Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних технологій

**КУРСОВА РОБОТА**

з дисципліни “Об’єктно-орієнтоване програмування”

Тема “ Калькулятор для обчислення кількості необхідних матеріалів для виконання конкретної роботи в сфері будівництва, облаштування квартир новобудови ”

Студента (ки) курсу групи Полікарпов О. В.

(прізвище та ініціали)

Керівник Рудніченко М.Д.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (прізвище та ініціали)

м. Одеса – 2018 рік

Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних технологій

**ЗАВДАННЯ**

НА КУРСОВУ РОБОТУ

(прізвище, ім’я, по батькові) (група)

1. Тема роботи “ Калькулятор для обчислення кількості необхідних матеріалів для виконання конкретної роботи в сфері будівництва, облаштування квартир новобудови”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Термін здачі студентом закінченої роботи
3. Початкові дані до проекту (роботи): список користувачів та команд, в яких вони працюють; список всіх проектів команди, до якої входить користувач, додаткові дані про кожен з них: назва, адміністратор, дедлайн, опис; список справ користувача.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): вступ, аналіз специфіки предметної області, проектування, опис процесу роботи, висновки, перелік посилань, код програми\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання видано \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийнято до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание

1 ОСНОВНЫЕ ПОНЯНИЯ О ЯЗЫКЕ JAVA. ОСОБЕННОСТИ ПЛЮСЫ И МИНУСЫ И КЛАССИФИКАЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ…………………………………………………………3

1.2 Когда и зачем он был создан? История создания Java а так же его особенности………………………………………………………………….3 1.3 Основные знания о языке языка…………………………………...3

1.4 Основные возможности Java……………………………………..5

1.5 Преимущества и недостатки……………………………………..7

1.6 Вывод по главе 1…………………………………………………10

1.7 Что такое язык программирования?..............................................11

1.8 Низкие уровни программирования……………………………….13

1.9 Высокие уровни программирования…………………………….15

1.9.1 Процедурно-ориентированные языки …………………..19

1.9.2 Проблемно-ориентированные языки…………………….20

1.9.3 Объектно-ориентированные языки………………………20

* 1. Сверхвысокоуровневый язык программирования……………..25

2 ПРОЕКТИРОВАНИЕ

2.1 Backlog

2.2 БД

2.3 CRC

2.4 UML

2.4.1 Use Case

2.5 Прототип

Вывод…………………………………………………………………………….27

Литература………………………………………………………………………28

ГЛАВА 1 ОСНОВНЫЕ ПОНЯНИЯ О ЯЗЫКЕ JAVA. ОСОБЕННОСТИ ПЛЮСЫ И МИНУСЫ

1.1 Основные понятия Java

**Java** — сильно типизированный [объектно-ориентированный язык программирования](https://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](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle" \o "Oracle)). Приложения Java обычно [транслируются](https://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 специальный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой компьютерной архитектуре с помощью [виртуальной Java-машины](https://ru.wikipedia.org/wiki/Java_Virtual_Machine). Дата официального выпуска — 23 мая 1995 года. На 2018 год Java — один из самых популярных языков программирования.

` 1.2 История создания и особенности Java

Изначально язык назывался Oak («Дуб»), разрабатывался Джеймсом Гослингом для программирования бытовых электронных устройств. Из-за того, что язык с таким названием уже существовал, вскоре Oak был переименован в Java. Назван в честь марки кофе Java, которая, в свою очередь, получила наименование одноимённого острова (Ява), поэтому на официальной эмблеме языка изображена чашка с горячим кофе. Существует и другая версия происхождения названия языка, связанная с аллюзией на кофе-машину как пример бытового устройства, для программирования которого изначально язык создавался. В соответствии с этимологией в русскоязычной литературе с конца двадцатого и до первых лет двадцать первого века название языка нередко переводилось как Ява, а не транскрибировалось, как это стало общепринятым позднее.

Впоследствии язык стал использоваться для написания клиентских приложений и серверного программного обеспечения.

1.3 Основные знания о языке языка

Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой 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 снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java: применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках,аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами архитектуры ARM).

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, согласно которому отмечается существенно более низкая производительность и бо́льшее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на C++.

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, заложенной в основу платформы .NET компанией Microsoft.

1.4 Основные возможности Java

* [Автоматическое управление памятью](https://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_(%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)).
* Расширенные возможности обработки исключительных ситуаций.
* Богатый набор средств фильтрации ввода-вывода.
* Набор стандартных коллекций: [массив](https://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](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [стек](https://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](https://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](https://ru.wikipedia.org/wiki/RMI)).
* Наличие классов, позволяющих выполнять [HTTP](https://ru.wikipedia.org/wiki/HTTP)-запросы и обрабатывать ответы.
* Встроенные в язык средства создания многопоточных приложений, которые потом были портированы на многие языки (например [Python](https://ru.wikipedia.org/wiki/Python" \o "Python)).
* Унифицированный доступ к [базам данных](https://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](https://ru.wikipedia.org/wiki/SQL)-запросов — на основе [JDBC](https://ru.wikipedia.org/wiki/JDBC), [SQLJ](https://ru.wikipedia.org/wiki/SQLJ);
* на уровне концепции объектов, обладающих способностью к хранению в базе данных — на основе [Java Data Objects](https://ru.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1" \o "Java Data Objects (страница отсутствует)) ([англ.](https://en.wikipedia.org/wiki/Java_Data_Objects)) и [Java Persistence API](https://ru.wikipedia.org/wiki/Java_Persistence_API" \o "Java Persistence API).
* Поддержка обобщений (начиная с версии 1.5).
* Поддержка лямбд, замыканий, встроенные возможности функционального программирования (с 1.8).

Пример программы приведенный на рисунке 1.1

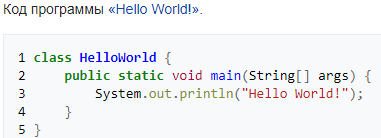


Рисунок 1.1 – пример программы

1.5 Преимущества и недостатки

Начнем с преимуществ языка

Простота

Синтаксис языка был унаследован от C++. Сегодня на фоне Python, Groove или Go его трудно назвать простым, однако тогда эволюционный вид позволил привлечь внимание Си-разработчиков.Такая форма конструкций не увеличивает скорость набора, однако читать, понимать и воспроизводить его просто.

Надёжность

Надёжность обеспечивается двумя принципами:

ООП. Иерархия наследования увеличивает читаемость кода и снижает количество невынужденных ошибок.Строгая типизация. Разработчику приходится выполнять больший объём работы, но данные интерпретируется однозначно.

Кроме того, первоначально в Java предполагался запрет прямого доступа к памяти, что также повышало бы надёжность. Но разработчики оставили несколько лазеек, например бэкдор sun.misc.Unsafe, которые этот запрет обходят.

Безопасность

Кроме сохранения общей формы конструкций, Java по сравнению с C++ формально лишился двух потенциальных опасностей: указателей и множественного наследования. На деле обе функции сохранены, но представлены в ином виде: вместо указателей используются значения, а в множественном наследовании участвуют не классы, а интерфейсы. Тем не менее, такая особенность java программирования почти исключает возможный урон от невнимательности разработчика.

Удобство

Дословно концепция Java звучит как: «Write once, run anywhere». То есть исполняемость кода не зависит от используемой операционной системы или установленного ПО. Достигается это благодаря транслированию в байт-код виртуальной машиной JVM.

Как нельзя кстати пригодилась эта особенность Java на android. Разнообразие производителей, моделей телефонов, характеристик — всё это могло бы негативно сказаться на работе приложений, если бы не существование такого универсального инструмента.

Производительность

Особенность Java, связанная с транслированием в байт-код, положительно сказывается и на производительности конечных продуктов. По скорости исполнения однотипные программы на Java уступают в 1,5-2 раза программам на C/C++, при этом превосходят JavaScript, Ruby, Python.

Развитая экосистема

За 22 года жизни язык оброс десятками IDE и фреймворков, сотнями сообществ и форумов, тысячами библиотек и плагинов. Всё это благоприятно сказывается на пороге вхождения в профессию, востребованности и качеству производимых с помощью java продуктов.

Безусловно, каждый популярный язык программирования уникален, каждый имеет свои недостатки и преимущества. Особенности Java не имеют революционного характера, они незначительны, но вместе с тем фундаментальны. Именно то, что отличает хороший язык от лучшего.

Закончим недостаткими языка

Менее быстрый и более требовательный

Низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти (ОП).

Большой объем стандартных библиотек и технологий создает сложности в изучении языка.

Постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение.

Так же там перечислялись некоторые особенности языка:

Узконаправленный

Скорее нейтральное качество чем недостаток но это и делает его узконаправленыым языком так как Java является полностью объектно-ориентированным языком. Например, C++ тоже является объектно-ориентированным, но в нем есть возможность писать программы не в объектно-ориентированном стиле, а в Java так нельзя.

Реализован с использованием интерпретации Р-кода (байт-кода). Т.е. программа сначала транслируется в машинонезависимый Р-код, а потом интерпретируется некоторой программой-интерпретатором (виртуальная Java-машина, JVM).

1.6 Вывод по главе 1

Язык Java подходит для вас если вы заинтересованы в объектно ориентированом програмировании так как он был для этого создан. Подведу итоги недостатков и достоинств: один из лучших языков для ООП и имеет легкий порог вхождения и понимания основных механик, но сложность заключается в дальнейшем так как будет много нового и сложного по сравнению с другими языками, а так же он требователен к оперативной памяти, но за это он гарантирует надежность.

1.7 Подробное описание языка программирования

**Язык программи́рования** — [формальный язык](https://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), предназначенный для записи [компьютерных программ](https://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%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Язык программирования определяет набор [лексических](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%B0), [синтаксических](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%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%D0%B5)) и [семантических](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%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%D0%B5)) правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — [ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE-%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_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0)) под её управлением.

Со времени создания [первых программируемых машин](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%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%BE%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B8#1835—1900-е:_первые_программируемые_машины) человечество придумало более восьми тысяч языков программирования (включая [эзотерические](https://ru.wikipedia.org/wiki/%D0%AD%D0%B7%D0%BE%D1%82%D0%B5%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%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), [визуальные](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%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 [игрушечные](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D1%83%D1%88%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)). Каждый год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты могут владеть десятком и более разных языков программирования.

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной [вычислительный процесс](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81), организовать управление различными объектами, и т. п. Язык программирования отличается от [естественных языков](https://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования [структурами данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и управления процессом вычислений.

Как правило, язык программирования определяется не только через спецификации стандарта языка, формально определяющие его [синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%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%D0%B5)) и [семантику](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%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%D0%B5)), но и через воплощения (реализации) стандарта — программные средства, обеспечивающих трансляцию или интерпретацию программ на этом языке; такие программные средства различаются по производителю, марке и варианту (версии), времени выпуска, полноте воплощения стандарта, дополнительным возможностям; могут иметь определённые ошибки или особенности воплощения, влияющие на практику использования языка или даже на его стандарт.

Основные требования, предъявляемые к языкам программирования:

наглядность - использование в языке по возможности уже существующих символов, хорошо известных и понятных как программистам, так и пользователям ЭВМ;

* Единство - использование одних и тех же символов для обозначения одних и тех же или родственных понятий в разных частях алгоритма. Количество этих символов должно быть по возможности минимальным;
* Гибкость - возможность относительно удобного, несложного описания распространенных приемов математических вычислений с помощью имеющегося в языке ограниченного набора изобразительных средств;
* Модульность - возможность описания сложных алгоритмов в виде совокупности простых модулей, которые могут быть составлены отдельно и использованы в различных сложных алгоритмах;
* Однозначность - недвусмысленность записи любого алгоритма. Отсутствие ее могло бы привести к неправильным ответам при решении задач.
* В настоящее время в мире существует несколько сотен реально используемых языков программирования. Для каждого есть своя область применения.
* Любой алгоритм, как мы знаем, есть последовательность предписаний, выполнив которые можно за конечное число шагов перейти от исходных данных к результату. В зависимости от степени детализации предписаний обычно определяется уровень языка программирования — чем меньше детализация, тем выше уровень языка.

По этому критерию можно выделить следующие уровни языков программирования:

* машинные;
* машинно-оpиентиpованные (ассемблеры);
* машинно-независимые (языки высокого уровня).
* Сверхвысокоуровневые

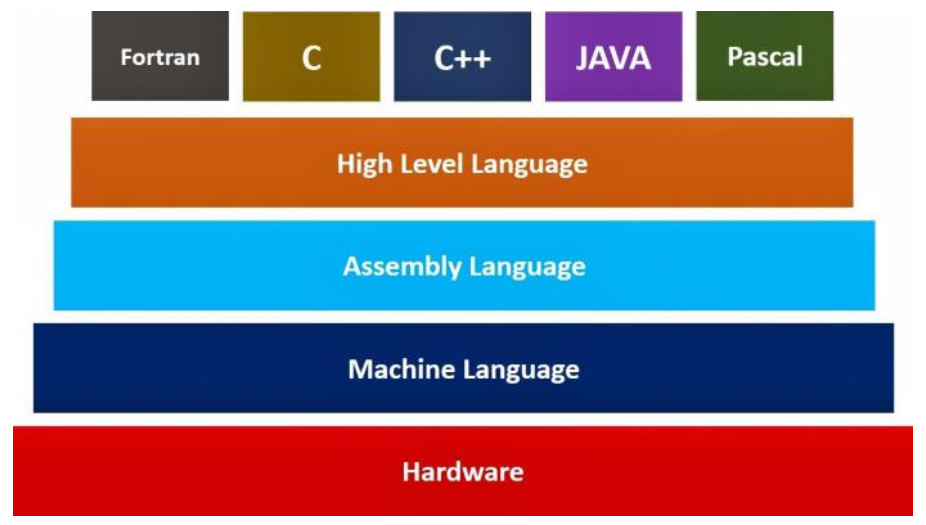


Рисунок 1.2 - Основные подгруппы языков программирования

Если язык близок к естественному языку программирования, то он называется языком высокого уровня, если ближе к машинным командам, – языком низкого уровня.

1.8 Низкие уровни программирования

**Низкоуровневый язык программирования** (язык программирования низкого уровня) — [язык программирования](https://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), близкий к программированию непосредственно в [машинных кодах](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) используемого реального или виртуального (например, байт-код, Microsoft .NET) процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских).

Иногда одно мнемоническое обозначение соответствует целой группе машинных команд, выполняющих одинаковое действие над разными ячейками памяти процессора. Кроме машинных команд языки программирования низкого уровня могут предоставлять дополнительные возможности, такие как [макроопределения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%80%D0%BE%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) (макросы). При помощи директив есть возможность управлять процессом трансляции машинных кодов, предоставляя возможность заносить константы и литеральные строки, резервировать память под переменные и размещать исполняемый код по определенным адресам. Часто эти языки позволяют работать вместо конкретных ячеек памяти с переменными.

Как правило, использует особенности конкретного семейства процессоров. Общеизвестный пример низкоуровневого языка — [язык ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0), хотя правильнее говорить о группе языков ассемблера. Более того, для одного и того же процессора существует несколько видов языка ассемблера. Они совпадают в машинных командах, но различаются набором дополнительных функций (директив и макросов).

Также к языкам низкого уровня условно можно причислить [CIL](https://ru.wikipedia.org/wiki/Common_Intermediate_Language), применяемый в платформе [Microsoft .NET](https://ru.wikipedia.org/wiki/Microsoft_.NET" \o "Microsoft .NET).

Языки же высокого уровня имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы. Эти языки более удобны для человека.

В группу языков низкого уровня входят машинные языки и языки символического кодирования: Автокод, Ассемблер. Операторы этого языка – это те же машинные команды, но записанные мнемоническими кодами, а в качестве операндов используются не конкретные адреса, а символические имена. Все языки низкого уровня ориентированы на определенный тип компьютера, т. е. являются машинно–зависимыми.Машинно–ориентированные языки – это языки, наборы операторов и изобразительные средства которых существенно зависят от особенностей ЭВМ (внутреннего языка, структуры памяти и т.д.).Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется языком программирования низкого уровня. В данном случае «низкий уровень» не значит «плохой». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.

При программировании на машинном языке программист может держать под своим контролем каждую команду и каждую ячейку памяти, использовать все возможности имеющихся машинных операций. Но процесс написания программы на машинном языке очень трудоемкий и утомительный. Программа получается громоздкой, труднообозримой, ее трудно отлаживать, изменять и развивать. Поэтому в случае, когда нужно иметь эффективную программу, в максимальной степени учитывающую специфику конкретного компьютера, вместо машинных языков используют близкие к ним машинно-ориентированные языки (ассемблеры). Язык ассемблера — это машинно-зависимый язык низкого уровня, в котором короткие мнемонические имена соответствуют отдельным машинным командам. Используется для представления в удобочитаемой форме программ, записанных в машинном коде. Язык ассемблера позволяет программисту пользоваться текстовыми мнемоническими (то есть легко запоминаемыми человеком) кодами, по своему усмотрению присваивать символические имена регистрам компьютера и памяти, а также задавать удобные для себя способы адресации. Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант, использовать в программе комментарии и др.

С помощью языков низкого уровня создаются очень эффективные и компактные программы, так как разработчик получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а окончательная программа не может быть перенесена на компьютер с другим типом процессора. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. В некоторых областях, например в машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие алгоритмы обработки изображений, требующие интенсивных вычислений.Таким образом, программы, написанные на языке ассемблера, требуют значительно меньшего объема памяти и времени выполнения. Знание программистом языка ассемблера и машинного кода дает ему понимание архитектуры машины. Несмотря на то, что большинство специалистов в области программного обеспечения разрабатывают программы на языках высокого уровня, таких, как Object Pascal или C, наиболее мощное и эффективное программное обеспечение полностью или частично написано на языке ассемблера.

1.9 Высокие уровни программирования

**Высокоуровневый язык программирования** — [язык программирования](https://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), разработанный для быстроты и удобства использования [программистом](https://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). Основная черта высокоуровневых языков — это [абстракция](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F), то есть введение смысловых конструкций, кратко описывающих такие [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [операции над ними](https://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), описания которых на [машинном коде](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (или другом [низкоуровневом языке программирования](https://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)) очень длинны и сложны для понимания.

Высокоуровневые языки программирования были разработаны для платформенной независимости сути алгоритмов. Зависимость от платформы перекладывается на инструментальные программы — [трансляторы](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80), компилирующие текст, написанный на языке высокого уровня, в элементарные машинные команды (инструкции). Поэтому, для каждой платформы разрабатывается платформенно-уникальный транслятор для каждого высокоуровневого языка, например, переводящий текст, написанный на [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%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)" \o "Delphi (язык программирования)) в элементарные команды [микропроцессоров](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) семейства [x86](https://ru.wikipedia.org/wiki/X86).Так, высокоуровневые языки стремятся не только облегчить решение сложных программных задач, но и упростить [портирование программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F" \o "Портирование программного обеспечения). Использование разнообразных [трансляторов](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) и [интерпретаторов](https://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) обеспечивает связь [программ](https://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%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), написанных при помощи языков высокого уровня, с различными [операционными системами](https://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) программируемыми устройствами и оборудованием, и, в идеале, не требует модификации [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (текста, написанного на высокоуровневом языке) для любой платформы.Такого рода оторванность высокоуровневых языков от аппаратной реализации [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) помимо множества плюсов имеет и минусы. В частности, она не позволяет создавать простые и точные инструкции к используемому оборудованию. Программы, написанные на языках высокого уровня, проще для понимания программистом, но менее эффективны, чем их аналоги, создаваемые при помощи [низкоуровневых языков](https://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). Одним из следствий этого стало добавление поддержки того или иного языка низкого уровня ([язык ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0)) в ряд современных профессиональных высокоуровневых языков программирования.

Примеры: [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%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)), [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [C#](https://ru.wikipedia.org/wiki/C_Sharp), [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%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)), [Fortran](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D1%82%D1%80%D0%B0%D0%BD), [Java](https://ru.wikipedia.org/wiki/Java), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), [Лисп](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%81%D0%BF), [Паскаль](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C_(%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)), [PHP](https://ru.wikipedia.org/wiki/PHP). Языкам высокого уровня свойственно умение работать с комплексными структурами данных. В большинстве из них интегрирована поддержка [строковых типов](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%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)), операций [файлового](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB) ввода-вывода и т. п.

Первым языком программирования высокого уровня считается [компьютерный язык](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [Plankalkül](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D0%BA%D0%B0%D0%BB%D0%BA%D1%8E%D0%BB%D1%8C" \o "Планкалкюль), разработанный немецким инженером [Конрадом Цузе](https://ru.wikipedia.org/wiki/%D0%A6%D1%83%D0%B7%D0%B5,_%D0%9A%D0%BE%D0%BD%D1%80%D0%B0%D0%B4) ещё в период [1942](https://ru.wikipedia.org/wiki/1942)—[1946 годах](https://ru.wikipedia.org/wiki/1946_%D0%B3%D0%BE%D0%B4). Однако транслятора для него не существовало до 2000 года. Первым в мире транслятором языка высокого уровня является ПП (Программирующая Программа), он же ПП-1, успешно испытанный в 1954 году. Транслятор ПП-2 (1955 год, 4-й в мире транслятор) уже был оптимизирующим и содержал собственный загрузчик и отладчик, библиотеку стандартных процедур, а транслятор ПП для [ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE-%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_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) [Стрела-4](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%B0_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE-%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_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0)) уже содержал и компоновщик (linker) из модулей. Однако, широкое применение высокоуровневых языков началось с возникновением [Фортрана](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D1%82%D1%80%D0%B0%D0%BD) и созданием [компилятора](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) для этого языка ([1957](https://ru.wikipedia.org/wiki/1957)).

Высокоуровневые языки программирования могут быть классифицированы по разным основаниям. Распространена схема, по которой они делятся на следующие основные разновидности:

* **процедурно-ориентированные** (задействуются в качестве инструмента при обработке информации на любом этапе вычислений);
* **проблемно-ориентированные** (используются как средство решения отраслевых и прикладных задач, формируемых при расширении областей применения ПК);
* **объектно-ориентированные** (могут быть частными случаями языков первых двух типов, однако, адаптируются к пользованию широким кругом разработчиков с разным уровнем подготовки, например, в виде решения с визуальным интерфейсом.

На сегодняшний момент по версии компании TIOBE Software лидирует язык программирования [Java](https://ru.wikipedia.org/wiki/Java" \o "Java). Рассмотрим теперь некоторые исторические и современные языки программирования высокого уровня, соответствующие данной классификации.

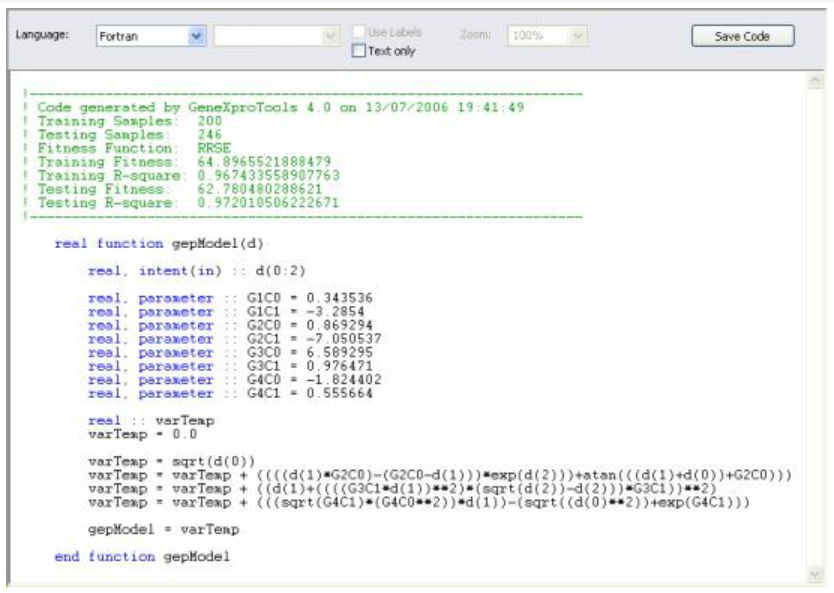


Рисунок 1.3 - Пример программы высокоуровнего языка программирования

1.9.1 Процедурно-ориентированные языки

К таковым можно отнести Фортран. Он считается первым языком программирования высокого уровня, созданным для широкого применения. Характеризуется простой структурой. К процедурно-ориентированным языкам также относится Бейсик. Считается одним из самых часто используемых при обучении программированию. Пример другого процедурно-ориентированного языка — СИ. Изначально он создавался для ОС UNIX. На его основе впоследствии был создан язык C++, дополненный инструментами объектно-ориентированного программирования. Еще один язык, относящийся к рассматриваемой категории — Паскаль. Часто также задействуется при обучении программированию. Возможности данного языка позволяют его использовать как очень мощный инструмент разработки профессиональных видов ПО.

1.9.2 Проблемно-ориентированные языки

К таковым можно отнести Лисп, Пролог. Первый язык был разработан в 1962 году — спустя несколько лет после создания Фортрана. Рассматривается, таким образом, как второй в истории. Активно задействовался в качестве инструмента работы программистов со строками символов. На практике Лисп использовался в системах, классифицируемых как экспертные, а также те, что предназначались для аналитических вычислений. Пролог нашел широкое применение в области логического программирования. На практике чаще всего задействуется в управлении алгоритмами искусственного интеллекта в соответствующих системах.

1.9.3 Объектно-ориентированные языки

Изучим теперь примеры языков программирования высокого уровня, которые относятся к категории объектно-ориентированных. В числе таковых — Visual Basic, Delphi, Visual Fortran, Java, отмеченный выше C++, а также Prolog ++. Фактически все они в своей основе содержат процедурно-ориентированные языки. Однако предполагается существенное их дополнение визуальными элементами управления с целью последующего освоения необходимых алгоритмов разработчиками, привыкшими к другим инструментам. Так, первый язык программирования высокого уровня — Фортран — может быть в оперативные сроки изучен IT-специалистами посредством возможностей Visual Fortran. Аналогичным методом можно быстро освоить Бейсик или Пролог.

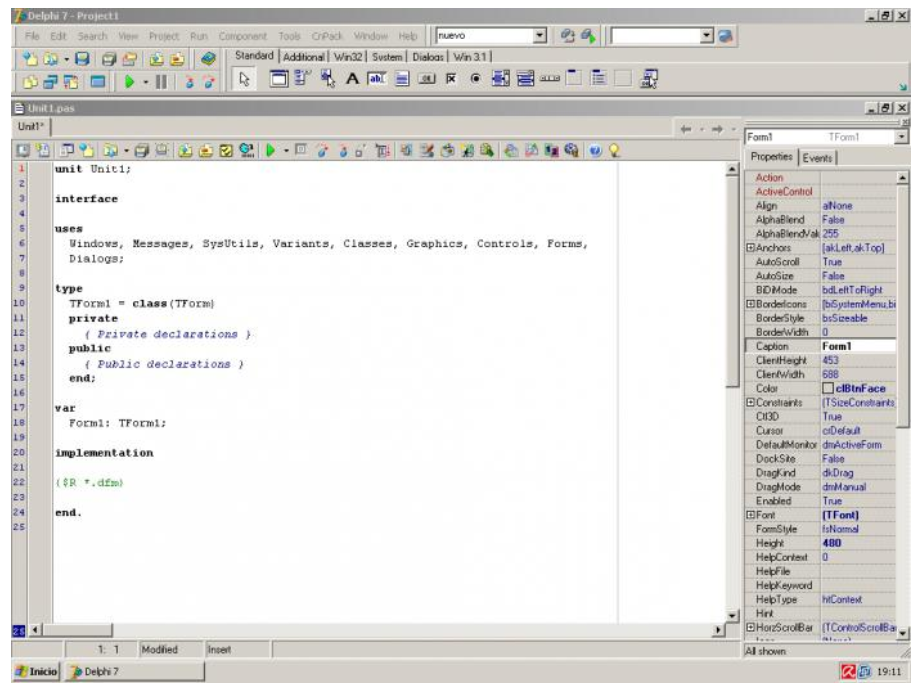


Рисунок 1.4 - пример программы Объектно-ориентированного программирования на языке Delphi

Осуществляется, в свою очередь, при использовании Delphi программирование на языке высокого уровня Object Pascal. Существует большое количество иных сред разработки ПО, классифицируемых как объектно-ориентированный язык. Данная сфера технологий разработки ПО активно развивается. Рассмотрим пример объектно-ориентированной программы на языке Java, но сначала, что такое само Java. **Java** — сильно типизированный [объектно-ориентированный язык программирования](https://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](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle" \o "Oracle)). Приложения Java обычно [транслируются](https://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 специальный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой компьютерной архитектуре с помощью [виртуальной Java-машины](https://ru.wikipedia.org/wiki/Java_Virtual_Machine). Дата официального выпуска — 23 мая 1995 года. На 2018 год Java — один из самых популярных языков программирования.

Программы на Java [транслируются](https://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 Java](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4_Java), выполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](https://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).

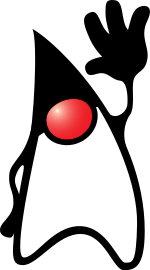
[](https://commons.wikimedia.org/wiki/File:Duke_(Java_mascot)_waving.svg?uselang=ru)

Рисунок 1.5 - Дюк, талисман Java

Достоинством подобного способа выполнения программ является полная независимость байт-кода от [операционной системы](https://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](https://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, в рамках которой 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 снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java:

* применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы ([JIT](https://ru.wikipedia.org/wiki/JIT)-технология) с возможностью сохранения версий класса в машинном коде,
* широкое использование [платформенно-ориентированного кода](https://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](https://ru.wikipedia.org/wiki/Jazelle" \o "Jazelle), поддерживаемая некоторыми процессорами [архитектуры ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0))).

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](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Google (компания)), согласно которому отмечается существенно более низкая производительность и бо́льшее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на [C++](https://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](https://ru.wikipedia.org/wiki/Common_Language_Infrastructure), заложенной в основу платформы [.NET](https://ru.wikipedia.org/wiki/.NET_Framework) компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft).

**import** java.util.Scanner;  
**public class** Calculator {  
 **public static final char** ADD = 43;  
 **public static final char** SUBTRACT = 45;  
 **public static final char** MULTIPLY = 42;  
 **public static final char** DIVIDE = 47;  
  
 **public static void** main(String[] args) {  
 **new** Calculator();  
 }  
  
 **public** Calculator() {  
 Scanner console = **new** Scanner(System.in);  
  
 **while** (**true**) {  
 System.out.print(**"First Number: "**);  
 **double** num1 = console.nextDouble();  
  
 System.out.print(**"Second Number: "**);  
 **double** num2 = console.nextDouble();  
 System.out.print(**"Operator (+-/\*): "**);  
 String input = console.next(**"[+-/\*]"**);  
 **char** operator = input.charAt(0);  
  
 **double** result = Double.NaN;  
  
 **switch** (operator) {  
  
 **case** ADD:  
 result = num1 + num2;  
 **break**;  
 **case** SUBTRACT:  
 result = num1 - num2;  
 **break**;  
 **case** MULTIPLY:  
 result = num1 \* num2;  
 **break**;  
 **case** DIVIDE:  
 result = num1 / num2;  
 **break**;  
 }  
  
 System.out.println(num1 + **" "** + operator + **" "** + num2 + **" = "** + result + **"\n"**);  
 }  
 }  
}

1.10 Сверхвысокоуровневый язык программирования

**Сверхвысокоуровневый язык программирования** (язык программирования сверхвысокого уровня, [англ.](https://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 high-level programming language*, VHLL) — [язык программирования](https://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) с очень высоким уровнем абстракции. В отличие от языков программирования высокого уровня, где описывается принцип «как нужно сделать», в сверхвысокоуровневых языках программирования описывается лишь принцип «что нужно сделать». Термин впервые появился в середине 1990-х годов для обозначения группы языков, используемых для быстрого прототипирования, написания одноразовых [скриптов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) и подобных задач.

Так, разработчики [Icon](https://ru.wikipedia.org/wiki/Icon_(%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)" \o "Icon (язык программирования)) (и его диалекта [Unicon](https://en.wikipedia.org/wiki/Unicon_(programming_language)" \o "en:Unicon (programming language)) (англ.)[русск.](https://ru.wikipedia.org/w/index.php?title=Unicon&action=edit&redlink=1)) описывают его как VHLL. К языкам сверхвысокого уровня также часто относят такие современные [сценарные](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) и [декларативные](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%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) (в частности [функциональные](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%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)) языки как [Python](https://ru.wikipedia.org/wiki/Python" \o "Python), [Ruby](https://ru.wikipedia.org/wiki/Ruby" \o "Ruby), [Haskell](https://ru.wikipedia.org/wiki/Haskell" \o "Haskell), а также [Perl](https://ru.wikipedia.org/wiki/Perl" \o "Perl) и предшествовавший ему мини-язык [AWK](https://ru.wikipedia.org/wiki/AWK).

Большой класс языков сверхвысокого уровня — это языки, используемые для специфических приложений и задач (то есть [предметно-ориентированные](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%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)). В связи с этой ограниченностью они могут использовать [синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%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%D0%B5)), который никогда не используется в других языках программирования, например, непосредственно синтаксис английского языка. Примером VHLL, распознающего синтаксис английского языка, может служить язык [компилятора](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) [текстовых квестов](https://ru.wikipedia.org/wiki/Interactive_Fiction) [Inform](https://ru.wikipedia.org/wiki/Inform" \o "Inform) версии 7.

2 ПРОЕКТИРОВАНИЕ

2.1 Backlog

**Видение системы:** система представляет собой десктопное приложение, позволяющее пользователю вводить данные квартиры в которой хочет сделать ремонт.

**Пользовательские роли:** пользователь, администратор.

**Работа с пользовательскими историями:**

Разрабатываем истории по принципу **Как <пользователь>, я могу <действие>, для того, чтобы <цель>**

1. Как пользователь я могу использовать программу для собственных нужд.
2. Как администратор я могу вносить изменения в базу данных различные стройматериалы.
3. Как пользователь я могу вводить свои личные данные, чтобы завести учетную запись.

* Должны быть введены настоящие данные пользователя.
* Учетная запись должна быть уникальной.

1. Как пользователь я могу изменить данные учетной записи.
2. Как пользователь я могу удалить свою учетную запись.

* Для удаления необходимо подтверждение администратора.

1. Как пользователь я могу ввести метраж своего помещения, для того, чтобы получить необходимые, рациональные вариации материалов для ремонта.
2. Как пользователь я могу хранить результаты своего списка материалов для ремонта помещений.

2.4 UML

2.4.1 Use-Case

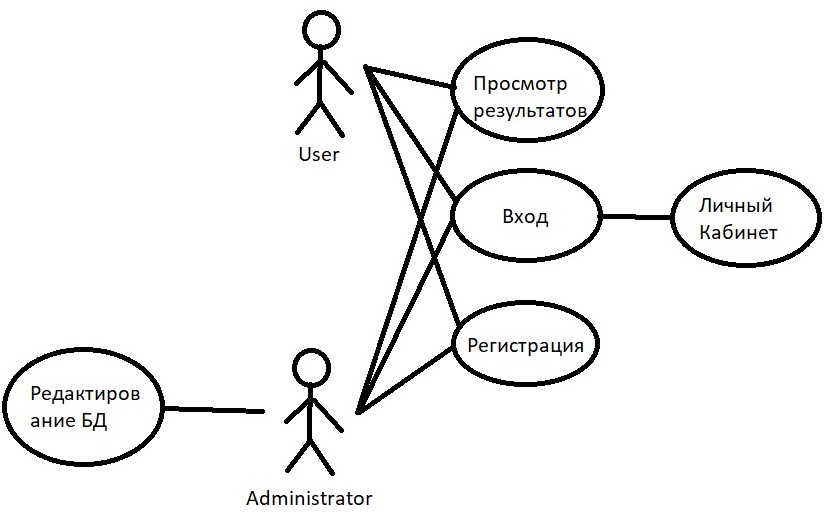


Рисунок - Диаграмма Use-Case

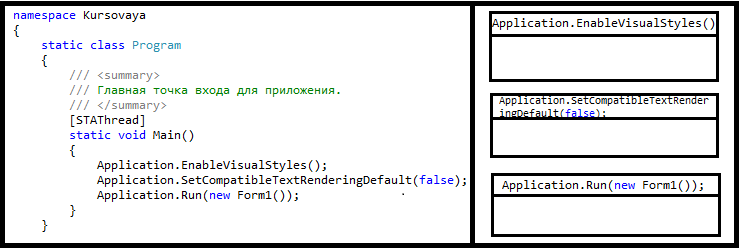
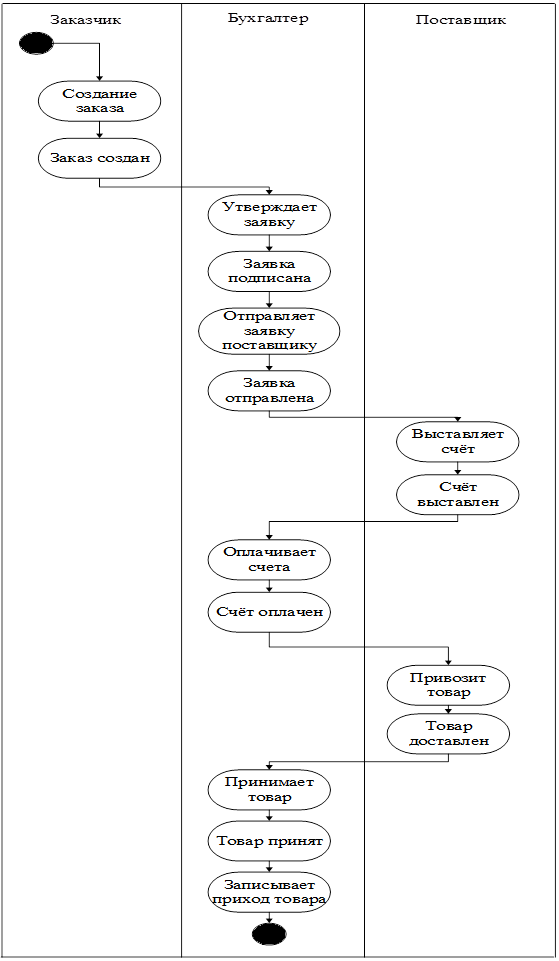


Рис. (1) - главный клас от которого все начинается



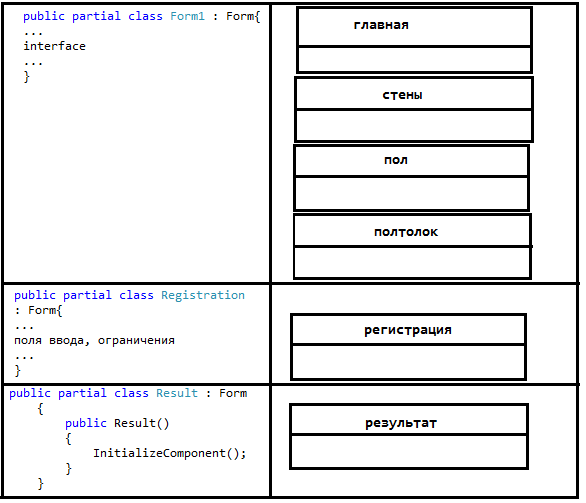


Рис. (2) - классы форм интерфейса, регистрации и результата.

**ЛИТЕРАТУРА**

* 1. <https://ru.wikipedia.org/wiki/Java>
  2. <https://www.syl.ru/article/199389/new_sovremennyiy-yazyik-programmirovaniya-vyisokogo-urovnya-primeryi-i-sravneniya>
  3. <https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%B5%D1%80%D1%85%D0%B2%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>
  4. <http://fb.ru/article/168378/spisok-yazyikov-programmirovaniya-yazyiki-programmirovaniya-nizkogo-i-vyisokogo-urovnya>
  5. <https://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>
  6. <https://studopedia.su/7_36667_yaziki-programmirovaniya-nizkogo-urovnya.html>
  7. <http://maksakov-sa.ru/ProgrProd/YazProgr/index.html>
  8. Начало работы с Java SE Шевченко Д.В. Проскурина Г.
  9. Java: Методы Программирования И.Н. Блинов В.С. Романчик