Лабораторная работа №1

Задание на лабораторную работу

В процессе написания тестовых заданий ознакомиться со структурой исходного кода для Java, изучить особенности областей видимости и использования пакетов.

Рекомендуется каждое следующее задание выполнять в новом каталоге (например, Task1, Task2 и так далее).

Лабораторная работа выполняется в командной строке.

Задание 1

Запустите компилятор javac без параметров и ознакомьтесь с форматом задания параметров компилятора.

Запустите программу java без параметров и ознакомьтесь с форматом задания параметров запуска виртуальной машины Java (JVM).

Задание 2

Создайте файл MyFirstProgram.java, содержащий исходный код одного класса с именем MyFirstClass, обеспечивающий вывод на экран вашей фамилии.

Откомпилируйте файл с помощью компилятора javac. Для этого необходимо выполнить команду “javac MyFirstProgram.java”. Запустите полученный файл класса на выполнение с помощью команды “java <Имя класса>”.

Обратите внимание на то, что на вход компилятора необходимо подавать имя файла с расширением, а на вход JVM – имя класса (без расширения).

Также не забывайте использовать опцию –classpath, если файл вашего класса находится не в текущем каталоге.

Задание 3

Измените исходный код таким образом, чтобы ваша фамилия была выведена в столбик по буквам. Буквы фамилии должны задаваться как параметры командной строки.

Откомпилируйте и запустите программу, добавив в командную строку ряд аргументов. Например, следующим образом: “java MyFirstClass И в а н о в”.

Задание 4

В том же файле MyFirstProgram.java после описания класса MyFirstClass добавьте описание второго класса MySecondClass, реализующего следующую функциональность:

- имеет поле, которое описывает одномерный массив целых чисел;
- метод для получения значения элемента массива;
- метод для модификации значения элемента массива;
- конструктор, создающий объект массива заданного размера и инициализирующий его элементы случайными числами;
- метод для получения среднего арифметического (возвращает значение среднего арифметического, а не выводит его на экран);
- метод, обеспечивающий вывод элементов массива на экран.

Код метода main() MyFirstClass должен обеспечивать создание объекта типа MySecondClass, замену первого элемента массива на заданное число, вычисление среднего арифметического, вывод на экран значения среднего арифметического, вывод на экран элементов массива.

Добейтесь работоспособности программы.

Задание 5

Вынесите код класса MySecondClass без изменений в отдельный файл с именем MyFirstPackage.java, и поместите его в поддиректорию MyFirstPackage. Добейтесь работоспособности программы.

Задание 6

Запустите программу jar, предназначенную для создания архивов, и ознакомьтесь с форматом задания ключей для формирования архивов.

Скопируйте в рабочую папку, сохранив структуру каталогов, только файлы с расширением class, полученные в результате выполнения задания 5.

Создайте файл manifest.mf, содержащий следующий код:

--- manifest.mf ----------

Manifest-Version: 1.0

Created-By: <Ваши фамилии>

Main-Class: MyFirstClass

Обратите внимание на то, что после имени класса надо обязательно поставить символ новой строки.

Создайте архив myfirst.jar, включив в него полученные ранее файлы классов и указав созданный вами манифест-файл. Запустите его на выполнение с помощью команды “java –jar myfirst.jar ”.

Вопросы

1. Три направления развития платформы Java.
2. Характерные особенности языка Java.
3. Три принципа ООП. Пример.
4. Достоинства и недостатки ООП.
5. Классы и объекты. Свойства объектов. Пример.
6. Члены класса. Модификаторы объявления класса.
7. Пакеты.
8. Пространства имен.
9. Модуль компиляции.
10. Поля.
11. Методы. Метод main.
12. Модификаторы доступа.
13. Создание объектов. Конструкторы.
14. Блоки инициализации. Статическая инициализация.
15. Комментарии. Простые типы. Массивы.
16. Операторы. Циклы.

Лабораторная работа №2

Задание на лабораторную работу

В процессе написания тестовых заданий ознакомиться с механизмом исключений в Java и концепцией интерфейсов.

Задание 1

Написать класс Автомобиль. Он должен содержать:

поле типа String, хранящее марку автомобиля,

метод для получения марки автомобиля,

метод для модификации марки автомобиля,

внутренний класс Модель, имеющий поля название модели и её цену, а также конструктор (класс Автомобиль хранит массив Моделей),

метод для модификации значения названия модели,

метод, возвращающий массив названий всех моделей,

метод для получения значения цены модели по её названию,

метод для модификации значения цены модели по её названию,

метод, возвращающий массив значений цен моделей,

метод добавления названия модели и её цены (путем создания нового массива Моделей), использовать метод Arrays.copyOf(),

метод удаления модели с заданным именем и её цены, использовать методы System.arraycopy, Arrays.copyOf(),

метод для получения размера массива Моделей.

Конструктор класса должен принимать в качестве параметров значение Марки автомобиля и размер массива Моделей.

Задание 2

Написать класс Мотоцикл, реализующий функциональность, сходную с классом из задания 1, основанный на двусвязном циклическом списке с головой.

```
public class Мотоцикл {
    private class Модель{
        String название модели = null;
        double цена = Double.NaN;
        Модель prev = null;
        Модель next = null;
    }
     private Модель head = new Модель();
     {
      head.prev = head;
      head.next = head;
     }
      private int size = 0;
// далее код по заданию
}
```

Задание 3

Описать классы ошибок задания несуществующего имени модели NoSuchModelNameException (объявляемое), дублирования названия моделей DuplicateModelNameException (объявляемое), задание неверной цены модели ModelPriceOutOfBoundsException (необъявляемое).

Изменить методы классов так, чтобы они корректно обрабатывали ошибки и выбрасывали исключения.

Задание 4

Описать интерфейс Транспортное средство имеющий методы, соответствующие общей функциональности двух созданных классов. Сделать так, чтобы оба класса реализовывали этот интерфейс.

Задание 5

Написать класс со статическими методами таким образом, чтобы он работал со ссылками типа интерфейса. В классе должен быть метод, возвращающий среднее арифметическое цен моделей для заданного Транспортного средства и методы, обеспечивающие вывод на экран всех моделей и всех цен на модели для заданного Транспортного средства.

Вопросы

1. Исключения. Родительский класс исключений. Выбрасывание исключений.
2. Объявляемые и необъявляемые исключения. Пример.
3. Синхронные и асинхронные исключения. Пример.
4. Предложение throws.
5. try, catch и finally.
6. Интерфейсы. Модификаторы в объявлениях интерфейсов. Пример простого интерфейса.
7. Объявление интерфейса. Константы и методы в интерфейсах.
8. Расширение интерфейсов. Наследование и сокрытие констант. Наследование, переопределение и перегрузка методов.
9. Пустые интерфейсы. Пример.
10. Отличия абстрактного класса от интерфейса.

Лабораторная работа №3

Задание на лабораторную работу

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

Задание 1

Модифицировать класс со статическими методами из предыдущей работы, добавив в него новые методы:

- записи информации о транспортном средстве в байтовый поток (использовать DataOutputStream)

  public static void outputТранспортное средство (Транспортное средство v, OutputStream out),

- чтения информации о транспортном средстве из байтового потока (использовать DataInputStream)

  public static Транспортное средство inputТранспортное средство (InputStream in),

- записи информации о транспортном средстве в символьный поток (использовать PrintWriter)

  public static void writeТранспортное средство (Транспортное средство v, Writer out),

- чтения информации о транспортном средстве из символьного потока (использовать BufferedReader или StreamTokenizer)

  public static Транспортное средство readТранспортное средство (Reader in).

В обоих случаях нужно записать марку транспортного средства, количество моделей, а затем список моделей и цен моделей. При записи строки в байтовый поток использовать метод getBytes() для перевода строки в массив байт. Перед строкой нужно записать её длину.

Проверить возможности методов (в методе main), в качестве реальных потоков используя файловые потоки (FileInputStream, FileOutputStream, FileReader и FileWriter), а также потоки System.in и System.out.

Задание 2

Модифицировать классы Автомобиль и Мотоцикл таким образом, чтобы они были сериализуемыми.

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

Вопросы

1. Потоки данных. Байтовые потоки. Базовые абстрактные классы байтовых потоков.
2. Символьные потоки. Базовые абстрактные классы символьных потоков.
3. Стандартные потоки.
4. InputStreamReader и OutputStreamWriter.
5. Примеры байтовых и символьных классов потоков.
6. Сериализация объектов. Подготовка классов к сериализации.
7. Порядок сериализации и десериализации. Настройка механизма сериализации.
8. Контроль версий объектов.

Лабораторная работа №4

Задание на лабораторную работу

В процессе написания тестовых заданий ознакомиться с методами класса Object и расширить функциональность имеющегося пакета.

Задание 1

Добавить в классы Автомобиль и Мотоцикл реализации методов String toString(). Использовать для формирования строки экземпляр класса StringBuffer.

Задание 2

Добавить в классы Автомобиль и Мотоцикл реализации методов boolean equals(Object obj). Метод должен возвращать true только в том случае, если объект, на который передана ссылка, является Транспортным средством и имеет такую же марку, список моделей и цен моделей, что и текущий объект. Использовать instanceof.

Задание 3

Добавить в классы Автомобиль и Мотоцикл реализации методов int hashCode().

Задание 4

Добавить в классы Автомобиль и Мотоцикл реализации методов Object clone(). Клонирование должно быть глубоким. Использовать super.clone().

Вопросы

1. Расширенный класс. Конструкторы расширенных классов. Порядок выполнения конструкторов.
2. Перегрузка и переопределение методов.
3. Сокрытие полей. Доступ к унаследованным членам. Возможность доступа и переопределение.
4. Сокрытие статических членов. Служебное слово super.
5. Совместимость. Явное преобразование типов.
6. Проверка типа.
7. Методы и классы final.
8. Методы и классы abstract.
9. Класс Object. Методы класса Object. Клонирование объектов.

Практическая работа №5

Задание на практическую работу

Ознакомиться с возможностями механизма рефлексии, новыми возможностями языка Java, появившимися в версии 5.0, коллекциями.

Задание 1

Написать код (можно в методе main()), который с помощью рефлексии вызывает метод для модификации значения цены модели по её названию класса Автомобиль.

В параметрах командной строки приложения указывается полное имя класса, имя метода, который следует вызвать у класса (метод нестатический) и числовые параметры для этого метода (типа int).

На экран должен быть выведен результат выполнения этого метода (информация об автомобиле, включая марку автомобиля, названия всех моделей и их цены).

Задание 2

В классе со статическими методами написать метод создания транспортного средства, который получает значение марки автомобиля, размер массива Моделей, ссылку типа интерфейса Транспортного средства, по которой средствами рефлексии определяет реальный класс объекта, находит в нем конструктор и создает объект средствами рефлексии и того же класса, что и переданный параметр. Если конструктор с параметрами типа String и int отсутствует, то следует вернуть пустую ссылку.

Задание 3

Добавить новый класс Скутер, использующий для хранения моделей транспортного средства коллекцию HashMap, причем в параметризованной форме. Марка скутера хранится в поле типа String. Класс должен реализовывать интерфейс Транспортное средство.

Задание 4

Добавить новый класс Квадроцикл, использующий для хранения моделей транспортного средства коллекцию ArrayList, причем в параметризованной форме. Марка квадроцикла хранится в поле типа String. Класс должен реализовывать интерфейс Транспортное средство.

Задание 5

Добавить новый класс Мопед, использующий для хранения моделей транспортного средства коллекцию LinkedList, причем в параметризованной форме. Марка мопеда хранится в поле типа String. Класс должен реализовывать интерфейс Транспортное средство.

Задание 6

Добавить в класс со статическими методами метод, возвращающий среднее арифметическое цен моделей для заданного массива Транспортных средств (с использованием аргумента переменной длины).

Задание 7

Изменить методы текстового чтения и записи в классе со статическими методами таким образом, чтобы они использовали возможности форматированного ввода и вывода. Метод записи должен использовать метод printf(), а метод чтения - класс Scanner.

Вопросы

1. Рефлексия. Возможности механизма рефлексии. Участники механизма рефлексии.
2. Получение представления класса. Возможности класса Class. Передача параметров в методы. Создание экземпляров классов. Вызов методов. Вызов статического метода.
3. Статический импорт.
4. Автоупаковка и автораспаковка (автобоксинг).
5. Аргументы переменной длины.
6. Улучшенный цикл for.
7. Настраиваемые типы. Особенности настраиваемых типов.
8. Ограниченные типы. Метасимвольный аргумент. Метасимвол с ограничениями.
9. Настраиваемые методы, конструкторы, интерфейсы.
10. Перечислимые типы.
11. Метаданные.
12. Классы-обертки примитивных типов.
13. Класс Math. Класс String и класс StringBuffer. Класс Arrays.
14. Классы для работы со временем и локализацией. java.util.Random.
15. Коллекции. Интерфейс Collection.
16. Интерфейс Set. Интерфейс List.
17. Интерфейс Iterator. Интерфейс Map.
18. Классы коллекций.
19. Класс Collections. Синхронизированные обертки. Неизменяемые обертки.

Лабораторная работа №6

Задание на лабораторную работу

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

Задание 1

Создать два класса нитей (наследуют от класса Thread), взаимодействующих с помощью промежуточного объекта типа интерфейс Транспортного средства.

Первая нить последовательно выводит на экран цены на модели транспортного средства.

Вторая нить последовательно выводит на экран названия моделей транспортного средства.

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

Задание 2

Создайте два новых класса нитей (реализуют интерфейс Runnable), обеспечивающих последовательность операций вывода моделей и цен на модели (т.е. на экран выводятся модель-цена-модель-цена…) независимо от приоритетов потоков. Для этого потребуется описать некий вспомогательный класс TransportSynchronizer, объект которого будет использоваться при взаимодействии нитей.

```
public class TransportSynchronizer {
    private Transport v;
    private volatile int current = 0;
    private Object lock = new Object();
    private boolean set = false;

    public TransportSynchronizer(Transport v) {
        this.v = v;
    }

    public double printPrice() throws InterruptedException {
        double val;
        synchronized(lock) {
            double [] p = v.getPrices();
            if (!canPrintPrice()) throw new InterruptedException();
            while (!set)
                lock.wait();
            val = p[current++];
            System.out.println("Print price: " + val);
            set = false;
            lock.notifyAll();
        }
        return val;
    }

    public void printModel() throws InterruptedException {
        synchronized(lock) {
            String [] s = v.getModels();
```

```
            if (!canPrintModel()) throw new InterruptedException();
            while (set)
                lock.wait();
            System.out.println("Print model: " + s[current]);
            set = true;
            lock.notifyAll();
        }
    }

    public boolean canPrintPrice() {
        return current < v.getSize();
    }

    public boolean canPrintModel() {
        return (!set && current < v.getSize()) || (set && current < v.getSize() - 1);
    }
}
```

Задание 3

Создайте два новых класса нитей (реализуют интерфейс Runnable), обеспечивающих вывод сначала всех моделей, а затем всех цен на модели (возможен вывод сначала всех цен, а затем всех моделей). Использовать ReentrantLock.

Задание 4

Создайте новый класс нитей (реализует интерфейс Runnable), обеспечивающий вывод на экран марки транспортного средства. В методе main() следует создать четыре участвующих в процессе транспортных средства и пул потоков размерностью два. Использовать класс Executors. Запустить программу несколько раз и проанализировать результаты.

Задание 5

Создайте новый класс нитей (реализует интерфейс Runnable), обеспечивающий считывание из файла названия марки и создание экземпляра класса, реализующего интерфейс Transport. Имя файла передаётся как параметр в конструктор нити. Также вторым параметром передаётся экземпляр BlockingQueue (рекомендуется использовать реализацию ArrayBlockingQueue). Нить должна считать название марки транспортного средства из файла, создать экземпляр транспортного средства и положить его в очередь, используя метод put().

Создайте вручную 5 текстовых файлов, в каждом из которых добавьте одну строчку, содержащую название марки автомобиля.

В методе main() следует создать экземпляр ArrayBlockingQueue и завести массив имён созданных файлов, для каждого из которых в цикле необходимо создать и запустить на исполнение экземпляр нити, передав в конструктор путь к очередному файлу и ссылку на очередь.

Далее в цикле необходимо считать из очереди все пришедшие транспортные средства, используя метод take(), и вывести их названия в консоль. Посмотрите, как меняется исполнение программы при замене метода put() на add() и при варьировании длины очереди от 1 до 5.

Вопросы

1. Проблемы однопоточного подхода. Особенности многопоточности.
2. Использование класса Thread.
3. Использование интерфейса Runnable.
4. Управление потоками. Нерекомендуемые действия над потоками. Прерывание потока.
5. Группы потоков. Операции в группе потоков.
6. Приоритеты потоков.
7. Демон-потоки. Пример. Демон-группы потоков.
8. Совместное использование ресурсов. Характерные ошибки.
9. volatile.
10. Специальные методы класса Object. Особенности использования методов класса Object.
11. java.util.concurrent.locks. Lock, ReentrantLock, ReadWriteLock.
12. Интерфейсы Callable и Future.
13. Интерфейсы Executor, ExecutorService, ScheduledExecutorService.
14. Пул потоков. Executors.

Лабораторная работа №7

Задание на лабораторную работу

В процессе написания тестовых заданий ознакомиться с базовыми принципами создания клиент-серверных приложений, основанных на применении сокетов.

Задание 1

Реализовать клиентскую часть приложения в виде метода `main()` нового класса.

Программа должна установить через сокеты соединение с сервером, после чего сериализовать массив объектов типа Транспортное средство, передать его серверу и получить от него в качестве результата среднее арифметическое значение цен моделей данного массива. Результат следует вывести на экран.

Задание 2

Реализовать (в отдельном модуле компиляции) серверную часть приложения в рамках модели последовательной обработки запросов.

Задание 3

Реализовать (в отдельном модуле компиляции) серверную часть приложения в рамках модели параллельной обработки запросов.

Вопросы

17. Модель OSI.
18. Модель «Клиент-сервер». Понятие порта. Абстракция сокета.
19. Пакет java.net. Класс Socket. Порядок работы с сокетом клиента.
20. Класс ServerSocket. Сервер параллельной обработки запросов.
21. Дейтаграммы.
22. Uniform Resource Locator.