**Задача 1.**

Создайте пользовательское исключение **IncorrectAgeException**, наследуемое от исключения **ArgumentException**, с сообщением «Некорректный возраст».

Создайте класс **People** с членами класса:

* Свойство **Age**;
* Конструктор от одного параметра, принимающий возраст человека.

Выбрасывайте исключение из конструктора, если возраст человека больше 120 или меньше 0.

Создайте класс **Druid** с членами класса:

* Свойство **Age**;
* Конструктор от одного параметра, принимающий возраст друида.

Выбрасывайте исключение из конструктора, если возраст друида больше 2000 или меньше 0.

В основной программе попробуйте создать 20 экземпляров класса **People** и 20 – класса **Druid**. Посчитайте количество успешно созданных экземпляров. Обрабатывайте исключения, выводя их сообщения.

Выведите количество успешно созданных объектов каждого класса.

**Задача 2.**

Смоделируйте экзамен по истории. Для этого:

Создайте класс **Student** с членами:

* Целочисленное свойство **CurrentAssessment**;
* Статический метод **PassExam**, принимающий в параметрах студента и экзамен и возвращающий оценку за экзамен.

Создайте класс **TimeIsOverException**, наследуемый от **Exception**, с сообщением “Экзамен завершается с оценкой 0. ))”

Создайте класс Exam с членами:

* DateTime ExamStartTime – дата и время начала экзамена;
* DateTime ExamStartTime – дата и время окончания экзамена;
* Конструктор, принимающий дату и время начала экзамена и время его написания;

Студент получает за экзамен автоматом его накопленную оценку, если **CurrentAssessment** ≥ 6. Иначе Студенту необходимо на написание экзамена случайное время от получаса до двух часов. В случае, если Студент не успевает написать экзамен, метод должен выбрасывать исключение **TimeIsOverException**. Иначе метод возвращает случайную оценку от 2 до 10.

В основной программе считайте с клавиатуры количество студентов, пришедших на экзамен, и их накопленные оценки.

Выведите на экран оценку каждого студента и сообщения об исключениях. В случае, если студент не успевает закончить работу, экзамен завершается с оценкой 0.

**Задача 3.**

Как известно, когда студент не хочет напрягаться и выполнять какое-то задание, его первая мысль – скопипастить работу другого студента. Напишите программу, моделирующую работу системы антиплагиата. Для этого в библиотеке классов определите статический класс **WorksComparer**. В классе определите метод **CompareWorks** с двумя параметрами: путями к работам двух студентов на диске, в котором, считывая данные из двух файлов, определите, какое количество строк совпадает. Если процент совпадения двух файлов превышает 60% кидайте исключение **PlagiarismFoundException** с сообщением «Обнаружена копипаста. Оценка за обе работы 0 / 2.»

В основной программе создайте два файла одинаковой длины и заполните их случайными целыми числами из диапазона (0, 100]. Сравните эти файлы и выведите результат сравнения. (Если процент совпадения двух файлов не превышает 60%, выводите сообщение: «Допустимый процент плагиата»).

Обработайте исключения, которые могут возникнуть при работе с файлами.