**Методические рекомендации к выполнению заданий**

* Работы обязательно сопровождаются отчетом, включающим код программы с комментариями и экранные формы результатов выполнения программного кода.
* Студент может выполнить не все задания, а только часть в каждой работе (если их несколько). Следует отметить, что в этом случае итоговая оценка будет рассчитываться пропорционально выполненным заданиям.

Проверяемое задание 1

**Тема 2. Persistence & Serialization. Персистентность и сериализация**

**Цель работы:** сформировать навыки работы с потоками ввода-вывода Java. Освоить технологию сериализации.

**Задание.** В процессе написания тестовых заданий ознакомиться с механизмом систем ввода и вывода данных.

**Рекомендации по выполнению задания**

1. Создайте класс Vectors, содержащий статические методы работы с векторами:

* умножение вектора на скаляр;
* сложение двух векторов;
* нахождение скалярного произведения двух векторов.

1. Модифицируйте класс Vectors, добавив в него новые методы:

* записи вектора в байтовый поток void outputVector(Vector v, OutputStream out);
* чтения вектора из байтового потока Vector inputVector(InputStream in);
* записи вектора в символьный поток void writeVector(Vector v, Writer out);
* чтения вектора из символьного потока Vector readVector(Reader in).

В обоих случаях записанный вектор должен представлять собой последовательность чисел, первым из которых является размерность вектора, а остальные числа являются значениями координат вектора.

В случае символьного потока рекомендуется считать, что один вектор записывается в одну строку (числа разделены пробелами). Для чтения вектора из символьного потока рекомендуется использовать класс StreamTokenizer.

Проверьте возможности методов (в методе main), в качестве реальных потоков используя файловые потоки, а также потоки System.in и System.out.

1. Модифицируйте классы ArrayVector и LinkedListVector (основанные на массиве и на связном списке) таким образом, чтобы они были сериализуемыми.

Продемонстрируйте возможности сериализации (в методе main), записав в файл объект, затем считав и сравнив с исходным (по сохраненным значениям).

1. Напишите MyClassToBePersisted.java, который содержит следующие свойства:

* поле профиля;
* поле группы.

Напишите SerializeMyClassToBePersisted.java, который создает экземпляр класса MyClassToBePersisted и сериализует его в файл в своем основном методе.

Напишите DeserializeMyClassToBePersisted.java, который считывает сериализованный файл и десериализует его в экземпляр класса MyClassToBePersisted в своем основном методе.

1. Предоставьте отчет, который содержит текст программного кода выполненных заданий и скрины результатов работы. Приложите к отчету скомпилированные программы по выполнению указанных заданий.

При оформлении кода использование комментариев обязательно.

Проверяемое задание 2

**Тема 3. Java NIO2**

**Цель работы:** сформировать навыки работы с технологиями NIO2 Java I/O.

**Задание.** Разработать программу ввода-вывода на основе применения технологии NIO2.

**Рекомендации по выполнению задания**

1. Разработайте программу, которая выводит свой код (содержимое файла \*.java). Во время разработки ОБЯЗАТЕЛЬНО пользуйтесь средствами NIO2:

* определите текущий каталог;
* имя файла;
* перейдите в каталог scr проекта;
* соберите имя файла с текстом программы (добавьте расширение \*.java).

1. Разработайте программу, которая записывает в типизированный файл целые числа, затем считывает и рекуррентно рассчитывает среднее.

В файл пишутся целые числа. Затем из этого файла как потока байт считываются числа byte, int и float.

* Произведите расчет считанных элементов при чтении данных разного типа.
* Определите количество чисел типа int.
* Рассчитайте среднее второй половины, третьей четверти.

1. Создайте текстовый файл, в него скопируйте не менее одной страницы документации по java. Считывайте символьный поток до тех пор, пока в третий раз не встретится слово «java».
2. Напишите пример, который подсчитывает, сколько раз конкретный символ, например e, появляется в файле. Символ можно указать в командной строке.
3. Предоставьте отчет, который содержит текст программного кода выполненных заданий и скрин результатов работы. Приложите к отчету скомпилированные при выполнении указанных заданий программы.

При оформлении кода использование комментариев обязательно.

Проверяемое задание 3

**Тема 4. Java Threads**

**Цель работы:** сформировать навыки работы с основными принципами создания многопоточных приложений.

**Задание.** В процессе написания тестовых заданий ознакомиться с общими принципами создания многопоточных приложений.

**Рекомендации по выполнению задания**

1. Создайте поток двумя способами:

* расширив класс Thread;
* переопределив его метод run(), реализовав интерфейс Runnable.

Если класс расширяет класс Thread, поток можно запустить, создав экземпляр класса и вызвав его метод start(). Продемонстрируйте пример расширения.

Если класс реализует интерфейс Runnable, поток можно запустить, передав экземпляр класса конструктору Thread-объекта и затем вызвав метод потока start(). Продемонстрируйте пример реализации.

1. Создайте два класса нитей (наследуют от класса Thread), взаимодействующих с помощью промежуточного объекта типа Vector.

Первая нить последовательно заполняет вектор (изначально он заполнен нулями) произвольными различными величинами (например случайными), отличными от нуля. Каждый раз, когда она помещает значение в вектор, она выводит на экран сообщение вида “Write: 100.5 to position 3”. По достижении конца вектора нить заканчивает свое выполнение.

Вторая нить последовательно считывает значения из вектора и выводит их на экран сообщениями вида “Read: 100.5 from position 3”. По достижении конца вектора нить заканчивает свое выполнение.

В методе main() следует создать 3 участвующих в процессе объекта и запустить нити на выполнение. Запустите программу несколько раз. Попробуйте варьировать приоритеты нитей.

1. Создайте два новых модифицированных класса нитей (реализуют интерфейс Runnable), обеспечивающих последовательность операций чтения-записи (т. е. на экран сообщения выводятся в порядке write-read-write-read-…) независимо от приоритетов потоков. Для этого потребуется описать некий вспомогательный класс, объект которого будет использоваться при взаимодействии нитей.
2. Добавьте в класс со статическими методами обработки векторов реализацию метода Vector synchronizedVector(Vector vector), возвращающего ссылку на оболочку указанного вектора, безопасную с точки зрения многопоточности. Для этого потребуется описать некий новый класс, реализующий интерфейс Vector, а также перегружающий методы класса Object.
3. Предоставьте отчет, в котором содержатся текст программного кода выполненных заданий и скрин результатов работы. Приложите к отчету скомпилированные программы по выполнению указанных заданий.

При оформлении кода обязательно используйте комментарии.

Проверяемое задание 4

**Тема 5. Java Threads. Синхронизация потоков (часть 2)**

**Цель работы:** сформировать навыки работы с технологиями Java Threads.

**Задание.** Разработать многопоточное приложение.

**Рекомендации по выполнению задания**

1. Объявите и инициализируйте глобальные данные/переменные, которые требуют синхронизации (например «count»). Объявите и инициализируйте связанный мьютекс.

Создайте потоки A и B для работы.

**Поток А**

* Работает до момента, когда должно произойти определенное условие (например, «счетчик» должен достичь указанного значения).
* Блокирует связанный мьютекс и проверяет значение глобальной переменной.
* Вызовает pthread\_cond\_wait(), чтобы выполнить блокирующее ожидание сигнала от Thread-B. Обратите внимание, что вызов pthread\_cond\_wait() автоматически и атомарно разблокирует связанную переменную мьютекса, чтобы она могла использоваться Thread-B.
* При получении сигнала поток А просыпается, мьютекс автоматически блокируется атомарно.

1. **Поток B**

* Меняет значение глобальной переменной, которую ожидает Thread-A.
* Проверяет значение глобальной переменной ожидания Thread-A. Если оно удовлетворяет желаемому условию, сигнализирует Thread-A.
* Разблокирует мьютекс.

1. Разработайте многопоточное приложение (использующее общую переменную):

* в потоке объявлена приватная переменная, например, i=0. При выполнении потока i инкриминируется, к примеру, 100 раз. При каждом инкременте на консоль выводится значение i;
* основная программа создает два потока. Необходимо проанализировать изменение i.

Выясните, как изменится поведение программы, если i объявить как локальную переменную (переменная метода run()).

1. Разработайте многопоточное приложение:

* один поток считывает текст собственной программы;
* второй поток получает считанную строку, анализирует вхождение образца и по условию выводит строку на экран.

1. Разработайте многопоточное приложение, реализующее кольцевой буфер в задаче «производитель – потребитель». Программа должна порождать процессы «производители» количеством K>1 и процессы «потребители» – M>1. Размер буфера N элементов. Частота работы производителя меньше в среднем в 10 раз частоты работы потребителя. Частоты задаются случайно. Производитель пишет в буфер в среднем 10 значений.
2. Предоставьте отчет, который содержит текст программного кода выполненных заданий и скрин результатов работы. Приложите к отчету скомпилированные при выполнении указанных заданий программы.

При оформлении кода использование комментариев обязательно.

Проверяемое задание 5

**Тема 7. Приложения с подключением баз данных**

**Цель работы:** сформировать навыки работы с технологиями Java DB.

**Задание.** Разработать приложение Java с подключением базы данных.

**Рекомендации по выполнению задания**

1. Разработайте приложение, которое автоматизирует одну из функций предметных областей.

Выбор функции осуществляется по первой букве фамилии.

|  |  |
| --- | --- |
| Первая буква ФИО | Функция |
| А, Б | Учет посещаемости студентов вуза |
| В, Г, Д | Учет успеваемости студентов в вузе |
| Е, Ё, Ж | Учет книг и журналов в библиотеке |
| З, И, Й | Учет компьютеров в организации |
| К, Л, М | Покупка билетов на самолет |
| Н, О, П | Покупка билетов в кинотеатр |
| Р, С, Т | Учет комплектующих компьютеров |
| У, Ф, Х, Ц | Учет программного обеспечения компьютеров |
| Ч, Ш, Щ, Ы | Учет мебели в организации |
| Э, Ю, Я | Регистрация в поликлинике |

Программа JDBC включает следующие шаги:

* Выделить Connection-объект для подключения к серверу базы данных.
* Выделить Statement-объект под Connection, созданный ранее для выполнения SQL-команды.
* Написать запрос SQL и выполнить его через операторы Statement и Connection.
* Обработать результат запроса.
* Закрыть Statement и Connection, чтобы освободить ресурсы.

Операции JDBC выполняются с помощью объектов Connection, Statement и ResultSet (определенных в пакете java.sql).

Обратите внимание, что использование программирования JDBC требует небольшого количества программирования. Потребуется только указать URL-адрес базы данных, написать запрос SQL и обработать результат запроса. Остальные коды представляют собой своего рода стандартный шаблон программы JDBC.

1. Создайте программу на языке Java без использования пользовательского интерфейса на основе сценариев:

* Сведения об объектах должны храниться в таблицах базы данных, чтение и редактирование данных осуществляться при помощи запросов SQL.
* Определите характеристики объектов. Обязательной характеристикой объекта является его уникальный идентификатор. Уникальность идентификаторов при выполнении операций добавления и редактирования объектов должна обеспечиваться средствами СУБД или средствами разрабатываемой программы.
* Программа должна выполнять следующие операции с данными:
  1. добавление нового объекта;
  2. изменение параметров существующего объекта;
  3. удаление объекта;
  4. поиск объектов по заданным критериям и вывод информации об объектах.

1. В установленной СУБД создайте новую базу данных с таблицами для хранения объектов в соответствии с заданной предметной областью.

Основные условия для выполнения задания:

* количество правильно спроектированных реляционных таблиц – не менее 5;
* при выполнении задания необходимо продемонстрировать использование PreparedStatement.

1. Установите соединение с базой данных из приложения Java. Управление соединением осуществляется через класс java.sql.Сonnection. Для установки соединения используется метод getConnection класса java.sql.DriverManager. Метод принимает на вход параметры соединения и в случае успеха возвращает объект класса Connection. В случае возникновения ошибки метод getConnection генерирует исключение SQLException.
2. Создайте запросы SQL через класс java.sql.Statement. Объект класса Statement создается в рамках заданного соединения при помощи метода createStatement класса Connection:

Statement s = con.createStatement();

1. Предоставьте отчет, в котором содержатся текст программного кода выполненных заданий и скрин результатов работы. Приложите к отчету скомпилированные программы по выполнению указанных заданий.

При оформлении кода использование комментариев обязательно.

Проверяемое задание 6

**Тема 8. Технологии по работе с XML**

**Цель работы:** сформировать навыки работы с технологиями Java по разбору XML-документов.

**Задание.** В процессе написания тестовых заданий ознакомиться с элементами языка XML и средствами Java для работы с XML-документами.

**Рекомендации по выполнению задания**

1. Разработайте схему XML-документа (XSD) для хранения:

* бинарного дерева;
* списка студентов;
* библиотеки;
* компьютерного оборудования;
* программного обеспечения на ПК.

1. Напишите программы, которые ЧИТАЮТ и ПИШУТ соответствующие файлы, используя:

* DOM- и SAX-парсеры сторонних разработчиков

или

* JAXP (поверх DOM и SAX).

1. Разработайте программу, которая подсчитает количество уникальных локальных имен элементов в XML-документе, для упрощения игнорируя имя пространства имен (можно взять любой готовый пример XML-документа).
2. Примените синтаксический анализатор Java XML при объявлении сотрудников компании. У каждого сотрудника есть уникальный идентификатор, имя и фамилия, возраст и зарплата. Сотрудники разделены идентификаторами. Создайте новый файл с именем Employees.xml. Выполните далее один из пунктов анализа:

* Разберите XML-файл с помощью парсера DOM. Реализация парсера DOM включена в выпуск JDK. Объектная модель документа предоставляет API-интерфейсы, которые позволяют создавать, изменять, удалять и переупорядочивать узлы. Парсер DOM анализирует весь XML-документ и загружает XML-содержимое в древовидную структуру. Используя классы [Node](https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/Node.html) и [NodeList](https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/NodeList.html), можно извлекать и изменять содержимое файла XML.
* Разберите XML-файл с помощью SAX-парсера. SAX – это API-интерфейс анализатора последовательного доступа, основанный на событиях, который предоставляет механизм для чтения данных из XML-документа, альтернативный предоставляемому парсером DOM. Синтаксическому анализатору SAX необходимо сообщать только о каждом событии синтаксического анализа по мере его возникновения, а минимальный объем памяти, необходимый для синтаксического анализатора SAX, пропорционален максимальной глубине XML-файла.
* Выполните синтаксический анализ XML-файла с помощью анализатора StAX. Streaming API for XML (StAX) – это интерфейс прикладного программирования для чтения и записи XML-документов. Парсер StAX – это анализатор XML, который может обрабатывать древовидные структурированные данные по мере их поступления. StAX был разработан как посредник между парсерами DOM и SAX. В анализаторе StAX точка входа – это курсор, который представляет точку в документе XML. Приложение перемещает курсор вперед, чтобы получить информацию из анализатора. Напротив, синтаксический анализатор SAX отправляет данные в приложение, а не извлекает их.
* Проанализируйте XML с помощью JAXB. Архитектура Java для связывания XML (JAXB) обеспечивает быстрый и удобный способ связывания схем XML и представлений Java, облегчая разработчикам Java включение данных XML и функций обработки в приложения Java. В рамках этого процесса JAXB предоставляет методы для демаршалинга (чтения) документов экземпляра XML в деревья контента Java и последующего маршалинга (записи) деревьев контента Java обратно в документы экземпляра XML. JAXB также предоставляет способ создания схемы XML из объектов Java.

Аннотации JAXB, определенные в пакете javax.xml.bind.annotation, могут быть использованы для настройки элементов программы Java на отображение схемы XML.

1. Предоставьте отчет, который содержит текст программного кода выполненных заданий и скрин результатов работы. Приложите к отчету скомпилированные при выполнении указанных заданий программы.

При оформлении кода использование комментариев обязательно.