Задача 1. Основы ОПП (инкапсуляция)

Задача, прежде всего на инкапсуляции, но местами уже наследование всплывает.

Для задач, которые не предполагают визуализацию, должен быть написан код, демонстрирующий работу с классом и показывающий, что ваш код рабочий.

Если в условии задачи что-то непонятно – попросить пояснить преподавателя.

Варианты:

1. Реализовать класс для представления чисел в произвольной системе счисления.

2. Реализовать свой собственный упрощенный аналог класса BigDecimal. Представлять большое число можно в виде массива/списка цифр. Реализовать функции сложения, вычитания, умножения, целочисленного деления и остатка от деления.

3. Класс для описания комплексных чисел. Реализовать основные арифметические операции для комплексных чисел. Дополнительно реализовать аналог класса Math для комплексных чисел с несколькими тригонометрическими функциями (в реализации можно использовать стандартный класс Math).

4. Реализовать класс с алгоритмом псевдослучайных чисел (существующий Random не использовать).

5. Реализовать класс Дата (операции +/- n дней/месяцев/лет, сравнение дат, форматирование даты по шаблону, разбор даты из строки и т.п.). Дата внутри хранится в виде целого числа - кол-во дней с начала летоисчисления (01.01.0001).

6. Реализовать класс Время (операции - аналогичные дате). Время внутри хранится в виде целого числа – кол-ва секунд с 00:00

7. Реализовать класс Дата + Время (реализация должна быть выполнена на основе 2-х предыдущих классов).

8. Реализовать класс Треугольник. Треугольник задается 3-мя точками (для точки – отдельный класс). Реализовать методы нахождения площади, периметра, проверки принадлежности точки треугольнику.

9. Реализовать класс студент с возможность хранения оценок по предметам за каждый семестр. Реализовать методы вычисления средней оценки за семестр и за все время обучения.

10. Реализовать класс, в котором хранятся 2 списка и реализованы методы вычисления:
    - элементов, которые есть в 2-х списках одновременно;
    - элементов, которые есть только в первом списке;
    - элементов, которые есть только во втором списке.

11. Описание интерфейса класса. Должна быть предусмотрена генерация кода. Т.е. необходимо в виде объектов (метод, параметр и т.п.) описать интерфейс и созданный объект интерфейса должен уметь возвращать свое описание в виде кода на Java. Предполагает наличие классов для описания параметра и метода.

12. (*) Реализовать класс «Таблица данных». Поддерживаемые возможности: именованные столбцы, произвольное количество строк, добавление, удаление строк, столбцов, работа с ячейками и т.д. Должен быть интерфейс (набор открытых методов и классов), позволяющий выполнить следующий код (идет работа с одной и той же ячейкой):

```
table.column("column_name").cell(5).setValue("123");
int value = table.row(5).cell("column_name").getValueAsInt("123");
table.cell("column_name", 5).setValue(value + 1);  // параметрический полиморфизм
table.cell(5, "column_name").setValue(value + 2);  // параметрический полиморфизм
```

13. (*) Реализовать класс Формула (интерпретатор). Должна быть возможность работы с данным классом следующим образом:

```
Formula f = new Formula();
f.prepare("x*x +y");
double v = f.execute(2, 5.7);
// и т.п.
```

14. (*) Объектное представление текстового документа (секции, списки) с форматированием текста в plain text. Классы - Параграф, Стиль и Документ. У Стиля свойства: отступы сверху, снизу, слева, справа, красная строка (измеряется в кол-ве строк или символов), выравнивание (влево/вправо/по центру/по ширине), признак списка (с номером или с маркером). Параграф наследуется от стиля, а также может иметь стиль (собственные параметры параграфа перекрывают параметры стиля). Кроме того, для параграфа может быть задан номер, с которого начинается нумерация. Документ состоит из параграфов и может печатать/сохранять форматированный документ в текстовом виде.

15. Словарь (ключ – объект, значение – объект/набор объектов). Словарь, кроме стандартных, должен содержать методы:
    - получить список ключей для заданного объекта,
    - удалить заданный объект для любых значений ключа,
    - удалить заданный объект для заданного ключа.

16. (*) Реализовать класс «Геометрическая фигура на плоскости» (многоугольник). Задается в виде набора точек (ограничена линиями, проведенными от точки [0] до [1], от [1] до [2] и так далее, последняя линия соединяет точку [n-1] с точкой [0]). Должны быть реализованы методы вычисления площади фигуры (легко найти алгоритм в интернете), периметра, перемещения фигуры, вычисления прямоугольника, описывающего данный многоугольник, а также масштабирования фигуры по вертикали и горизонтали относительно верха/низа/середины и левого края/правого/середины фигуры, а также отображения фигуры. Для демонстрации работы реализовать отображение фигуры на форме. Редактор многоугольника делать не требуется (можно задавать фигуру в коде или считывать из файла). В идеале (часть задания, которую можно не выполнять) сделать метод сохранения фигуры в svg-формате. Непонятные вам моменты по заданию уточнить у преподавателя.

17. Более простой вариант предыдущей задачи – все то же самое, но для прямоугольника. Непонятные вам моменты по заданию уточнить у преподавателя.

18. (*) Эмуляция файловой системы. Классы: директория, файл (наверно, директория должна наследоваться от файла). Должна поддерживаться концепция текущей директории, относительно которой возможны все операции. Для упрощения задачи будем считать, что файлы только текстовые и записать/прочитать файл можно только целиком (также возможна дозапись в файл).
    Реализовать простейший эмулятор командной строки с командами: cd (перемещение по

файловой системе), ls (список файлов), mkdir (создание директории), rm (удаление), echo (печать строки), cat (печать файла) и tree (печать содержимого директории в виде дерева), а также мозможностью перенаправления вывода в файл с помощью выражений ">" и ">>".

Пример возможной сессии:

```
mkdir "Первая папка"
cd "Первая папка"
echo "строка 1" > ./file1.txt
echo "строка 2" >> ./file1.txt
cat file1.txt >../copy1.txt
ls / >ls.txt
cd ..
tree .
```

Что конкретно делает каждая команда – поискать в Интернете (естественно, требуется реализации в самом простейшем виде, без ключей и т.п.).

В результате чего должна получиться файловая система:

```
/
  Первая папка/
    file1.txt
    ls.txt
  copy1.txt
```

Команда tree примерно это и должна напечатать.

19. Реализовать класс «Геометрическая фигура Звезда» (вида https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D0%B5%D0%B7%D0%B4%D0%B0_(%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)#/media/File:Etoile_%C3%A0_quatre_branches.svg). Задается параметрами: координатами центра, внешним радиусом, внутренним радиусом и кол-вом "лучиков". Реализовать методы: отображение, перемещение, масштабирование (см. задачу № 14). В идеале (часть задания, которую можно не выполнять) сделать метод сохранения фигуры в svg-формате.

20. Описать класс для представления Тарифов на международную связь. Данный класс хранит в себе список направлений в виде код (префикс) направления, название направления, цена минуты разговора. Класс должен уметь загружать информацию из текстового файла заданного формата (проще всего csv), сохранять тарифы в такой же файл, иметь возможность добавления/удаления/модификации направления. И самое главное, класс должен уметь считать стоимость звонка, заданного вызываемым номером и длительностью в секундах. Для этого надо найти направление с самым длинным кодом (префиксом), подходящим к номеру, перевести длительность звонка в минуты (с округлением вверх, звонки короче 6 секунд не тарифицируются). Интерфейс для демонстрации может быть консольным.

21. Описать в виде класса калькулятор для расчета платежей по кредиту. Кредит задается параметрами: сумма кредита, срок кредита в месяцах, фиксированная процентная ставка, способ погашения кредита (дифференцированные платежи / аннуитетные платежи). Класс должен позволять задать каждый их этих параметров, а также рассчитать сумму платежей в n-ый по счету месяц, а также общую суммы платежей по кредиту. Интерфейс может быть консольным.

22. Реализовать класс, моделирующий солнечную систему. Добавляются планеты со скоростью и радиусом вращения и диаметром и это все визуализируется. За положение планет в заданный момент времени отвечает класс планеты (объекты данного класса).

23. (*) Реализовать класс Светофор. Класс светофор – невизуальный, содержит таймер, который переключает цвета, имеет события изменения состояния (цвета), поддерживает различные конфигурации светофора (пешеходный, с тремя кругами, с дополнительной секцией поворота и т.п.), позволяет задать тайминги для переключений состояния светофора от одного к другому (с зеленого на желтый, например, и т.п.). Т.е. класс Светофор – это своего рода эмулятор микроконтроллера, который управляет светофором. Продемонстрировать на форме работу данного класса. Форма отвечает только за перерисовку текущего состояния светофора при возникновении событий изменения состояния светофора.

24. Реализовать класс для представления лабиринта на клеточном поле. Должен быть реализован метод поиска путей в лабиринте.

25. Реализовать класс Словарь (ключ – строка, значение – объект). Можно самый примитивный вариант на основе массива.

26. Динамический стек на основе массива – обобщенная (generics) реализация. При создании стека размер внутреннего массива равен capacity – передается в конструктор, если используется конструктор без параметром, то значение по умолчанию = 10. Если в процессе работы со стеком добавляется (capacity + 1)-ый элемент, то размер внутреннего массива должен быть увеличен в 2 раза, т.е. capacity становится равным 2 * capacity (предыдущее значение) элементам. Более подробно принципы работы такой схемы только на примере ArrayList описаны в статье https://habr.com/ru/post/128269/.

27. Динамическая кольцевая очередь на основе массива – обобщенная (generics) реализация. Стратегия поведения при нехватке размера внутреннего массива для хранения всех добавленных элементов очереди – см. комментарии к предыдущей задаче.

28. Реализовать класс, одновременно являющийся и стеком и очередью (собственная реализация на основе двух-связанного списка, не используя готовые классы, спросить у преподавателя как).

29. Динамическая кольцевая двойная очередь (Deque) на основе массива – обобщенная (generics) реализация. Стратегия поведения при нехватке размера внутреннего массива для хранения всех добавленных элементов очереди – см. комментарии к задаче о стеке (задача №13).

30. Очередь с приоритетом на основе массива или связного списка (готовые классы из стандартной библиотеки не использовать) – обобщенная (generics) реализация. При создании очереди задается java.util.Comparator<T>, который используется для определения приоритета добавляемых элементов).

31. Класс для описания вектора в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами и т.п.

32.

**. Остальные задачи позже. Предложения принимаются.