Специализированные языки программирования

Задания на лабораторные работы.

Примечание: задания для лабораторных могут быть сформулированы как цепочка последовательных действий. Для выполнения работы *недостаточно* один раз пройтись по этой последовательности. Необходимо уметь комплексно решать задание (например, создание ветки (branch) в локальном репозитории git, внесение модификаций и ее синхронизация с github, слияние с master). Рекомендация - по заданию пройтись минимум 3 раза, с выполнением частей в произвольном порядке, чтобы быть уверенным (-ной) в том, что есть понимание сути выполняемых действий, и возможности их выполнить в нужном для прикладной задачи порядке, без привязки к прямому заданию лабораторной.

По каждой лабораторной необходим печатный отчет (достаточно одного на бригаду, в которой может быть до 3-х человек включительно). Отчет обязательно должен иметь: титульный лист; описание хода работы с исходными кодами (или выполненными командами) и результатом их работы; выводы по работе.

Лабораторная работа №1.
Система управления версиями Git, Web-сервис GitHub

1. Создание собственного репозитория, добавление и удаление файлов оттуда.
2. Модификация файлов в репозитории, просмотр изменений.
3. Проверка текущего статуса, отличий между staging area (буферной зоной), версиями, хранящимися в репозитории.
4. Просмотр списка коммитов в текущей ветке (branch), назначение меток (tag) коммитам.
5. Создание новых веток (branch), переключение между различными коммитами и ветками (branch).
6. Слияние различных веток: без конфликтов и с конфликтами, их разрешение.
7. Подключение удаленного (remote) репозитория на этом же компьютере (в другом каталоге на жестком диске), синхронизация с изменениями, внесенными туда.
8. Создание репозитория на GitHub, синхронизация с изменениями оттуда, внесение изменений из локального репозитория.

Лабораторная работа №2.

1. Объявить переменные всех примитивных типов: без инициализации, инициализацией и с вычислением при инициализации. Распечатать их значения (System.out.println(), System.out.printf()). Целочисленные значения вывести в десятичном и шестнадцатеричном формате.

2. Выполнить приведение целочисленных типов с допустимым расширением диапазона. Выполнить приведение double к float, float к int с некорректным преобразованием (переполнением).
3. Объявить две переменные без инициализации: как поле класса и как локальную переменную в методе. Вывести их на печать и пояснить отличие.
4. Объявить массивы целых значений и строк. Вывести их на печать минимум двумя разными способами.
5. Создать собственное перечисление (enum). Ввести с клавиатуры целое число (Scanner in = new Scanner(System.in); int i = in.nextInt();), сопоставить введенное целое со значением перечисления (использовать switch), значение перечисления распечатать на экране.
6. Объявить константы (по собственному выбору). Сделать консольный мини-калькулятор (перевод дюймов в сантиметры, увеличение скорости в вакууме в зависимости от времени свободного падения и т.п.)
7. Объявить целые, инициализировать их шестнадцатеричными значениями, распечатать, выполнить знаковые и беззнаковые сдвиги, результат тоже распечатать. Пояснить разницу.
8. Ввести целое значение с клавиатуры и проверить допустимость в рамках разрешенного диапазона (например, количества этажей в доме, которые объявлены константой). Напечатать, корректно ли введенное значение.
9. Напечатать множество неотрицательных нечетных чисел максимальной мощности, чье произведение меньше 10 000 (while).
10. Реализовать решето Эратосфена: найти все простые числа от 2 до n (которое ввести с клавиатуры). Для этого формируется ряд чисел от 1 до n. 1 пропускается. 2 тоже пропускается, но вычеркивается каждое второе число (каждое четное). Следующее число (3) пропускается, но затем вычеркивается каждое третье число и т.п.

    После успешного выполнения - оптимизировать алгоритм с точки зрения минимизации вычислительной сложности (использовать факт из теории чисел).

Лабораторная работа №3.

1. При помощи статического метода реализовать пузырьковую сортировку массива целых чисел (длина массива задается при вызове метода сортировки).
2. Задана “неровная матрица” (или ступенчатый массив - двумерный массив, строки в котором имеют различную длину: int [][] arr; arr = new int[5][]; arr[0] = new int[3]; arr[1] = new int[7]; и т.д.). Матрица содержит целые значения. Отсортировать каждую строку (использовать Arrays.sort()), распечатать матрицу на экране. Повторно распечатать матрицу, но выводить на экран только отрицательные значения (использовать цикл с break).
3. Создать собственный класс с полями доступа public и private, проверить их доступность из другого класса текущего пакета.
4. Сделать класс с перегруженными конструкторами (по умолчанию, с заданными параметрами, создание копии другого объекта).
5. Сделать абстрактный класс “автомобиль” и его наследников (“легковой”, “грузовой” и т.п.), продемонстрировать полиморфные методы (вывод названия

автомобиля, перемещение за единицу времени, вывод потребленного топлива, перевезенного груза и пассажиров, загрязнение окружающей среды; предусмотреть возможности для электромобилей).

Лабораторная работа №4.

1. Реализовать собственный класс с несколькими полями и методами, переопределить унаследованный метод toString().
2. Реализовать метод, принимающий пять разнотипных параметров (в т.ч. собственных классов), тремя разными способами: с явным заданием пяти разных параметров, через массив object[] и varargs. Напечатать на экране тип параметра и его значение (для класса - основные поля).
3. Реализовать вложенный класс, статический вложенный класс, продемонстрировать доступность полей и методов внешнего класса.
4. Реализовать локальный класс, проверить доступность полей и методов внешнего класса, локальных переменных метода (изменяемых и нет), который создает локальный класс.
5. Разработать собственный интерфейс шифрования, на его основе интерфейсы симметричного и асимметричного шифрования, на основе симметричного - блочного и поточного шифрования. Создать класс, реализующий интерфейс поточного шифрования (XOR с константой) со статическим методом, на вход принимающим байтовый массив и возвращающий его же на выходе. Задать начальное значение массива, зашифровать и расшифровать, напечатать каждый из результатов.
6. Создать три анонимных класса на основе интерфейса, продемонстрировать полиморфизм. Создать анонимный класс, вызвать его методы без определения локальной переменной.

Лабораторная работа №5.

1. С клавиатуры ввести несколько слов (фраз), объединить их через StringBuilder (каждый раз выводя на печать текущую емкость), распечатать итоговое значение.
2. Перебирая символы строки, распечатать информацию о каждом: является ли цифрой, символом (верхнего или нижнего регистра) или пробелом.
3. Реализовать программу, которая получает два имени текстовых файлов через параметр командной строки, затем выполняет копирование одного файла в другой, без использования системных функций копирования (открывать, читать и записывать данные - самостоятельно, через классы FileInputStream, FileOutputStream, PrintWriter, FileWriter и т.п.). Использовать автозакрытие файлов через try(), обрабатывать ошибки ввода-вывода (печатать про них информацию).
4. Разработать программу, запрашивающую имя файла у пользователя, затем печатающего содержимое файла как шестнадцатеричный дамп на экране. В случае возникновения ошибки (файл не найден, запрет доступа) программа

также печатает сообщение об ошибке. В конце работы программа в любом случае выводит сообщение (любое) о завершении собственной работы.

5. Разработать класс, реализующий интерфейс Closeable. В конструкторе вывести сообщение о создании, в методе close() - о закрытии. Проверить функционирование блока try() с ресурсами.
6. Каскадирование: разработать три собственных класса, реализующих исключения. В блоке try {}, в зависимости от пользовательского ввода, выбросить одно из двух. Перехватывать оба возможных варианта, в обработчике выбросить третье исключение и позже перехватить его. На финальном этапе (перехвате третьего) распечатать стек вызовов каждого из перехваченных исключений.

Лабораторная работа №6.

1. Из текстового файла, содержащего слова (состоящие из латинских букв A-Z в разных регистрах), знаки препинания (,.!?-), пробелы, цифры и спецсимволы (“№;%:*()+) извлечь все слова и записать в выходной файл, разделенные пробелами.
2. Запросить у пользователя логин и пароль (пароль - не отображаемый на экране). Выполнить примитивное хэширование (сложить все символы как 16-битовые целые, умножая предыдущий результат на 5 и добавляя к сумме 7 для каждого символа пароля). Сравнить с заранее заданным в программе значением, и по результату напечатать “доступ предоставлен” или “в доступе отказано”.
3. Сохранить в одном двоичном файле (последовательно): целое число, булево значение, строку с латинскими и кириллическими буквами, два объекта, в полях которых также присутствуют классы (String, Long и т.п.). Из этого файла прочесть значения и распечатать их на экране (для классов - содержимое их полей).
4. Записать в файл массив из 64 байт, закрыть его. Открыть файл на запись, изменить знак каждого байта с четным индексом (0-го, 2-го, 4-го и т.д.), не перезаписывая весь файл, а только конкретные значения, закрыть файл. Открыть файл на чтение, прочесть массив целиком и распечатать его.
5. Запросить у пользователя имя архива, создать такой ZIP-файл в текущем каталоге. Далее итеративно запрашивать имя нового файла в архиве, добавлять данные, введенные пользователем с клавиатуры, в этот файл. Признак того, что ввод текущего файла (внутри архива) завершен - “q!” в самом начале строки. Признак того, что завершен ввод и текущего файла, и всего архива - “Q!” в самом начале строки. Проверить созданных архив (список файлов внутри, и содержание каждого из вложенных файлов) с помощью стандартной программы архивации.
6. С помощью стандартной программы архивации создать ZIP-архив, содержащий файлы (текстовые и двоичные), без каталогов. Написать программу, принимающую имя архива как параметр командной строки (запуск - java программа имя_архива.zip) и извлекающего все файлы в текущий каталог.

Лабораторная работа №7.

1. Определить домашний каталог пользователя, распечатать полный путь и отдельно, начиная с корневого каталога, имя каждого вложенного каталога. Сделать то же самое для текущего каталога (текущий каталог обозначается через "." - одна точка).
2. Создать временный файл для хранения данных, удаляемый при закрытии. Записать туда данные, ожидать пользовательского ввода (System.in.read();), после чего напечатать финальное сообщение и завершить работу программы. Проверить создание файла, его содержимое и автоматическое удаление.
3. Написать утилиту, в командной строке принимающую имя каталога и первые буквы имен файлов/каталогов (опционально), и выводящую содержимое каталога в соответствии с маской. Например, "myls /tmp abc" напечатает список всех файлов и подкаталогов в каталоге /tmp, начинающихся с abc.
4. Разработать утилиту, выполняющую копирование файла, заданного как первый аргумент в командной строке, в каталог/файл, указанного как второй аргумент (указывается полный путь). При необходимости утилита создает нужное дерево подкаталогов. Например, "mycp /home/user/1.txt /tmp/subdir/99" скопирует файл из домашнего каталога в /tmp/, создав еще 2 подкаталога "subdir/99".

Лабораторная работа №8.

1. Создать обобщенный класс (generic), объединяющий три объекта одного класса. Класс объектов, принимаемых на входе, и сам созданный обобщенный класс, реализуют интерфейс Comparable<T>. Метод compareTo созданного класса возвращает среднее арифметическое результатов сравнения всех трех вложенных объектов. Создать два объекта для разработанного обобщенного класса для хранения трех объектов класса Long, распечатать результат сравнения двух объектов между собой.
2. Разработать обобщенный класс MyStack, реализующий функциональность стека (крайний элемент, помещенный в стек, при чтении возвращается первым). Кроме методов push(E elem) и E pop(), помещающих элемент в стек и извлекающих его, реализовать методы копирования уже существующего стека (отдельный метод и через конструктор), выбрасывания исключения при попытке извлечения элемента из пустого стека, проверки количества элементов в стеке, реализация интерфейса Comparable<T> (стек больше, если в нем больше элементов; если одинаково - элементы сравниваются, начиная с крайнего помещенного, результат для стека - compareTo для первого элемента, отличающееся от нуля; в противном случае стеки равны), методы equals,

hashCode и toString(). Если стек реализуется через массив, а не связанный список, предусмотреть метод задания размера для внутреннего хранилища.

3. Расширить реализацию MyStack для возможности использования иерархии классов (в один и тот же объект стека можно поместить и Integer, и Long, и Double). Убрать реализацию интерфейса Comparable<>.
   Разработать методы, перемещающие все элементы:
   a. в текущий из другого стека (этот стек может содержать элементы класса, унаследованного от класса элементов текущего стека);
   b. из текущего - в другой (этот стек может содержать элементы суперкласса относительно класса элементов текущего стека).

   Реализация MyStack<E> должна корректно обрабатывать вызовы:

```
MyStack<Number> stN = new MyStack<>();
MyStack<Integer> stI = new MyStack<>();
MyStack<Double> stD = new MyStack<>();

for(int i = 0; i < 3; i++) {
      stI.push( new Integer( i ) );
      stD.push( new Double( i + 3 ) );
}

System.out.println( "Integer stack: " + stI.toString() );
System.out.println( "Double stack: " + stD.toString() );

stN.addAll( stI );
stD.moveElementsTo( stN );

System.out.println( "Number stack: " + stN.toString() );
```

   Результат работы:

```
Integer stack: [ 0, 1, 2 ]
Double stack: [ 3.0, 4.0, 5.0 ]
Number stack: [ 2, 1, 0, 5.0, 4.0, 3.0 ]
```

   Для реализации необходимо использовать подстановочные типы (wildcards), при этом помнить про принцип PECS (слайды лекций 154-159, материал книги https://cebulko.com/Programming-Resources/Effective%20Java%20-%202nd%20Edition.pdf , стр. 129-141).

   Настоятельно рекомендую решить задачу самостоятельно. И только если возникает крайний случай и полный тупик, посмотреть часть решения здесь: https://github.com/Roman-Oliynykov/JavaLabTaskOnWildcardsSolution.git

Лабораторная работа №9.

1. Загрузить из файла множество слов, вывести все слова (каждое - только один раз) тремя способами: итераторами, for-each, toString() для Set<>. Применять HashSet<>.
2. Из двух файлов сформировать два множества слов, вывести их объединение, пересечение, разность.
3. Загрузить из файла множество слов, загрузить их в ArrayList<>, вывести частоту появления каждого слова (дополнительно использовать Set<> для коллекции уникальных элементов). Пользоваться только методами коллекций (не организовывать собственный подсчет через циклы и т.п.).
4. Загрузить из файла множество слов, отсортировать (повторения допускаются), вывести на экран. Перемешать в случайном порядке, снова вывести.
5. В файле в текстовом формате занесены данные о студентах (формат жестко фиксирован), в каждой строке содержится:

фамилия, группа, название учебной дисциплины (одним словом), успеваемость по дисциплине за семестр.
Фамилии студентов, дисциплин и пр. могут повторяться (студент изучает несколько предметов. Данные находятся в произвольном порядке, например:

Петров КБ-41 java 99
Семенов КС-42 ПЗС 92
Петров КБ-41 КСС 95
Петров КБ-41 КСМ 95
Иванов КУ-33 экономика 80
Семенов КС-42 ТКС 94

Прочесть данные из файла в LinkedList, состоящий из элементов класса Student, содержащего необходимые поля (фамилия, группа, дисциплина, успеваемость). Класс Student реализует интерфейс Comparable<Student> (единственный метод интерфейса - compareTo(Student other); по умолчанию реализуется сортировка по фамилии).
Детальнее про интерфейс:
https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

Вывести: список учебных дисциплин (алф. порядок), список групп (алфавитный порядок).
Реализовать без циклов, используя TreeSet<> с собственным компаратором наподобие

```
Set<Student> courses = new TreeSet<>( new Comparator<Student>() {
                public int compare(Student s1, Student s2) {
                        return s1.course.compareTo( s2.course );
```

```
        }
    });
```

(не забыть добавить в коллекцию сами элементы, одного создания с компаратором - недостаточно).
Детальнее про интерфейс (+ см. пример в лекции):
https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html

Используя цикл с итератором по LinkedList, вывести:

список студентов определенной группы (алфавитный порядок), список студентов определенной группы с оценками по выбранному предмету (в порядке уменьшения балла успеваемости; ввести собственный компаратор через анонимный класс, см. пример выше и пример в лекции).

Лабораторная работа №10.

1. Разработать класс “Товар”, в котором предусмотреть стоимость, срок реализации и уникальный код (буквенно-цифровой, обычно называемый “артикул”). Все поля объекта делать immutable (неизменяемые; они инициализируются только конструктором и внутри класса объявлены как private final). Реализовать очередь на основе ArrayDeque<> (https://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html), в которую внести несколько наименований товара на реализацию. Сформировать еще несколько наименований товара, добавить в начало или в конец очереди в зависимости от срока реализации. Распечатать содержимое очереди, извлекая все элементы, начиная с начала или с конца.
2. Реализовать предыдущее задание на основе PriorityQueue<> (https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html). Приоритет - срок реализации.
3. Для класса “Товар” из задания 1 разработать отображение (Map) артикула в сам товар (поиск товара по артикулу). Создать базу из нескольких товаров, вывести на экран результаты поиска одного товара по артикулу и саму базу.
4. Для баз из двух разных магазинов (на основе задания 3) создать общую. Базы могут содержать коллизионные значения, когда одинаковым артикулам соответствуют разные товары. Проверить на конфликты, сделать общую базу для бесколлизионных артикулов. Для коллизионных создать отдельное мультимножество. Распечатать исходные базы, объединенную базу и мультимножество коллизионных товаров (артикул->ArrayList<Good>).
5. Объединенную базу из предыдущего задания распечатать по возрастанию срока годности товара.

6. Разработать класс “Магазин”, в котором предусмотреть методы: получения полного списка товаров (объектов класса из задания 1) и их количества в наличии (хранение количества реализовать через WeakHashMap); поиск товара по артикулу, вывод их количества в магазине.
   Все поля класса делать исключительно private. Если предоставляется внешний доступ к внутренней коллекции (по полному списку товаров), выдавать их только через немодифицируемые коллекции.

Лабораторная работа №11.

1. Разработать серверное приложение, принимающее текстовые запросы по сети, и реализующее функциональность простого калькулятора.
   По сети сервер принимает запрос, содержащий текстовую строку вида “77/11”, “77 / 11” и т.п. (обрабатываются операции сложения, умножения, деления и вычитания). Выход - по команде “exit”, в случае деления на 0 или переполнения - вывод соответствующего сообщения.
   Сетевой ввод-вывод дублируется на локальную консоль сервера (System.out), ошибки дублируются - на System.err.
   Проверка функциональности: подключиться через telnet на соответствующий порт (telnet IP порт, например, telnet localhost 9876), ввести данные, увидеть результат на экране.
2. Сделать серверное приложение многопоточным, когда на каждый клиентский запрос создается отдельный поток исполнения и каждый клиент работает независимо.
3. Разработать клиента для серверного калькулятора, который выполняет 5 запросов на расчеты через случайные интервалы времени (длиной от 3 до 7 секунд). Данные на расчетов тоже генерируются случайно (запрос с делением на 0 допускается). Данные, отправляемые и принимаемые по сети, дублируются на локальную консоль клиента (System.out).
4. Создать java-приложение, параллельно запускающее 3 независимых клиента для серверного калькулятора (клиенты работают в рамках одного приложения).
5. Разработать java-приложение, выполяющее подключение к Twitter и выполяющее:
   - поиск по ключевому слову;
   - вывод списка сообщений пользователя (timeline).

   Пример приложения с аутентификацией и отправкой сообщений в Twitter есть в лекциях.
   Библиотека: twitter4j (http://twitter4j.org/)

Примеры кода (есть ссылка с основной страницы): http://twitter4j.org/en/code-examples.html

Регистрация собственного приложения Twitter: https://apps.twitter.com/

Лабораторная работа №12.

1. Разработать java-приложение, в качестве аргумента командной строки принимающее URL web-страницы и сохраняющее ее в файл. При проверке работы необходимо задать http:// и https:// адреса, а сохраненный файл открыть браузером с локального диска и проверить соответствие (для страниц с простой разметкой).
2. Создать java-приложение, сохраняющее в текстовый файл курс какой-либо иностранной валюты к гривне за 2 года (одно значение в месяц). Информацию загружать с сайта Приватбанка: https://api.privatbank.ua/archiverate.html. Данные загрузить в Google tables / LibreOffice calc / Microsoft Excel и построить диаграмму.
3. При наличии собственной карты Приватбанка и регистрации в Приват24:
    a. заказать выпуск виртуальной карты для платежей в Интернет (если она еще не выпущена);
    b. разработать приложение, выполняющее пополнение мобильного телефона через API https://api.privatbank.ua/directfill.html
    c. проверить работу программы в тестовом режиме.
    d. **из программы удалить ID мерчанта и пароль** (если они были прописаны в коде); помнить, что этой информации достаточно для снятия/перевода денег с карты; **отчеты, в которых эти данные все-таки будут присутствовать, не принимаются,** без возможности перезащиты.
4. При отсутствии доступа к Приват24 (если нет возможности выполнить задание №3): загрузить библиотеку Facebook4j (http://facebook4j.org/en/index.html), зарегистрироваться как разработчику facebook (https://developers.facebook.com/apps) и на основе примеров (http://facebook4j.org/en/code-examples.html) сделать java-приложение, публикующее какое-либо изображение с комментарием.