термины и их определения**

1. "ПРОГРАММА ДЛЯ ЭВМ (ПрЭВМ)" - программное обеспечение " Basest – real estate estimation ".
2. "ДОКУМЕНТАЦИЯ" - комплект документов, передаваемых Лицензиаром Лицензиату, включающий руководство пользователя по применению и обслуживанию программы для ЭВМ.
3. "ПРОИЗВОДСТВЕННАЯ ПЛОЩАДКА" - научные лаборатории и кафедры Лицензиата.
4. "РАБОЧЕЕ МЕСТО" - конкретная ЭВМ, на которой используется Программа для ЭВМ.

**2. Предмет Договора**

1. Лицензиар предоставляет Лицензиату на срок действия настоящего Договора и за вознаграждение, уплачиваемое Лицензиатом, неисключительную лицензию на использование ПрЭВМ. При этом Лицензиату предоставляется право на установку ПрЭВМ не более чем на 10 (десяти) Рабочих местах.
2. Лицензиар передает Лицензиату Документацию к ПрЭВМ.
3. Предоставленное Лицензиату в рамках настоящего Договора право ограничено Производственной площадкой.
4. Лицензиар осуществляет авторский контроль за соблюдением объемов использования ПрЭВМ по настоящему Договору, при этом Лицензиат обеспечивает возможность такого контроля.
5. Лицензиар сохраняет за собой право самому использовать ПрЭВМ и предоставлять неисключительные лицензии на право ее использования третьим лицам.

**3. Обеспечение Договора**

1. Лицензиар передает Лицензиату ПрЭВМ в объеме и виде, достаточном для ее использования, и Документацию в течение 15 (пятнадцати) дней со дня подписания настоящего Договора. ПрЭВМ передается Лицензиату в виде диска, содержащего ПрЭВМ. По факту передачи ПрЭВМ и Документации составляется акт сдачи-приемки с перечнем переданных материалов, подписываемый обеими Сторонами.
2. Если Лицензиат установит неполноту или неправильность полученных ПрЭВМ или Документации, то Лицензиар в течение 15 (пятнадцати) дней после сообщения ему об этом Лицензиатом обязан передать недостающие материалы или устранить недостатки ранее переданных ПрЭВМ и Документации.
3. Для оказания помощи в освоении ПрЭВМ Лицензиар по просьбе Лицензиата оказывает консультации пользователям ПрЭВМ.
4. Для целей использования ПрЭВМ в объеме, предусмотренном п. 2.1 настоящего Договора, Лицензиат может изготавливать в необходимом ему количестве копии ПрЭВМ и копии Документации.

**4. Усовершенствования**

1. Лицензиар обязуется незамедлительно информировать Лицензиата о всех произведенных им усовершенствованиях ПрЭВМ и, при желании Лицензиата, передать ему в согласованные сроки новые варианты ПрЭВМ. В отношении новых вариантов ПрЭВМ, переданных Лицензиаром Лицензиату, распространяются все условия настоящего Договора.
2. Лицензиат обязуется предоставлять Лицензиару информацию об использовании ПрЭВМ, которая могла бы быть полезной для усовершенствования ПрЭВМ.

**5. Платежи**

1. За предоставление прав, предусмотренных настоящим Договором, Лицензиат выплачивает Лицензиару единовременное вознаграждение в размере 60000(шестьдесят тысяч) рублей.
2. Вознаграждение, предусмотренное п. 5.1 настоящего Договора, выплачивается Лицензиатом в течение 30 (тридцати) дней, следующих после подписания акта приемки-сдачи.

**6. Реклама**

1. Лицензиат обязуется при опубликовании результатов исследований, полученных с использованием ПрЭВМ, сообщать в рекламных целях, что исследования производились с использованием ПрЭВМ Лицензиара с указанием авторского права Лицензиара.

**7. Защита передаваемых прав**

1. Лицензиат обязуется не вносить самовольно каких-либо изменений в ПрЭВМ и Документацию и не дополнять их какими-либо комментариями. Подобные изменения или дополнения возможны только с согласия Лицензиара.
2. Лицензиат обязуется предпринимать все необходимые меры для предотвращения несанкционированного копирования ПрЭВМ и Документации третьими лицами, а также несанкционированной передачи ПрЭВМ и Документации работниками Лицензиата третьим лицам.
3. Если Лицензиату станет известно о противоправном использовании ПрЭВМ третьими лицами, то он незамедлительно сообщит об этом Лицензиару.

**8. Ответственность Сторон и разрешение споров**

1. За невыполнение или ненадлежащее выполнение обязательств по настоящему Договору Стороны несут имущественную ответственность в соответствии с действующим законодательством.
2. Стороны освобождаются от ответственности за неисполнение или ненадлежащее исполнение обязательств, принятых по настоящему Договору, если неисполнение явилось следствием обстоятельств непреодолимой силы (форс-мажор).
3. Сторона, нарушившая свои обязательства по настоящему Договору, освобождается от ответственности за неисполнение или ненадлежащее исполнение этих обязательств, если это нарушение было вызвано причинами, за которые отвечает другая Сторона.
4. В случае возникновения споров между Лицензиаром и Лицензиатом по вопросам, предусмотренным настоящим Договором, Стороны примут все меры к разрешению их путем переговоров между собой. В случае невозможности разрешения указанных споров путем переговоров они будут разрешаться в порядке, предусмотренном действующим законодательством.

**9. Срок действия Договора и условия его расторжения**

1. Настоящий Договор заключен на срок 2 года и вступает в силу с даты его подписания обеими Сторонами.
2. По истечении срока действия настоящего Договора Лицензиат вправе использовать ПрЭВМ, включая усовершенствованные варианты, на Производственной площадке на любом количестве Рабочих мест. При этом обязательства Лицензиата, предусмотренные пп. 7.1 и 7.2 настоящего Договора, сохраняются бессрочно.
3. Действие настоящего Договора по обоюдному согласию Сторон может быть досрочно прекращено, но не ранее чем через три месяца после предложения об этом одной из Сторон. При этом Лицензиат не освобождается от обязательств по платежам, возникшим до расторжения настоящего Договора.
4. Настоящий Договор может быть досрочно расторгнут в одностороннем порядке со стороны Лицензиара из-за невыполнения Лицензиатом своих обязательств по пп. 7.1 или 7.2. В этом случае Лицензиат лишается права дальнейшего использования ПрЭВМ в любой форме и обязан вернуть ее Лицензиару.
5. Если Лицензиат откажется от дальнейшего использования ПрЭВМ, то он уничтожит все имеющиеся у него копии ПрЭВМ.

**10. Заключительные положения**

1. Все изменения и дополнения к настоящему Договору действительны только в тех случаях, если они совершены в письменной форме и подписаны обеими Сторонами.
2. Стороны не имеют права передавать свои права и обязательства по настоящему Договору третьим лицам без письменного согласия на то другой Стороны.
3. Во всем остальном, что не предусмотрено условиями настоящего Договора, будут применяться нормы законодательства Российской Федерации.

**11. Адреса Сторон**

1. ЛИЦЕНЗИАР: Корсуков Дмитрий Владимирович, адрес: 197342, г. Санкт-Петербург, ул. Торжковская, д. 6, кв. 9
2. ЛИЦЕНЗИАТ: СПбГЭТУ, адрес: 197376, Санкт-Петербург, ул. Проф. Попова, д. 5.

Настоящий Договор составлен в двух экземплярах для каждой из Сторон и подписан "\_\_\_"\_\_\_\_\_\_\_\_\_\_ 201\_\_ г. в г. Санкт-Петербурге.

|  |  |
| --- | --- |
| **ЛИЦЕНЗИАР:** | **От ЛИЦЕНЗИАТА:** |
| \_\_\_\_\_\_\_\_\_\_\_ Д. В. Корсуков | Первый проректор по научно-образовательной деятельности СПбГЭТУ  \_\_\_\_\_\_\_\_\_\_\_\_ М.Ю. Шестопалов |

ЗАКЛЮЧЕНИЕ

Целью создания дипломного проекта является программа, предназначенная для определения рыночной стоимости объектов недвижимости и земельных участков с последующим составлением отчета. Использование современных технологий, Java, SQL обеспечит быстрое и надежное функционирование объекта, а главное обеспечит легкость интегрирования в различные системы, за счет гибкой системы настройки под критерии пользователя, доступности и распространенности.

В рамках настоящего дипломного проекта разработана программа для автоматизации основных аспектов оценки недвижимости. Данная программа обеспечивает удобное управление расчетами, в том числе калькуляциями.

Описание работ по разработке программы автоматизации оценки недвижимости:

* Формулирование целей и требований к проекту.   
  На данном этапе идет анализ тех частей работы оценщика, которые подразумевают монотонную работу, расчеты и т.д.
* Введение в предметную область.

На данном этапе идет изучение предметной области, ознакомление с материалами (документами, экономическими расчетами), при необходимости консультации специалиста данной предметной области.

* Изучение литературы.

На данном этапе идет изучение функциональных возможностей различного рода программных сред, их взаимодействие с различного вида системами.

* Анализ требований к программному комплексу.

На данном этапе производится анализ основных, функциональных, системных и других требований к программному комплексу.

* Разработка концептуальных моделей.

На данном этапе идет разработка архитектуры системы, подсистем, а так же принципы их взаимодействия.

* Разработка графического интерфейса.

На данном этапе идет разработка графического интерфейса.

* Тестирование программного продукта.

На данном этапе проводится тестирование работы отдельных модулей и системы в целом.

* Оформление проекта, разработка документации, отладка

На данном этапе идет оформление проекта, разработка технической документации, руководства пользователя и отладка.

Для проведения экономического обоснования работы рассчитывается себестоимость научно-технической разработки на основании трудоемкости работ и сметы затрат по соответствующим статьям калькуляции. Этапы проведения экономического обоснования проекта:

* определение потребительских свойств продукта, связанных с совершенствованием продукта;
* анализ рынка сбыта проектируемой продукции и разработка плана маркетинга;
* описание процесса производства продукции и оценка ресурсов, необходимых для усовершенствования разрабатываемого продукта;
* описание конкретных работ и организационно-технических мероприятий, направленных на практическую реализацию проекта;
* оценка интегрального экономического эффекта, получаемого от модернизации программного продукта.

В завершении дипломного проекта было составлено технико-экономическое обоснование, где был определён экономический эффект, а также был приведён раздел, посвящённый охране интеллектуальной собственности.

# СПИСОК ЛИТЕРАТУРЫ

* 1. *Монахов Вадим.* [Язык программирования Java и среда NetBeans](http://barsic.spbu.ru/www/comlan/java_vm.html) — 3-е изд. — СПб.: [БХВ-Петербург](http://ru.wikipedia.org/wiki/%D0%91%D0%A5%D0%92-%D0%9F%D0%B5%D1%82%D0%B5%D1%80%D0%B1%D1%83%D1%80%D0%B3_%28%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%29), 2011. — 704 с.
  2. *Джошуа Блох.* Java. Эффективное программирование = Effective Java — М.: [Лори](http://ru.wikipedia.org/w/index.php?title=%D0%9B%D0%BE%D1%80%D0%B8_%28%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%29&action=edit&redlink=1), 2002. — 224 с.
  3. *Брюс Эккель.* Философия Java = Thinking in Java — 3-е изд. — СПб.: [Питер](http://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%82%D0%B5%D1%80_%28%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%29), 2003. — 976 с.
  4. *Герберт Шилдт, Джеймс Холмс.* Искусство программирования на Java = The Art of Java — М.: [Диалектика](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B0%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_%28%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%29&action=edit&redlink=1), 2005. — 336 с.
  5. *Любош Бруга.* Java по-быстрому: Практический экспресс-курс = Luboš Brůha. Java Hotová řešení — М.: [Наука и техника](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%83%D0%BA%D0%B0_%D0%B8_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0), 2006. — 369 с.
  6. <http://mexmat.sgu.ru> – Java пособие для проведения практических занятий
  7. <http://javaportal.ru> – пособие по объектно-ориентированному языку программирования Java

# ПРИЛОЖЕНИЕ