ИНТЕРФЕЙСЫ И ШАБЛОНЫ ООП

ЛАБОРАТОРНАЯ РАБОТА №6

Лабораторная работа БЕЗ вариантов. Необходимо выполнить ВСЕ задания в одном проекте. В классах могут присутствовать методы со спецификатором доступа private вспомогательного характера. Организовать проверку входных данных. К каждому классу и методу написать xml-документацию.
Каждая задача оценивается по 3 балла, xml-документация оценивается в 2 балла. Максимально за лабораторную работу можно получить 10 баллов (8 баллов за решение задач + 2 балла за оформление отчета). За решение с использованием шаблонов можно получить дополнительные 5 баллов, которые можно добавить к любой контрольной точке.

Задание 1. Кот

1	**Кот мяукает.** Создайте сущность Кот, которая описывается следующим образом: • Имеет Имя (строка) • Для создания необходимо указать имя кота. • Может быть приведен к текстовой форме вида: “кот: Имя” • Может помяукать, что приводит к выводу на экран следующего текста: “Имя: мяу!”, вызвать мяуканье можно без параметров. • Может помяукать N раз, что приводит к выводу на экран следующего текста: “Имя: мяу-мяу-…-мяу!”, где количество “мяу”равно N. Создайте кота по имени “Барсик”, и затем пусть он помяукает сначала один раз, а затем три раза.
2	**Интерфейс Мяуканье.** Разработайте метод, который принимает набор объектов способных мяукать и вызывает мяуканье у каждого объекта. Мяукающие объекты должны иметь метод со следующей сигнатурой: `public void meow();` Дополните решение предыдущей задачи так, чтобы иметь возможность передать Кота в написанный вами метод и протестируйте работоспособность решения передав в него нескольких котов, а также создайте любой другой произвольный класс и передайте в написанный метод его объекты.

3	**Количество мяуканий.**
	Необходимо воспользоваться классом Кот и методом принимающим всех мяукающих из предыдущего задания. Необходимо таким образом передать кота в указанный метод, что бы после окончания его работы узнать сколько раз мяукал кот за время его работы. На рисунке показан пример работы. Перед вызовом метода создаем кота, отправляем ссылку на кота в метод, после окончания его работы выводим количество мяуканий на экран. Кота изменять нельзя.
	`Meowable m = ...//создаем кота`
	`Funs.meowsCare(m);`
	`System.out.println(...) // вывод: кот мяукал 5 раз`

Задание 2. Дроби

1	**Дроби.** Создайте сущность Дробь со следующими особенностями: • Имеет числитель: целое число • Имеет знаменатель: целое число • Дробь может быть создана с указанием числителя и знаменателя • Может вернуть строковое представление вида “числитель/знаменатель” • Может выполнять операции сложения, вычитания, умножения и деления с другой Дробью или целым числом. Результатом операции должна быть новая Дробь (таким образом, обе исходные дроби не изменяются) (Обязательно использовать перегрузки!) • Необходимо корректно обрабатывать отрицательные значения. Учтите, что знаменатель не может быть отрицательным. Затем необходимо выполнить следующие задачи: 1. Создать несколько экземпляров дробей. 2. Написать по одному примеру использования каждого метода. 3. Вывести на экран примеры и результаты их выполнения в формате «1/3 * 2/3 = 2/9» 4. Посчитать f1.sum(f2).div(f3).minus(5)
2	**Сравнение дробей.** Переопределите метод сравнения объектов по состоянию таким образом, чтобы две дроби считались одинаковыми тогда, когда у них одинаковые значения числителя и знаменателя.
3	**Клонирование дроби. Интерфейс ICloneable.** Переопределите метод клонирования, таким образом, чтобы при его вызове возвращался новый объект Дроби, значения полей которого будут копиями оригинальной Дроби.
4	В класс Дробь, добавить интерфейс на два метода: получение вещественного значения, установка числителя и установка знаменателя. Сгенерировать такую версию дроби, которая будет кэшировать вычисление вещественного значения.