## Раздел 1. Общая часть

## Тема 1.1. Анализ методов решения

Существует два метода решения задач — ручной и автоматизированный.

Ручной метод решения задачи подразумевает собой организацию всей системы силами сотрудников предприятия, базируясь на принципах разделения труда, ведения документации на бумажных носителях. Данный метод утратил свою актуальность, его эффективность крайне мала, велик риск ошибок в процессе работы.

Автоматизированный метод представляет собой разработку программного обеспечения, обеспечивающего постоянную, быструю, стабильную работу системы книжного магазина, не требует ручного труда, позволяет хранить всю информацию в базе данных.

Для решения задачи выпускной квалификационной работы целесообразно выбрать автоматизированный метод разработки программного обеспечения, поскольку в век современных технологий деятельность любой организации, предприятия должна быть автоматизирована.

## Тема 1.2. Анализ инструментальных средств

На текущий момент существует много различных инструментов для разработки приложений, для работы с базой данных. Чтобы определиться с конечным вариантом используемого инструментария, необходимо провести сравнительный анализ — отобрать несколько программ для каждой задачи, сравнить их функционал, выбрать наиболее подходящую, удобную для разработчика. На сегодняшний день есть много различных СУБД. Их выбор обширен.

Для сравнительного анализа были выбраны СУБД PostgreSQL, MySQL, MSSQL Management Studio, так как они наиболее популярны и часто используемые в разработке и администрировании баз данных. Оценивались по следующим критериям: особенности, стоимость и максимальный размер базы данных. Результат сравнительного анализа средств СУБД представлен в таблице 1.

Таблица 1 — Сравнительный анализ средств СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | MSSQL Management Studio | MySQL | PostgreSQL |
| Множественное одновременное использование | - | + | + |
| Бесплатная | + | + | + |
| Неограниченное количество места | + | + | + |
| Защита от подбора пароля | - | - | + |
| Графический интерфейс | + | - | + |

В результате проведенного анализа СУБД, исходя из особенностей каждого из представленных вариантов, для использования в данной работе будет использоваться СУБД PostgreSQL, поскольку функционал данной СУБД обширен, интерфейс интуитивно понятен.

Следующий этап анализа инструментальных средства — выбор языка программирования и среды разработки самого приложения.

Для разработки приложения было выбрано три языка программирования — C#, Java и Python. Выбор конкретно данных языков обусловлен тем, что они наиболее востребованы на рынке профессий в данный момент, охватывают больше сфер разработки — от написания простых программ до создания сложных проектов с разработкой графического интерфейса и работы с базами данных. Для оценивания были выбраны следующие критерии: простота в изучении, процесс разработки программ, количество библиотек и производительность. Результат сравнительного анализа языков программирования представлен в таблице 2.

Таблица 2 — Анализ языков программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерии | | C# | Java | Python | |
| Высокий уровень ООП | | + | + | +- | |
| Простота в изучении | - | | +- | | + |
| Разрабатывать код можно где угодно | - | | - | | + |
| Большое количество библиотек | - | | +- | | + |
| Высокая производительность | + | | + | | - |
| Подходит для создания windows-приложений | +- | | + | | +- |

По итогу анализа в качестве используемого языка программирования был выбран Java, поскольку он специализируется на работе с объектно-ориентированным программированием, является быстрым языком, позволяют наполнить программу обширным функционалом.

Заключительный этап — выбор среды разработки.

Так как в качестве используемого языка программирования был выбран Java, необходимо выбирать и анализировать среды разработки, поддерживающие данный язык программирования, либо специализирующиеся только на данном языке программирования.

Интегрированная среда разработки (IDE) — комплекс программных средств, используемый программистами для разработки программного обеспечения.

Среда разработки включает в себя:

1. текстовый редактор;
2. транслятор (компилятор и/или интерпретатор);
3. средства автоматизации сборки;
4. отладчик.

На данный момент существует множество различных сред разработки. Выбор ограничивается лишь желанием разработчика.

Для сравнительного анализа были выбраны среды разработки Microsoft Visual Studio, Eclipse, Microsoft Visual Studio Code, так как являются наиболее популярными и часто используемыми. Проанализировав данные среды разработки, были определены преимущества и недостатки каждой из них. Результат анализа представлен в таблице 3.

Таблица 3 — Сравнительный анализ сред разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Microsoft Visual Studio | Eclipse | IntelliJ IDEA |
| Полностью бесплатная | + | + | +- |
| Большое количество расширений | + | +- | + |
| Тяжеловесная IDE | + | + | - |
| Встроенные СУБД | + | - | + |
| Поддержка GIT | + | - | + |
| Подсказки по ошибкам | + | - | + |
| Комбинации клавиш (помощь в написании кода) | - | - | + |

В результате выбор пал на среду разработки IntelliJ IDEA, так как данная IDE специализируется на разных языках, включая выбранный язык Java, не имеет ничего лишнего, обладает всеми средствами и инструментами для работы — установка различных расширений для работы с базами данных и разработки полноценного приложения Windows.

## Тема 1.3. Описание языка программирования

### Тема 1.3.1 Общие сведения

На сегодняшний момент язык Java является одним из самых распространенных и популярных языков программирования. Первая версия языка появилась еще в 1996 году в недрах компании Sun Microsystems, впоследствии поглощенной компанией Oracle. Java задумывался как универсальный язык программирования, который можно применять для различного рода задач. И к настоящему времени язык Java проделал большой путь, было издано множество различных версий. Текущей версией является Java 22, которая вышла в марте 2024 года. А Java превратилась из просто универсального языка в целую платформу и экосистему, которая объединяет различные технологии, используемые для целого ряда задач: от создания десктопных приложений до написания крупных веб-порталов и сервисов.

Кроме того, язык Java активно применяется для создания программного обеспечения для множества устройств: обычных ПК, планшетов, смартфонов и мобильных телефонов и даже бытовой техники. Достаточно вспомнить популярность мобильной ОС Android, большинство программ для которой пишутся именно на Java.

Особенности Java.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как С или С++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#. Поэтому, если вы знакомы с одним из этих языков, то овладеть Java будет легче.

Еще одной ключевой особенностью Java является то, что она поддерживает автоматическую сборку мусора. А это значит, что вам не надо освобождать вручную память от ранее использовавшихся объектов, как в С++, так как сборщик мусора это сделает автоматически за вас.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

Oracle JDK, OpenJDK и прочие JDK.

Для разработки на языке программирования Java нам потребуется специальный комплект инструментов, который называется JDK или Java Development Kit. Однако стоит отметить, что существуют разные реализации JDK, хотя все они используют один и тот же язык - Java. Две наиболее популярных реализации - Oracle JDK и OpenJDK.

Oracle JDK всецело развивается компанией Oracle. OpenJDK же представляет открытый проект, который развивается сообществом Java-разработчиков, а также рядом компаний, в том числе Oracle, Red Hat и рядом других.

Наибольшие различия с точки зрения лицензирования и поддержки. Согласно лицензии Oracle JDK можно использовать бесплатно для персональных нужд, а также для разработки, тестирования и демонстрации приложений. В остальных случаях (например, для получения поддержки) необходима коммерческая лицензия в виде подписки. А OpenJDK полностью бесплатна.

В плане функционала, набора возможностей Oracle JDK и OpenJDK практически не должны отличаться. А вот в плане производительности существуют замечания, что Oracle JDK может работать несколько быстрее, чем OpenJDK. Кроме того, некоторые разработчики отмечают, что OpenJDK чуть более глючная, а Oracle JDK более стабильная. Кроме того, Oracle гарантирует обновления в течение некоторого периода времени, c OpenJDK обновления под вопросом.

И стоит отметит, что есть еще куча реализаций от различных компаний. Например, компания Microsoft предоставляет свою реализацию. В частности, она может устанавливаться по умолчанию на некоторые версии Windows. И ее можно найти на Windows в папке C:\Program Files\Microsoft.

Для данного руководства не играет никакой роли, что использовать OpenJDK, JDK от Oracle, Microsoft или от какого-то другого вендора.

### Тема 1.3.2 Общая структура программ

Основным строительным блоком программы на языке Java являются инструкции (statement). Каждая инструкция выполняет некоторое действие, например, вызовы методов, объявление переменных и присвоение им значений. После завершения инструкции в Java ставится точка с запятой (;). Данный знак указывает компилятору на конец инструкции.

Например:

System.out.println("Hello Java!");

Данная строка представляет вызов метода System.out.println, который выводит на консоль строку "Hello Java!". В данном случае вызов метода является инструкцией и поэтому завершается точкой с запятой.

Кроме отдельных инструкций распространенной конструкцией является блок кода. Блок кода содержит набор инструкций, он заключается в фигурные скобки, а инструкции помещаются между открывающей и закрывающей фигурными скобками:

{

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

System.out.println("Welcome to Java!");

}

В этом блоке кода две инструкции, которые выводят на консоль определенную строку.

Выполнение программы. Метод main

Java является объектно-ориентированным языком, поэтому всю программу можно представить как набор взаимодействующих между собой классов и объектов. В первой главе при создании первого приложения программа была определена следующим образом:

public class Program{

public static void main (String args[]){

System.out.println("Hello Java!");

}

}

То есть основу нашей программы составляет класс Program. При определении класса вначале идет модификатор доступа public, который указывает, что данный класс будет доступен всем, то есть мы сможем его запустить из командной строки. Далее идет ключевое слово class, а затем название класса. После названия класса идет блок кода, в котором расположено содержимое класса.

Входной точкой в программу на языке Java является метод main, который определен в классе Program. Именно с него начинается выполнение программы. Он обязательно должен присутствовать в программе. При этом его заголовок может быть только таким:

public static void main (String args[])

При запуске приложения виртуальная машина Java ищет в главном классе программы метод main с подобным заголовком, и после его обнаружения запускает его.

Вначале заголовка метода идет модификатор public, который указывает, что метод будет доступен извне. Слово static указывает, что метод main - статический, а слово void - что он не возвращает никакого значения. Далее в скобках у нас идут параметры метода - String args[] - это массив args, который хранит значения типа String, то есть строки. При запуске программы через этот массив мы можем передать в программу различные данные.

После заголовка метода идет его блок, который содержит набор выполняемых инструкций.

Комментарии

Код программы может содержать комментарии. Комментарии позволяют понять смысл программы, что делают те или иные ее части. При компиляции комментарии игнорируются и не оказывают никакого влияния на работу приложения и на его размер.

В Java есть два типа комментариев: однострочный и многострочный. Однострочный комментарий размещается на одной строке после двойного слеша //. А многострочный комментарий заключается между символами /\* текст комментария \*/. Он может размещаться на нескольких строках. Например:

/\*

многострочный комментарий

Объявление нового класса,

который содержит код программы

\*/

public class Program{ // начало объявления класса Program

// определение метода main

public static void main (String args[]){ // объявление нового метода

System.out.println("Hello Java!"); // вывод строки на консоль

} // конец объявления нового метода

} // конец объявления класса Program

### Тема 1.3.3 Использования основных алгоритмических конструкций

В Java используются различные алгоритмические конструкции, позволяющие эффективно решать задачи программирования. Вот некоторые из них:

Линейный алгоритм - это последовательность действий, выполняемых однократно и строго последовательно. Пример:

public class LinearAlgorithm {

public static void main(String[] args) {

System.out.println("Привет мир!");

System.out.println("Это линейный алгоритм.");

}

}

Разветвляющийся алгоритм - это алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий. Пример:

import java.util.Scanner;

public class BranchingAlgorithm {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Введите число: ");

int number = scanner.nextInt();

if (number % 2 == 0) {

System.out.println("Число четное.");

} else {

System.out.println("Число нечетное.");

}

}

}

Циклический алгоритм - это алгоритм, команды которого повторяются некое количество раз подряд. В Java есть несколько видов циклов:

Постусловный цикл (while) - цикл, который выполняется до тех пор, пока условие истинно. Пример:

public class WhileLoop {

public static void main(String[] args) {

int i = 1;

while (i <= 10) {

System.out.println(i);

i++;

}

}

}

Предусловный цикл (do-while) - цикл, который сначала выполняет тело цикла, а затем проверяет условие. Пример:

public class DoWhileLoop {

public static void main(String[] args) {

int i = 1;

do {

System.out.println(i);

i++;

} while (i <= 10);

}

}

Цикл со счетчиком (for) - цикл, который удобно использовать, когда известно количество итераций. Пример:

public class ForLoop {

public static void main(String[] args) {

for (int i = 1; i <= 10; i++) {

System.out.println(i);

}

}

}

Эти примеры демонстрируют базовые алгоритмические конструкции в Java, которые позволяют создавать эффективные и понятные программы.

### Тема 1.3.4 Средства отладки программы

Отладка — важный аспект процесса разработки, позволяющий разработчикам выявлять и исправлять проблемы или ошибки в своих приложениях. Инструменты отладки позволяют разработчикам анализировать и глубоко погружаться в код, выявлять проблемы и устранять их для повышения производительности и стабильности приложения.

Инструменты отладки Java обычно предоставляют ряд функций, таких как установка точек останова, пошаговое выполнение кода, мониторинг переменных, оценка выражений и проверка стека вызовов. Используя эти инструменты, разработчики Java могут изолировать проблемы, оценить влияние на приложение и предпринять необходимые корректирующие действия. Некоторые популярные инструменты отладки Java включают:

1. Отладчик Java (jdb): jdb, входящий в состав Java Development Kit (JDK), представляет собой отладчик командной строки, который позволяет разработчикам устанавливать точки останова, пошагово выполнять код и проверять состояние приложения во время выполнения. Хотя jdb может не иметь дружественного интерфейса современных графических отладчиков, он остается полезным инструментом для разработчиков, которым комфортно работать с интерфейсами командной строки.
2. Перспектива отладки Eclipse: Eclipse IDE имеет специальную перспективу отладки с мощными функциями отладки. Разработчики могут устанавливать точки останова, пошагово выполнять код и проверять переменные и выражения во время выполнения кода. Благодаря интуитивно понятному интерфейсу и полной интеграции со IDE, Eclipse Debug Perspective является важным инструментом для разработчиков Java, работающих с Eclipse.
3. Отладчик IntelliJ IDEA: IntelliJ IDEA включает в себя отладчик, который предлагает расширенные возможности, такие как возможность оценки выражений на лету, условных точек останова и регистрации точек останова, а также представление памяти, которое помогает отслеживать утечки памяти. IntelliJ IDEA Debugger — лучший выбор для разработчиков Java, использующих IntelliJ IDEA IDE.

Включение инструментов отладки в процесс разработки позволяет разработчикам Java обнаруживать ошибки на раннем этапе, сокращая время, затрачиваемое на выявление и решение проблем, и приводя к созданию более стабильных и надежных приложений. Оттачивая свои навыки отладки и используя правильные инструменты, вы будете хорошо подготовлены к решению любых проблем, с которыми могут столкнуться ваши Java-приложения.

### Тема 1.3.5 Методы и средства взаимодействия с базой данных

Для взаимодействия с базами данных на Java используются различные методы и средства, которые обеспечивают эффективное управление данными. Вот некоторые из них:

JDBC (Java Database Connectivity) - стандартный API для подключения к базам данных. Позволяет выполнять запросы, вставлять, обновлять и удалять данные. Пример использования:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class JdbcExample {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/mydb";

String username = "root";

String password = "password";

try {

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

statement.executeUpdate("INSERT INTO users (id, name) VALUES (1, 'John Doe')");

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

JPA (Java Persistence API) - спецификация Java EE, обеспечивающая удобство работы с базами данных через объектно-реляционное отображение (ORM). Пример использования:

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class JpaExample {

public static void main(String[] args) {

EntityManagerFactory emf =

Persistence.createEntityManagerFactory("my-pu");

EntityManager em = emf.createEntityManager();

User user = new User();

user.setId(1);

user.setName("John Doe");

em.getTransaction().begin();

em.persist(user);

em.getTransaction().commit();

em.close();

emf.close();

}

}

Hibernate - популярная реализация JPA, предоставляющая дополнительные возможности и производительность. Пример использования:

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateExample {

public static void main(String[] args) {

Configuration configuration = new Configuration().configure();

SessionFactory sessionFactory = configuration.buildSessionFactory();

Session session = sessionFactory.openSession();

session.beginTransaction();

User user = new User();

user.setId(1);

user.setName("John Doe");

session.save(user);

session.getTransaction().commit();

session.close();

sessionFactory.close();

}

}

Эти методы и средства позволяют разработчикам эффективно взаимодействовать с базами данных, обеспечивая гибкость и удобство при работе с данными.

## Тема 1.4. Описание СУБД

PostgreSQL (или просто Postgres) - это мощная и надежная объектно-реляционная система управления базами данных (СУБД). Она предоставляет полный спектр функций для хранения и управления структурированными данными, включая поддержку сложных запросов, транзакций, индексирования и многих других возможностей.

PostgreSQL является открытым исходным кодом и поддерживает множество операционных систем, включая Linux, Windows, MacOS и другие. Он также обладает широкой поддержкой сторонних разработчиков и сообщества пользователей, что обеспечивает быстрые и эффективные обновления и исправления ошибок.

PostgreSQL предлагает расширенные возможности для работы с данными, такие как полнотекстовый поиск, географические информационные системы, JSON и XML обработка, а также поддержку многих языков программирования для написания хранимых процедур и функций.

В целом, PostgreSQL является надежным выбором для разработчиков и организаций, которым требуется надежная и мощная система управления базами данных для обработки больших объемов данных и выполнения сложных запросов.

Основные особенности PostgreSQL:

1. Объектно-реляционная модель: PostgreSQL поддерживает функциональности реляционных СУБД (такие как таблицы, запросы, индексы) в сочетании с объектно-ориентированными возможностями (классы, методы, наследование), что делает его более гибким и мощным.
2. Многопользовательская и многозадачная: Postgres поддерживает одновременное подключение нескольких пользователей и запросов, обеспечивая при этом высокую производительность и надежность.
3. Транзакции: PostgreSQL обеспечивает механизм транзакций для безопасного и согласованного доступа к данным. Это позволяет избежать потери данных и обеспечивает целостность базы данных.
4. Расширяемость: PostgreSQL обладает множеством встроенных и сторонних расширений, которые позволяют добавлять новые функциональности и интегрировать СУБД с другими приложениями.
5. Производительность: Postgres обладает высокой производительностью и масштабируемостью, поддерживая работу с большими объемами данных и высокой нагрузкой.
6. Поддержка множества языков программирования: PostgreSQL поддерживает множество языков программирования для написания хранимых процедур и функций (например, PL/pgSQL, PL/Python, PL/Perl).

PostgreSQL является одной из самых популярных СУБД в мире и широко используется как для разработки веб-приложений, так и для бизнес-приложений.