|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| ***Заданий для практических занятий.*** | | | | |
| **Задание на практическую работу №17**  1. Реализовать запись в файл введённой с клавиатуры информации 2. Реализовать вывод информации из файла на экран 3. Заменить информацию в файле на информацию, введённую с клавиатуры 4. Добавить в конец исходного файла текст,  **Задание на практическую работу №18**  1.Необходимо инстанцировать класс и выполнить exceptionDemo (). Программа даст сбой, и вы получите следующее сообщение java.lang.ArithmeticException: / by zero at Exception1.exceptionDemo(Exception1.java:12)  Это говорит нам о том, что программа пытается выполнить деление на ноль, который он не в состоянии выполнить. Выполните следующее:  1.1 Замените 2/0 на 2,0 / 0,0 и повторно вызовите метод. Объясните, что произойдет  1.2 Измените код в листинге 1 на следующий: public class Exception2 {   public void exceptionDemo() {   Scanner myScanner = new Scanner( System.in );   System.out.print( "Enter an integer ");  String intString = myScanner.next();  int i = Integer.parseInt(intString); System.out.println( 2 / i );   }  } 2. С помощью перехватывания исключений можно оказывать влияние на поведение программы. В вашем решении в предыдущем упражнении вы можете добавить новый пункт - catch в начале списка пунктов catch. Выполите это, чтобы поймать общее исключение класса Exception.  3.Перезапустите программу с приведенными выше данными и обратите внимание на ее поведение. Объясните новое поведение программы 4.И наконец добавьте блок finally к решению. Повторно запустите программу, чтобы проверить ее поведение. Объясните новое поведение программы  **Задание на практическую работу №19**  1. Написать программную реализацию для онлайн покупок в зарубежном интернет-магазине. Клиент совершает онлайн покупку. При оформлении заказа у пользователя запрашивается фио и номер ИНН. В программе проверяется, действителен ли номер ИНН для такого клиента. Исключение будет выдано в том случае, если введен недействительный ИНН. 2. Предлагается модернизировать задачу из практической работы №13 – задача сортировки студентов по среднему баллу. Необходимо разработать пользовательский интерфейс для задачи поиска и сортировки (использовать массив интерфейсных ссылок- пример в лекции 5. Дополнить ее поиском студента по фио – в случае отсутствия такого студента необходимо выдавать собственное исключение.  **Задание на практическую работу №20**  1. Написать метод для конвертации массива строк/чисел в список.  2. Написать класс, который умеет хранить в себе массив любых типов данных (int, long etc.). 3. Реализовать метод, который возвращает любой элемент массива по индексу. 4. Написать функцию, которая сохранит содержимое каталога в список и выведет первые 5 элементов на экран. 5. \*Реализуйте вспомогательные методы в классе Solution, которые должны создавать соответствующую коллекцию и помещать туда переданные объекты. Методы newArrayList, newHashSet параметризируйте общим типом T. Метод newHashMap параметризируйте парой <K, V>, то есть типами К- ключ и V-значение. Аргументы метода newHashMap должны принимать  **Задание на практическую работу №21**  1. Реализовать класс Абстрактная фабрика для комплексных чисел  2. Реализовать класс Абстрактная фабрика для различных типов стульев: Викторианский стул, Многофункциональный стул, Магический стул, а также интерфейс Стул, от которого наследуются все классы стульев, и класс Клиент, который использует интерфейс стул в своем методе Sit (Chair chair).  3. Вводная. Компании нужно написать редактор текста, редактор изображений и редактор музыки. В трёх приложениях будет очень много общего: главное окно, панель инструментов, команды меню будут весьма схожими. Чтобы не писать повторяющуюся основу трижды, вам поручили разработать основу (каркас) приложения, которую можно использовать во всех трёх редакторах. На совещании в компании была принята следующая архитектура: Исходные данные. Есть некий базовый интерфейс IDocument, представляющий документ неопределённого рода. От него впоследствии будут унаследованы конкретные документы: TextDocument, ImageDocument, MusicDocument и т.п. Интерфейс IDocument перечисляет общие свойства и операции для всех документов. • При нажатии пунктов меню File -> New и File -> Open требуется создать новый экземпляр документа (конкретного подкласса). Однако каркас не должен быть привязан ни к какому конкретному виду документов. • Нужно создать фабричный интерфейс ICreateDocument. Этот интерфейс содержит два абстрактных фабричных метода: CreateNew и CreateOpen, оба возвращают экземпляр IDocument • Каркас оперирует одним экземпляром IDocument и одним экземпляром ICreateDocument. Какие конкретные классы будут подставлены сюда, определяется во время запуска приложения. Требуется:  1. создать перечисленные классы. Создать каркас приложения — окно редактора с меню File. В меню File реализовать пункты New, Open, Save, Exit. 2. продемонстрировать работу каркаса на примере текстового редактора. Потребуется создать конкретный унаследованный класс TextDocument и фабрику для него — CreateTextDocument. В данной практической работе рекомендуется использовать следующие паттерны: Абстрактная фабрика и фабричный метод.  **Задание на практическую работу №22**  1.Доработайте класс Student предусмотрите поле для хранения даты рождения, перепишите метод toString() таким образом, чтобы он разработайте метод, возвращал строковое представление даты рождения по вводимому в метод формату даты (например, короткий, средний и полный формат даты). 2. Напишите пользовательский код, который формирует объекты Date и Calendar по следующим данным, вводимым пользователем: <Год><Месяц><Число> <Часы1><минуты> 3Написать программу, выводящую фамилию разработчика, дату и время получения задания, а также дату и время сдачи задания. Для получения последней даты и времени использовать класс Date из пакета java.util.\* (Объявление Dated=newDate() или метод System.currentTimeMillis().  **Задание на практическую работу №23**  1. Создать словарь (Map<String, String>), записать в него десять записей по принципу «фамилия» - «имя». Удалить людей, имеющих одинаковые имена. Вывести словарь на печать 2. Описать класс City, с полями: город, страна. Записать исходные данные в HashMap, где ключ - страна, значение - город. Map<String,ArrayList<String>> map = new HashMap(); Записать необходимо таким образом, чтобы каждому ключу(стране) соответствовал список значений(городов).  **Задание на практическую работу №24**  1. Разработать класс Person, в котором имеется функция, возвращающая Фамилию И.О. Функция должна учитывать возможность отсутствия значений в полях Имя и Отчество. Программу оптимизировать с точки зрения быстродействия. 2. Доработать класс адреса, который из полученной строки формата "Страна[d] Регион[d] Город[d] Улица[d] Дом[d] Корпус[d] Квартира" ([d] – разделитель, например, «запятая») выбирает соответствующие части и записывает их в соответствующие поля класса Address. Учесть, что в начале и конце разобранной части адреса не должно быть пробелов. Все поля адреса строковые. Разработать проверочный класс не менее чем на четыре адресных строки. В программе предусмотреть две реализации этого метода: а) разделитель – только запятая (использовать метод split()); Внимание, при разработке нужно учесть, что  б) разделитель – любой из символов ,.;- (класс StringTokenizer). 3. Реализуйте класс Shirt: Метод toString() выводит объяснение и значение полей построчно. Дан также строковый массив: shirts[0] = "S001,Black Polo Shirt,Black,XL"; shirts[1] = "S002,Black Polo Shirt,Black,L"; shirts[2] = "S003,Blue Polo Shirt,Blue,XL"; shirts[3] = "S004,Blue Polo Shirt,Blue,M"; shirts[4] = "S005,Tan Polo Shirt,Tan,XL"; shirts[5] = "S006,Black T-Shirt,Black,XL"; shirts[6] = "S007,White T-Shirt,White,XL"; shirts[7] = "S008,White T-Shirt,White,L"; shirts[8] = "S009,Green T-Shirt,Green,S"; shirts[9] = "S010,Orange T-Shirt,Orange,S"; shirts[10] = "S011,Maroon Polo Shirt,Maroon,S"; Преобразуйте строковый массив в массив класса Shirt и выведите его на консоль. 4. Разработайте класс, который получает строковое представление телефонного номера в одном из двух возможных строковых форматов: +<Код страны><Номер 10 цифр>, например “+79175655655” или “+104289652211” или 8<Номер 10 цифр> для России, например “89175655655” и преобразует полученную строку в формат: +<Код страны><Три цифры>–<Три цифры>–<Четыре цифры> 5. В методе main считай с консоли имя файла, который содержит слова, разделенные пробелом. В методе getLine() используя StringBuilder расставьте все слова в таком порядке, чтобы последняя буква данного слова совпадала с первой буквой следующего не учитывая регистр. Каждое слово должно участвовать 1 раз.  **Задание на практическую работу №25**  1.Ознакомиться с классами Pattern, Matcher и PatternSyntaxException  2. Выбрать IPv4 адреса во всех возможных, представлениях: десятичном, шестнадцатеричном и восьмеричном. С точками и без. Подробнее про IP адреса можно узнать в википедии  3. Составить регулярное выражение, определяющее является ли заданная строка IP адресом, записанным в десятичном виде. – пример правильных выражений: 127.0.0.1, 255.255.255.0. – пример неправильных выражений: 1300.6.7.8, abc.def.gha.bcd.  4. Создать запрос для вывода только правильно написанных выражений со  скобками (количество открытых и закрытых скобок должно быть одинаково). – пример правильных выражений: (3\*+\*5)\*–\*9\*×\*4. – пример неправильных выражений: ((3\*+\*5)\*–\*9\*×\*4.  **Задание на практическую работу №26**  1. Реализовать очередь на массиве • Найдите инвариант структуры данных «очередь». Определите функции, которые необходимы для реализации очереди. Найдите их пред- и постусловия. • Реализуйте классы, представляющие циклическую очередь с применением массива. В результате выполнения задания должны быть реализованы следующие классы: класс ArrayQueueModule должен реализовывать один экземпляр очереди с использованием переменных класса, класс ArrayQueueADT должен реализовывать очередь в виде абстрактного типа данных (с явной передачей ссылки на экземпляр очереди), класс ArrayQueue должен реализовывать очередь в виде класса (с неявной передачей ссылки на экземпляр очереди)  В задании у классов должны быть реализованы следующие функции(процедуры) / методы:  enqueue – добавить элемент в очередь;  element – первый элемент в очереди;  dequeue – удалить и вернуть первый элемент в очереди;  size – текущий размер очереди;  isEmpty – является ли очередь пустой;  clear – удалить все элементы из очереди. Замечания: инвариант, пред- и постусловия записываются в исходном коде в виде комментариев. Также обратите внимание на инкапсуляцию данных и кода во всех трех реализациях. Напишите тесты реализованным классам.  2. Реализация структуры данных Очередь на связном списке  Для выполнения задания необходимо: -Определить интерфейс очереди Queue и описать его контракт. -Реализовать класс LinkedQueue — очередь на связном списке. -Выделить общие части классов LinkedQueue и ArrayQueue в базовый класс AbstractQueue. Напишите тесты реализованным классам. Дополнительные задания 3. Вычисление выражений 3.1. Разработайте классы Const, Variable, Add, Subtract, Multiply, Divide для вычисления выражений с одной переменной. 3.2 Классы должны позволять составлять выражения вида  new Subtract(  new Multiply(  new Const(2),  new Variable("x")  ),  new Const(3)  ).evaluate(5) При вычислении такого выражения вместо каждой переменной подставляется значение, переданное в качестве параметра методу evaluate (на данном этапе имена переменных игнорируются). Таким образом, результатом вычисления приведенного примера должно стать число 7. 3.3. Для тестирования программы должен быть создан класс Main, который вычисляет значение выражения x2−2x+1, для x, заданного в командной строке. 3.4. При выполнение задания следует обратить внимание на: • Выделение общего интерфейса создаваемых классов. • Выделение абстрактного базового класса для бинарных операций. Задание 4  4.1. Доработайте предыдущее задание, так что бы выражение строилось по записи вида   x \* (y - 2)\*z + 1 Для этого реализуйте класс ExpressionParser с методом TripleExpression parse(String). 4.2. В записи выражения могут встречаться: умножение \*, деление /, сложение +, вычитание -, унарный минус -, целочисленные константы (в десятичной системе счисления, которые помещаются в 32-битный знаковый целочисленный тип), круглые скобки, переменные (x, y и z) и произвольное число пробельных символов в любом месте (но не внутри констант). 4.3. Приоритет операторов, начиная с наивысшего • унарный минус; • умножение и деление; • сложение и вычитание. Пример для выражения 1000000\*x\*x\*x\*x\*x/(x-1) вывод программы должен иметь следующий вид:   x f  0 0  1 division by zero  2 32000000  3 121500000  4 341333333  5 overflow  **Задание на практическую работу №27**  1.Реализовать в виде паттерна MVC  Написать калькулятор для чисел с использованием RPN (Reverse Polish Notation в пер. на русск. яз. - обратной польской записи)  2. Напишите программу-калькулятор арифметических выражений, записанных в обратной польской нотации (RPN-калькулятор).  Постройте систему тестов и проверьте, что ваш калькулятор успешно проходит все тесты и «защищён от дурака» (как дурака-пользователя программы, так и дурака-программиста, использующего ваш стек, и калькулятор). Например, если вводится выражение, в котором число операций превосходит число помещенных в стек элементов (например, 1 2 + \*), то программа не допустит уменьшения переменной sp до отрицательных значений, а выдаст предупреждение «Невозможно выполнить POP для пустого стека». Работу можно выполнить в двух вариантах (1вариант проще, второй-труднее)  3. Реализовать игру "змейка" в виде GUI c использованием подходящих паттернов и модели MVC. Программа должна быть декомпозирована на соответствующие классы: BodyPos, GameController, GameModel GameView, SnakePanel, Main  **Задание на практическую работу №28**  Написать программу реализующую оплату в интернет-магазине клиентом покупок. Использование паттерна Стратегия должно реализовать выбор платёжного метода в интернет-магазине. Когда пользователь сформировал заказ, он получает выбор из нескольких платёжных средств: электронного кошелька или кредитной карты. В данном случае конкретные стратегии платёжных методов не только проводят саму оплату, но и собирают необходимые данные на форме заказа.  **Задание на практическую работу №29**  В процессе написания тестовых заданий ознакомьтесь с принципами создания динамических структур в Java, механизмом исключений и концепцией интерфейсов.  Замечание: в процессе выполнения задания НЕЛЬЗЯ пользоваться утилитными классами Java (за исключением java.util.HashMap).  1.Создайте класс Drink – напитка. Класс описывает сущность – напиток и характеризуется следующими свойствами - стоимостью, названием и описанием. Класс должен быть определен как неизменяемый (Immutable class). Создайте класс Dish , описывает напитки 2. Создайте интерфейс Item – для работы с позициями заказа. Интерфейс определяет 3 метода: возвращает стоимость, возвращает название., возвращает описание позиции. Классы Drink и Dish, которые должны реализовывать этот интерфейс. 3. Создайте класс InternetOrder, который моделирует сущность интернет заказ в ресторане или кафе. Класс основан на циклическом двусвязном списке с выделенной головой и может хранить как блюда, так и напитки. Внимание: список реализуется самостоятельно!  Конструкторы: не принимающий параметров (для списка создается только головной элемент, сам список пуст), принимающий массив позиций заказа (создаем список из N позиций). Класс InternetOrder моделирует сущность ""нтернет-заказ"" в ресторане или кафе. Класс основан на циклическом двусвязном списке с выделенной головой и может хранить как блюда, так и напитки. Внимание: список вы реализуется самостоятельно! Класс Dish сделайте неизменяемым (аналогично Drink). Класс Order должен хранить (удалять и добавлять) не только экземпляры класса Dish, но и Drink (для этого разработайте классы Order и TablesOrderManager). Методы: −добавляющий позицию в заказ (принимает ссылку типа Item). Пока этот метод возвращает истину после выполнения операции добавления элемента.  4. Переименуйте класс Order из предыдущего задания в RestaurantOrder. Создайте интерфейс Order – позиции заказа.  Интерфейс должен определять следующие методы:  −добавления позиции в заказ (принимает ссылку типа Item), при этом возвращает логическое значение.  −удаляет позицию из заказа по его названию (принимает название блюда или напитка в качестве параметра). Возвращает логическое значение.  −удаляет все позиции с заданным именем (принимает название в качестве параметра). Возвращает число удаленных элементов.  −возвращает общее число позиций заказа в заказе.  −возвращает массив позиций заказа.  −возвращает общую стоимость заказа.  −возвращает число заказанных блюд или напитков (принимает название в качестве параметра).  −возвращает массив названий заказанных блюд и напитков (без повторов).  −возвращает массив позиций заказа, отсортированный по убыванию цены.  Замечание: Классы InternetOrder и RestaurantOrder должны реализовывать интерфейс Order. 5. Переименуйте класс TablesOrderManager в OrderManager. Добавьте ему еще одно поле типа java.util.HasMap<String, Order>, которое содержит пары адрес-заказ, и методы (работающие с этим полем):  Методы класса: −перегрузка метода добавления заказа. В качестве параметров принимает строку – адрес и ссылку на заказ.  −перегрузка метода получения заказа. В качестве параметра принимает строку – адрес.  −перегрузка метода удаления заказа. В качестве параметра принимает строку – адрес заказа. −перегрузка метода добавления позиции к заказу. В качестве параметра принимает адрес и Item.  −возвращающий массив имеющихся на данный момент интернет-заказов.  −возвращающий суммарную сумму имеющихся на данный момент интернет-заказов.  −возвращающий общее среди всех интернет-заказов количество заказанных порций заданного блюда по его имени. Принимает имя блюда в качестве параметра. Методы должны работать с интерфейсными ссылками Order и Item.  6.Создайте пользовательское исключение OrderAlreadyAddedException, выбрасываемое при попытке добавить заказ столику или по адресу, если со столиком или адресатом уже связан такой заказ.  Конструктор классов Drink и Dish должен выбрасывать исключение java.lang.IllegalArgumentException при попытке создать блюдо или напиток со стоимостью меньше 0, без имени или описания (если параметры имя и описание - пустые строки).  Создайте не объявляемое исключение IllegalTableNumber, выбрасываемое в методах, принимающих номер столика в качестве параметра, если столика с таким номером не существует.  **Задание на практическую работу №30**  1. Разработка кода по UML диаграмме с использованием классов, разработанных в практической работе №30.  2. Разработка интерфейса пользователя для интерактивного взаимодействия.    **Задание на практическую работу №31**  1. Использовать Maven для подключения модуля JUnit  2.Задание на виды тестирования  2.1 Модульное тестирование: − В среде IDE IntelliJ IDEA открыть исходный код тестируемого приложения (собственное, или из предыдущей практической работы)  − Добавить Unit-тест для одной из функций. − Запустить тест и просмотреть результаты. − Создать несколько разных тестов для проверки значений и перехвата исключений.  2.2 Покрытие кода тестами: − Установить параметры сбора статистики покрытия кода. − Повторить модульные тесты и просмотреть данные о покрытии кода.  2.3 Нагрузочное тестирование: − Для тестируемого приложения (собственное или из предыдущей практической работы ) провести профайлинг (оценку производительности). − Выполнить тест и просмотреть результаты.  **Задание на практическую работу №32**  Задание на сериализацию. Для задания из практической работы №30 обеспечить сохранение объектов в файл и чтение из файла | | | | |