Лабораторная работа 03. Обработка исключительных ситуаций

Задание I.

1) Создать статический метод, возвращающий

$$f(x) = \begin{cases} x + sin^2\left(\frac{1}{x-a} + 4\right), & x < 0 \\ \frac{ax}{\sqrt{a^2 - x^2}}, & x \geq 0 \end{cases}$$

причем величина *x* передается через параметр метода,
а значение величины *a* вычисляется внутри метода как случайное целое число из диапазона [-50, 50].

Протестировать работу метода на подходящих примерах, при получении не числовых значений (NaN, Infinity) выводить соответствующие текстовые сообщения. Использовать try-catch.

2) Создать статические методы, для

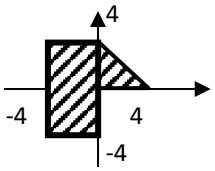
a) ввода с консоли части элементов одномерного вещественного массива (в диапазоне индексов от k1 до k2 включительно; индексы передаются через параметры метода),
остальные элементы массива не меняются

b) заполнения случайными числами части одномерного вещественного массива (в диапазоне индексов от k1 до k2 включительно; индексы передаются через параметры),
остальные элементы массива не меняются

Протестировать методы на подходящих примерах

3) В программе, используя разработанные методы и конструкцию try-catch, решить следующие задачи:

1. Ввести два целых числа k1, k2. Ввести целое число n.
   Объявить одномерный массив x из n элементов и заполнить его:
   элементы с индексами от k1 до k2 ввести с клавиатуры,
   остальные заполнить случайными числами.
   Вывести полученный массив
2. Одномерный массив y из n элементов заполнить значениями f(x[i]), вывести полученный массив
3. Для всех пар (x[i], y[i]), как координат точек на плоскости, вывести те, что попали в заштрихованную область и их количество



При отсутствии значения y[i] точку игнорировать, остальные точки - обрабатывать

4. Для пар (x[i], y[i]), как координат точек на плоскости, подсчитать длину ломаной линии, соединяющей их в порядке индексации в массиве.

   При отсутствии значения y[i] расчет прекратить и вывести сообщение об этом

4) Создать собственные классы исключений для обработки ошибок, возникающих в методах из пп 1) и 2).

Сделать по крайней мере одно из них отслеживаемым (checked) и хотя бы одно – не отслеживаемым (unchecked)

В полях этих классов должны храниться данные, позволяющие анализировать причины ошибок (приводящие к ошибкам значения аргументов методов, локальных переменных,…)

Создать реализацию методов из пп 1) и 2) с использованием этих классов. Продемонстрировать их применение на подходящих примерах.

Задание II(*доп). Создать класс для обработки текстовых сообщений

Класс должен содержать поле строкового типа для хранения одного текстового сообщения

Набор методов класса должен обеспечивать следующие возможности

0. Инкапсуляция данных в классе: геттеры, сеттеры, конструкторы, ввод-вывод, сравнение,…

1. Генерация нового текста - статические методы, каждый возвращает новый объект такого класса с сообщением, построенным по заданному правилу:
   а) сообщение из случайного набора символов Unicode
   б) сообщение из случайного набора символов Unicode, без повторений
   в) сообщения из заданного диапазона символов Unicode
   г) сообщения из заданного набора отдельных символов (произвольное количество)
2. Расчет и проверка характеристик
   а) относительная частота (вероятность) символа в сообщении
   б) количество информации в сообщении по формуле Хартли*
   в) количество информации в сообщении по формуле Шеннона*

(* при расчете количества информации алфавитом считать только символы, имеющиеся в сообщении)

   г) проверка, что текст является палиндромом (реализовать в двух версиях: итерационно и рекурсивно)
3. Формирование измененных сообщений
   а) на основе сообщения строится новое сообщение-палиндром.
      Например, исходное “abc”, новое “abccba”
   б) из исходное сообщения исключаются все символы указанного набора
   в) исходное сообщение переставляется циклически на k позиций вправо или влево
      Например, исходное “abcdefg”;
      сдвиг вправо на 3 дает “efgabcd”;
      сдвиг влево на 2 дает “cdefgab”;
   г) из набора объектов с сообщениями формируется один новый объект, текст в котором состоит из первых символов сообщений в исходном наборе

Обеспечить контроль возможных ошибок конструкцией try..throw..catch

Протестировать все методы на подходящих примерах

Обработка исключительных ситуаций

Нередко в процессе выполнения программы могут возникать ошибки, при том необязательно по вине разработчика. Некоторые из них трудно предусмотреть или предвидеть, а иногда и вовсе невозможно. Так, например, может неожиданно оборваться сетевое подключение при передаче файла. Подобные ситуации называются исключениями.

Исключения - это ошибки, обнаруженные во время выполнения программы, тогда как синтаксические ошибки - это ошибки, обнаруженные во время компиляции программы в байт-код.

Исключение прерывает нормальное выполнение программы.

В языке Java предусмотрены специальные средства для обработки подобных ситуаций. Одним из таких средств является конструкция try...catch...finally. При возникновении исключения в блоке try управление переходит в блок catch, который может обработать данное исключение. Если такого блока не найдено, то пользователю отображается сообщение о необработанном исключении, а дальнейшее выполнение программы останавливается. И чтобы подобной остановки не произошло, и надо использовать блок try..catch.

При использовании блока try...catch вначале выполняются все инструкции между операторами try и catch. Если в блоке try вдруг возникает исключение, то обычный порядок выполнения останавливается и переходит к инструкции catch. Поэтому когда выполнение программы дойдет до строки numbers[4]=45;, программа остановится и перейдет к блоку catch

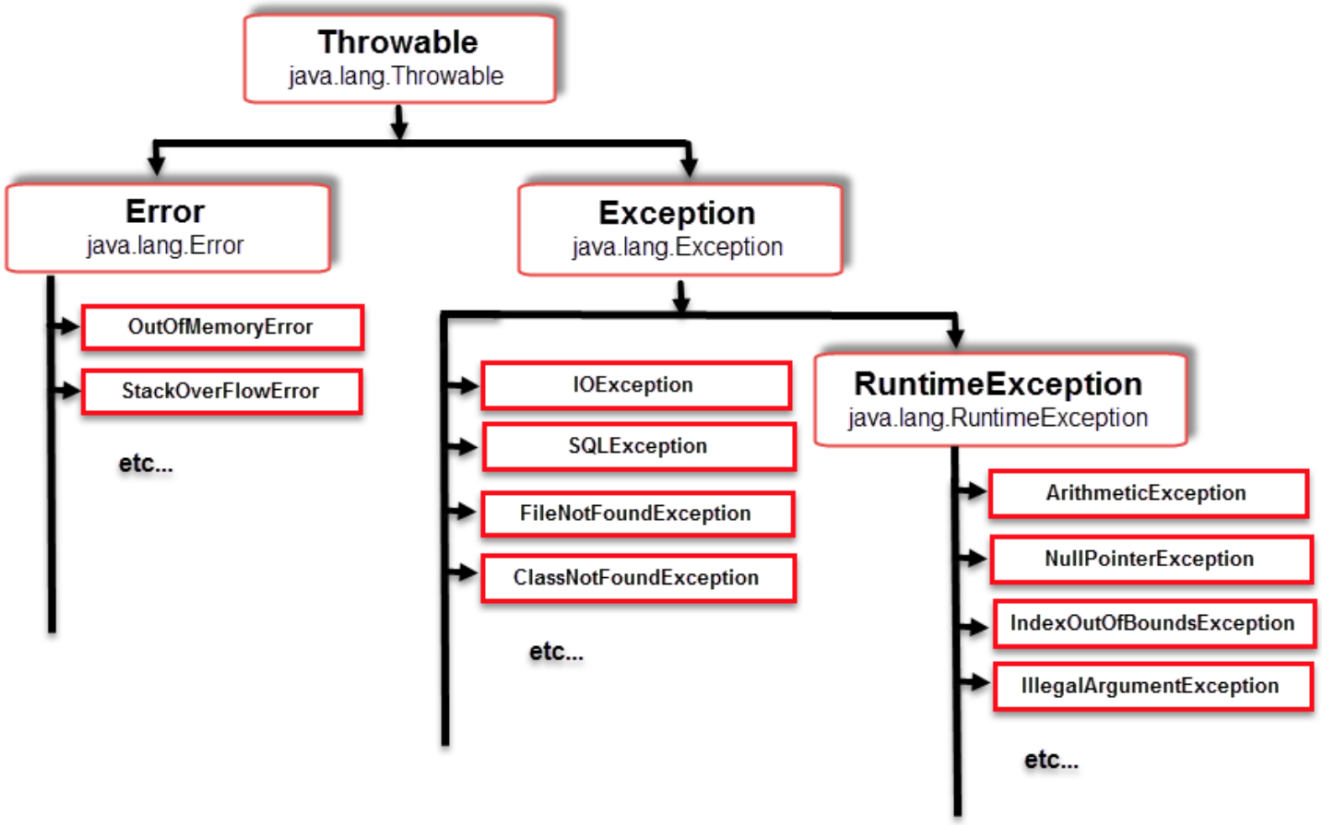
Выражение catch имеет следующий синтаксис: catch (тип_исключения имя_переменной). В данном случае объявляется переменная ex, которая имеет тип Exception. Но если возникшее исключение не является исключением типа, указанного в инструкции catch, то оно не обрабатывается, а программа просто зависает или выбрасывает сообщение об ошибке.

Но так как тип Exception является базовым классом для всех исключений, то выражение catch (Exception ex) будет обрабатывать практически все исключения. Обработка же исключения в данном случае сводится к выводу на консоль стека трассировки ошибки с помощью метода printStackTrace(), определенного в классе Exception.

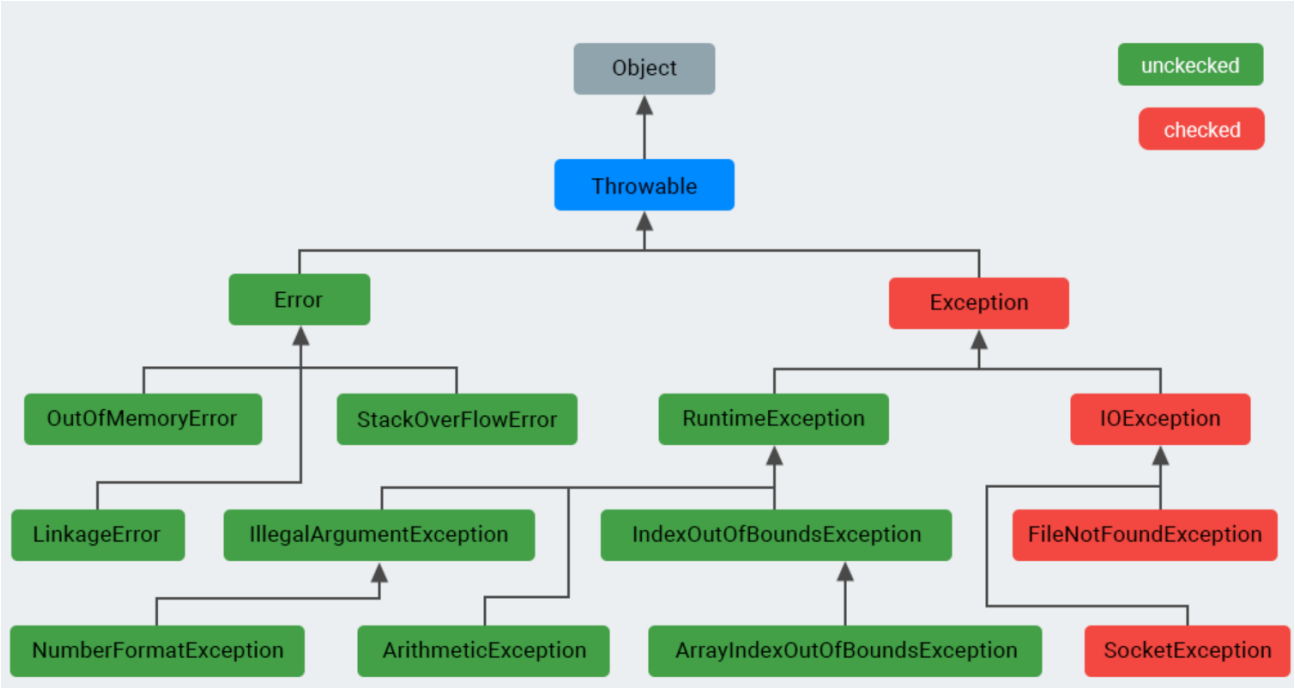
После завершения выполнения блока catch программа продолжает свою работу, выполняя все остальные инструкции после блока catch.

Конструкция try..catch также может иметь блок finally. Однако этот блок необязательный, и его можно при обработке исключений опускать. Блок finally выполняется в любом случае, возникло ли исключение в блоке try или нет

Иерархия классов исключений в java



Проверяемые исключения: checked exceptions



Все исключения в Java делятся на 2 категории — проверяемые (checked) и непроверяемые (unchecked).