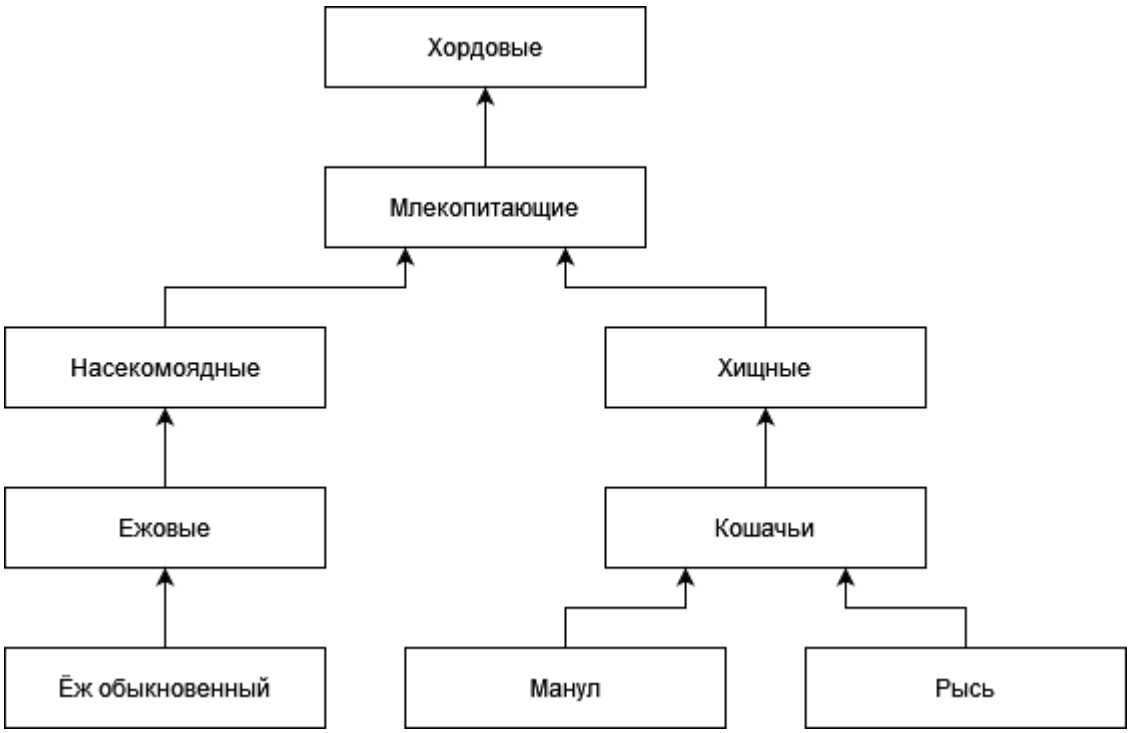
1. В компьютерной игре герой (класс Hero) может перемещаться между двумя точками (метод move) различными способами: идти пешком, ехать на лошади, лететь и т. п. Реализовать классы, позволяющие выбирать и менять в ходе выполнения программы способ перемещения героя, используя паттерн “стратегия” (strategy). Продемонстрировать работу реализованных классов.

2. Написать аннотацию с целочисленным параметром. Создать класс, содержащий публичные, защищенные и приватные методы (2-3 каждого вида) с параметрами, аннотировать любые из них. Вызвать из другого класса все аннотированные защищенные и приватные методы столько раз, сколько указано в параметре аннотации.

3. Дана иерархия животного царства:



Создать метод `segregate` вида:

`segregate(SrcCollection, Collection1, Collection2, Collection3)`

Где:

`SrcCollection` – исходная коллекция животных

`Collection1, Collection2, Collection3` – коллекции, в которые должны быть распределены соответственно ежи, манулы и рыси из `SrcCollection`

Необходимо, чтобы была возможность вызвать метод следующими способами:

`segregate(Млекопитающие, Ежовые, Кошачьи, Хищные)`

`segregate(Хищные, Хордовые, Манулы, Кошачьи)`

`segregate(Ежовые, Насекомоядные, Хищные, Хищные)`

Продемонстрировать работу метода

4. Реализовать программу-переводчик.
   4.1. При запуске программы выполняется чтение словаря в следующем формате: слово или выражение | перевод
   4.2. Затем читается входной файл и выполняется перевод
   4.3. Перевод осуществляется по следующим правилам:
      - регистр букв игнорируется
      - если искомого слова нет в словаре – выводится без перевода
      - если есть несколько подходящих вариантов, выбирается вариант максимальной длиной левой части
   4.4. Результат перевода выводится в консоль
   4.5. Создать и применить пользовательские исключения:
      - `InvalidFileFormatException` см. пункт 4.1;
      - `FileReadException` файла не существует, нет доступа к файлу и т.д.

5. С использованием только Stream API реализовать следующие методы:
   - метод, возвращающий среднее значение списка целых чисел;
   - метод, приводящий все строки в списке в верхний регистр и добавляющий к ним префикс «_new_»;
   - метод, возвращающий список квадратов всех встречающихся только один раз элементов списка;
   - метод, принимающий на вход коллекцию строк и возвращает все строки, начинающиеся с заданной буквы, отсортированные по алфавиту;
   - метод, принимающий на вход коллекцию и возвращающий ее последний элемент или кидающий исключение, если коллекция пуста;
   - метод, принимающий на вход массив целых чисел, возвращающий сумму чётных чисел или 0, если чётных чисел нет;
   - метод, преобразовывающий все строки в списке в Map, где первый символ – ключ, оставшиеся – значение;

6. Создать супервизор (управляющую программу), которая контролирует исполнение абстрактной программы.

Абстрактная программа работает в отдельном потоке и является классом с полем перечисляемого типа, который отражает ее состояние

- UNKNOWN – перед первым запуском
- STOPPING - остановлена
- RUNNING - работает
- FATAL ERROR – критическая ошибка

и имеет поток-демон случайного состояния, который в заданном интервале меняет её состояние на случайное.

У супервизора должны быть методы остановки и запуска абстрактной программы, которые меняют ее состояние. Супервизор является потоком, который циклически опрашивает абстрактную программу, и если ее состояние STOPPING, то перезапускает ее. Если состояние FATAL ERROR, то работа абстрактной программы завершается супервизором. Все изменения состояний должны сопровождаться соответствующими сообщениями в консоли. Супервизор не должен пропустить ни одного статуса абстрактной программы. Использовать конструкции с wait/notify.

7. Создать приложение с графическим интерфейсом для заданий 1–6. Для этого приложения должна быть реализована возможность выбора из списка любого приложения, ввод входных данных и его выполнение. Модифицировать задания 1–6 так, чтобы весь вывод происходил в текстовых областях, защищенных от редактирования.