Лабораторная работа №3

Основы Java для автоматизации тестирования

Цель: Получить базовые сведения о языке Java и его возможностях для последующего использования полученных знаний в целях написания автоматизированных тестов.

План занятия:

1. Изучить теоретические сведения.
2. Выполнить практическое задание.

1. Теоретические сведения

Ознакомиться со сведениями представленные в справочнике для лабораторной №3

Перед началом работы необходимо будет установить следующее:

- Open JDK (https://openjdk.java.net)
- Java IDE

Рекомендуемая IDE - IntelliJ IDEA Community. Но можно воспользоваться любой другой на своё усмотрение.

2. Практические задания

Необходимо реализовать ряд небольших программ на языке Java в соответствии с условиями описанными в каждом задании.

Задание 1

Реализуйте программу, которая берет 2 целых числа из консоли:

- размер массива;
- число для поиска в массиве.

На основе введённых данных должен быть сгенерирован массив случайных целых чисел заданного размера. Программа должна проверить присутствует

ли заданное пользователем число среди элементов полученного массива и вывести результат на экран. Результатом работы программы должен быть вывод сгенерированного массива чисел, а так же строка true либо false, в зависимости от наличия искомого элемента в сгенерированном массиве.

Попробуйте используйте несколько алгоритмов поиска (минимум 2).

Например:

- последовательный поиск;
- бинарный поиск.

Сравните время выполнения для каждого алгоритма

Задание 2

Реализуйте программу, которая генерирует массив целых чисел (аналогично тому как это было сделано в предыдущем задании)и сортирует его, используя как минимум 2 из следующих алгоритмов сортировки:

- сортировка пузырьком;
- сортировка вставками;
- сортировка выбором.

Результатом работы программы должен быть вывод исходного и отсортированного массивов.

Задание 3

Напишите программу, которая запрашивает ввод 1 символа из консоли, подсчитывает количество вхождений этого символа в предварительно созданный текст и выводит его на консоль. Текст может быть жестко закодирован как константная строка в вашем коде или передаваться из консоли.

Задание 4

Ханойская башня:

Есть 3 палочки «A», «B», «C». На палочке «A» находятся «n» (это число нужно взять из консоли) дисков, представленных целыми числами от 1

до n (где 1 - диск с наименьшим диаметром, а n - с наибольшим). Вы должны переместить все диски с палочки «A» на «C», следуя правилу, что каждый диск может быть помещен только на диск большего диаметра (диск с диаметром n может быть только самым нижним диском, диск с диаметром 1 может быть помещен на любой диск)

Напишите программу, которая будет получать количество дисков и напечатает шаги, необходимые для перемещения всех дисков с «A» на палочку «C» с выводом каждого перемещения дисков в консоль.

Например: «1 moved from A to C».

Реализация должна использовать рекурсию.

Задание 5

Реализуйте класс **Author** со следующими характеристиками:

Поля класса:

- firstName
- lastName

Методы:

- Конструктор с возможностью инициализации всех полей класса Author.
- Метод для удобного вывода полного имени автора на экран.

Реализуйте класс **Book** со следующими функциями:

Поля:

- название книги;
- автор;
- цена.

Методы:

- Конструктор с возможностью инициализации всех полей класса Book.

- Метод для удобного вывода полной информации о книге на экран

Для обоих классов должен соблюдаться принцип инкапсуляции.

Напишите отдельный класс **BookDemo** с методом main() который будет инициализировать объект класса Book (например, с названием «Developing Java Software», автором Russel Winder и ценой 79,75) и выводить полные данные о книге в консоль в удобном формате.

Задание 6

Написать класс **Clock** для работы с временем, представленным часами, минутами и секундами. Класс должен включать следующее:

Поля:

- часы (диапазон 0 - 23)
- минуты (диапазон от 0 до 59)
- секунды (диапазон 0 - 59)

Методы:

- конструктор по умолчанию (без переданных параметров; следует инициализировать представленное время до 12: 0: 0)
- конструктор с тремя параметрами: часы, минуты и секунды.
- конструктор с одним параметром: значением времени в секундах после полуночи (его следует преобразовать в значение времени в часах, минутах и секундах)
- **setClock** - с одним параметром: значением времени в секундах после полуночи (его следует преобразовать в значение времени в часах, минутах и секундах)
- **tick** - без параметров, который увеличивает время, сохраненное в объекте Clock, на одну секунду;
- **tickDown**, который уменьшает время, сохраненное в объекте Clock, на одну секунду;
- **addClock** - который принимает объект типа Clock в качестве параметра. Метод должен добавлять время, переданное в метод в объекте Clock, ко

времени, представленному в текущем классе для которого вызывается метод;

- **subtractClock** - который принимает объект типа Clock в качестве параметра и возвращает разницу между временем, представленным в текущем объекте Clock, и временем, переданным в метод в объекте Clock. Разница во времени должна быть возвращена как объект класса Clock;
- **printClock** без параметров который возвращает представление объекта Clock в форме строки формата "(чч: мм: сс)", например "(03:02:34)"

Напишите отдельный класс **ClockDemo** демонстрирующий работу всех указанных выше методов. Обратите особое внимание на корректность работы реализации класса Clock при использовании граничных значений часов\минут\секунд.

Задание 7

Необходимо написать программу реализующую элементарную черепашью графику. Программа должна отрисовывать двумерный массив символов в консоли (доску) и предоставлять возможность рисовать на ней, обрабатывая ввод в формате пары значений: <направление> <расстояние>. Под рисованием мы подразумеваем изменение символов пройденных черепашкой на доске.

Детали примерной реализации:

Создайте класс **TurtleGraphics**, реализующий простую графическую логику черепахи.

Поля:

- **board** - двумерный массив символов, представляющих чертежную доску;
- **penPosition** - поле класса Position для хранения текущей позиции пера (по умолчанию 0,0)
- **cellChar** - символ, представляющий чистую ячейку чистой доски (например, символ '.')
- coloredCellChar - символ, представляющий закрашенную ячейку доски (например, символ 'o')

- **penChar** - символ, представляющий текущую позицию пера на доске (например, символ 'x')

Реализуйте необходимые методы / конструкторы для инициализации платы в соответствии с пользовательским вводом.

Также реализуйте следующие методы:

- **movePen (Direction dir, int length)** - должен переместить перо на доске в указанном направлении, заменяя все посещенные ячейки доски на символы «coloredCellChar»;
- **movePenUp(int length)**
- **movePenDown(int length)**
- **movePenLeft(int length)**
- **movePenRight(int length)**
- **clearBoard()** - должен вернуть доску в исходное состояние

Position - простой класс, содержащий x и y позицию пера в поле.

Направление может задаваться перечислением, представляющим 4 направления: UP, DOWN, RIGHT, LEFT

Методы рисования должны корректно обрабатывать попытки выйти за границы доски во время рисования.

Не стесняйтесь добавлять любые другие методы, которые, по вашему мнению, могут потребоваться.

Реализуйте класс **TurtleGraphicsDemo** позволяющий пользователю рисовать используя **TurtleGraphics** класс.

Например: попросите пользователя указать желаемую ширину / высоту доски и начальную позицию пера. Обрабатывайте ввода данных пользователем в формате «u 2», что означает, что перо следует переместить на 2 позиции вверх (или, например, «r 3», «l 4», «d 1» и т. д.) Перерисовывайте доску после каждого ввода. Пользователь должен так же иметь возможность вернуть доску в исходное состояние.

Пример доски:

```
. . . x . . .

. . . o . . .

. . . o . . .

. . . . . . .
```