ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ПРОЕКТА

1. На основе сущностей предметной области создать классы их описывающие.
2. Типы данных должны соответствовать их сущности (например, для описания даты использовать тип Date)
3. Классы и методы должны иметь отражающую их функциональность названия.
4. Для хранения списков или наборов классов, использовать классы-коллекции.
5. Использовать возможности ООП: классы, наследование, полиморфизм, инкапсуляция.
6. Каждый класс должен иметь исчерпывающее смысл название и информативный состав.
7. Наследование должно применяться только тогда, когда это имеет смысл.
8. При кодировании должны быть использованы соглашения об оформлении кода java code convention.
9. Классы должны быть грамотно разложены по пакетам.
10. Для хранения параметров инициализации использовать файлы (Если нет требования консольного ввода).
11. Предусмотреть корректную обработку неправильных действий пользователя.
12. Везде, где это возможно, использовать лямбда-выражения и Stream-API.
13. Создание собственных классов-исключений приветствуется.
14. **Система** «**Вектора**». В отдельном модуле cоздать класс для определения одномерных массивов целых чисел (векторов). Предусмотреть возможность обращения к отдельному элементу массива с контролем выхода за пределы индексов, возможность задания произвольных границ индексов при создании объекта и выполнения операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов, умножения и деления всех элементов массива на скаляр, печати (вывода на экран) элементов массива по индексам и всего массива. При необходимости в класс можно добавлять поля, методы либо свойства. Создать форму для работы с классом. Предусмотреть проверку корректности обрабатываемых данных и операций над ними.
15. **Система** **Фургон кофе.** Загрузить фургон определенного объема грузом на определенную сумму из различных сортов кофе, находящихся в тому же в разных физических состояниях (зерно, молотый, растворимый в банках и пакетиках) (задаётся в текстовом файле). Сумма груза и объём фургона задаётся пользователем. Учитывать объем кофе вместе с упаковкой. Если по заданным параметрам загрузить фургон невозможно, попросить пользователя ввести другие параметры. Провести сортировку товаров на основе соотношения цены и веса. Найти товар в фургоне по его названию или физическому состоянию (задаётся пользователем). Вывести количество данного товара.
16. **Система** **Цветочница.** Создать класс **Цветок**. Создать объекты-цветы разного наименования и стоимости. Собрать букет из цветов на определённую сумму. Входные данные: стоимость букета (целое), выходные данные: букет - список цветов, с суммарной стоимостью, равной заданной. Цветы в букете могут повторяться. Если на заданную сумму подобрать букет невозможно, сообщить об этом пользователю и попросить ввести другую сумму. Провести сортировку цветов в букете по их стоимости; по их названиям (тип сортировки вводится пользователем). Найти количество цветов заданного типа в букете (розы, гвоздики и пр. – вводится пользователем).
17. **Система** **Новогодний подарок.** Определить класс **Конфета**. Создать несколько объектов-конфет. Собрать детский подарок определённого веса. Входные данные: вес подарка (целое), выходные данные: подарок - список конфет, с суммарным весом, равным заданному. Конфеты в подарке могут повторяться. Если на заданный вес подобрать подарок невозможно, сообщить об этом пользователю и попросить ввести другой вес. Провести сортировку конфет в подарке на основе общего веса конфет каждого типа; на основе количества конфет каждого типа. Найти конфеты в подарке, соответствующие заданному диапазону содержания сахара.
18. **Система** **Домашние электроприборы.** Определить иерархию электроприборов. Включить некоторые в розетку. Посчитать потребляемую мощность. Провести сортировку приборов в квартире на основе мощности; на основе их наименования. Найти приборы в квартире, соответствующие заданному диапазону параметров.
19. **Система Звукозапись.** Определить иерархию музыкальных композиций. Записать на диск сборку. Посчитать продолжительность. Провести перестановку композиций диска на основе принадлежности к стилю; на основе их продолжительности. Найти композиции, соответствующие заданному диапазону длины треков.
20. **Система драгоценные Камни.** Класс **Драгоценный камень**. Отобрать камни для ожерелья заданного веса (в каратах). Входные данные: вес ожерелья (целое), выходные данные: ожерелье - список камней, с суммарным весом, равным заданному. Камни в ожерелье могут повторяться. Если на заданный вес подобрать ожерелье невозможно, сообщить об этом пользователю и попросить ввести другой вес. Посчитать общую стоимость ожерелья. Провести сортировку камней ожерелья на основе ценности; на основе веса. Найти камни в ожерелье, соответствующие заданному диапазону параметров прозрачности.
21. **Система странствующий Рыцарь.** Определить иерархию амуниции рыцаря. Создать объекты – элементы амуниции разной стоимости (различные кирасы, шлемы, наколенники и пр.). Экипировать рыцаря на заданную стоимость. Входные данные: стоимость полной экипировки (целое), выходные данные: комплект брони с суммарной стоимостью, равной заданной. Учитывать, что некоторые элементы экипировки – парные. Если на заданную сумму подобрать броню невозможно, сообщить об этом пользователю и попросить ввести другую сумму. Провести сортировку амуниции на основе веса; на основе стоимости. Найти элементы амуниции, соответствующие заданному диапазону параметров цены.
22. **Система УПРАВЛЕНИЕ АВТОБУСНЫМ ПАРКОМ**

Вся информация в приложении хранится в трех файлах:

автобусы (№ автобуса; марка; количество мест; водитель)

маршруты (№ маршрута; пункт отправления; пункт назначения; время отправления; время прибытия)

рейсы (идентификатор рейса; № автобуса; № маршрута; дата рейса)

Приложение должно поддерживать следующие возможности:

* добавление автобуса
* изменение водителя на автобусе (если он не назначен на другой автобус)
* удаление автобуса (если автобус не назначен ни на один рейс)
* добавление маршрута
* изменение времени отправления и прибытия маршрута
* удаление маршрута (если по нему нет ни одного рейса)
* добавление рейса
* удаление рейса
* поиск рейсов по водителю, пункту назначения и пункту отправления

1. **Система** «Геометрические **фигуры**».

Создать:

Класс "Точка", имеющий 2 координаты.

Реализовать:

- конструктор для инициализации координат;

- процедуру изменения координат точки на величину step

Класс "Отрезок", наследующий от класса "Точка". Реализовать конструкторы:

- с параметрами 2 точки,

- с параметрами 2 пары координат.

Реализовать:

- функцию вычисления длины отрезка

- процедуру изменения начала и конца отрезка на величину step

Класс "Прямоугольник", наследующий от класса "Отрезок". Реализовать конструкторы:

- с параметрами 2 точки,

- с параметрами 2 пары координат.

- с параметрами 2 отрезка,

Реализовать:

- функцию вычисления площади прямоугольника;

- функцию вычисления периметра прямоугольника;

- процедуру перемещения всех вершин прямоугольника на величину step

Создать класс "Круг" , наследующий от класса "Отрезок". Реализовать конструкторы:

- с параметром отрезок (радиус)

- с параметрами 2 точки: центр и точка на границе круга

- с параметрами 2 пары координат: координаты центра и точки на границе круга

Реализовать:

- функцию вычисления площади круга;

- функцию вычисления периметра круга;

- процедуру перемещения центра круга на величину step

Класс "Эллипс" , наследующий от класса "Круг". Реализовать конструкторы:

- с параметрами 2 отрезка (радиусы)

- с параметрами 3 точки: центр и точки на границе эллипса (концы малого и большого радиусов)

- с параметрами 3 пары координат: координаты центра и точек на границе эллипса (концов малого и большого радиусов)

Реализовать:

- функцию вычисления площади эллипса (площадь эллипса: Pi\*R\*r);

- функцию вычисления периметра эллипса (периметр эллипса: (4\*Pi\*D\*d+(D-d)^2) / (D + d) );

- процедуру перемещения центра эллипса на величину step

- возможность добавления фигур в список с сортировкой по площади, периметру (задаётся пользователем).

При необходимости в систему можно добавлять классы, поля, методы либо свойства. Предусмотреть проверку корректности обрабатываемых данных и операций над ними.

1. **Система** «**квадратное уравнение**». Создать класс «Квадратное уравнение». Поля класса – коэффициенты уравнения, корни уравнения; методы класса – расчет корней. Предусмотреть конструктор для инициализации полей класса. В конструкторе аналитически определить отрезки, на котором существуют корни, в методах реализовать получение корней методами дихотомии, Ньютона, простых итераций. Если корней не существует, после создания объекта класса должно отображаться соответствующее сообщение. При необходимости в класс можно добавлять поля, методы либо свойства. Предусмотреть проверку корректности обрабатываемых данных и операций над ними.
2. **Система** **калькулятор** **для обработки математических операций** с вещественными числами ("-", "+", "/", "\*", "(", ")", "0"-"9",".", «корень», «степень»). Определить абстрактный класс «Operation». Каждую математическую операцию реализовать, как наследник класса Operation. Пользователь вводит математическое выражение с клавиатуры.
3. **Система** **калькулятор** **для обработки математических операций с матрицами** вещественных чисел (сложение, умножение, умножение на число, транспонирование, вычисление определителя). Пользователь вводит с клавиатуры одну или две матрицы, в зависимости от требуемой операции, и саму операцию.
4. **Система** **Игра «крестики-нолики»**. Пользователь вводит с клавиатуры координаты своего хода, программа выводит координаты своего и (или) сообщение о выиграше/проиграше. Предусмотреть возможность выбора, кто будет делать первый ход.
5. **Система меню столовой.** В меню включаются не менее 5 классов продуктов (задаются в текстовом файле). Реализовать для пользователя возможность выбора продуктов на завтрак. Реализовать процедуру подсчёта в завтраке продуктов заданного класса (класс вводится с клавиатуры). Реализовать интерфейс Nutritious (Питательный), содержащий метод calculateCalories(). Каждый из классов продуктов должен его реализовать.

Реализовать обработку специальных параметров (начинающихся с дефиса):

-calories вычислить и напечатать общую калорийность завтрака;

-sortByNameUp отсортировать продукты в завтраке по возрастанию (использовать название продуктов);

-sortByNameDown отсортировать продукты в завтраке по убыванию (использовать название продуктов);

-sortByCountUp отсортировать продукты в завтраке по возрастанию (использовать количество продуктов);

-sortByCountDown отсортировать продукты в завтраке по убыванию (использовать количество продуктов);

-sortByCaloryUp отсортировать продукты в завтраке по возрастанию (использовать общую калорийность продуктов);

-sortByCaloryDown отсортировать продукты в завтраке по убыванию (использовать общую калорийность продуктов).

1. **Система пункт проката видеодисков**

Вся информация в приложении хранится в трех файлах:

– пользователи проката (фамилия; имя; отчество; адрес; телефон);

– видеодиски (идентификатор диска; название; жанр; год выпуска; режиссер);

– прокат (идентификатор проката; телефон пользователя; идентификатор диска; дата выдачи; предполагаемая дата возврата; возвращен ли диск).

Приложение должно поддерживать следующие возможности:

– добавление пользователя

– изменение информации о пользователе

– удаление пользователя (если у него нет невозвращенных дисков и нет просроченных дисков)

– добавление диска

– удаление диска (если он не в прокате)

– выдача диска в прокат (если диск доступен и у пользователя нет просроченных дисков)

– возврат диска

– поиск клиента по фамилии и телефону

– поиск диска по названию, жанру, году выпуска, режиссеру (задаётся пользователем)