МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

ЛАБОРОТОРНА РОБОТА №2

з дисципліни «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему "Пакети та вийняткові ситуації"

Студента 7 курсу групи АД-181
Гежа Н. І.
Перевіряв
доцент Рудніченко Н. Д.

Одеса - 2019

ЗМІСТ

ВСТУП

Ціль роботи — отримання знань і практичних навичок роботи з пакетами, винятками та їх обробкою, створення власних винятків у мові програмування Java.

ТЕОРЕТИЧНА ЧАСТИНА

Виняткова ситуація - це помилка, яка виникає в результаті виконання програми. Виняток в Java - це об'єкт, Який описує виняткову ситуацію (помилку).

При виникненні виняткової ситуації в процесі виконання програми автоматично створюється об'єкт, що описує цю виняткову ситуацію. Цей об'єкт передається для обробки методу, в якому виникла виняткова ситуація. Кажуть, що виключення викидається в метод. Після отримання об'єкта виключення метод може обробити його або передати для обробки далі (куди саме - інше питання).

Винятки (об'єкти, що описують виняткові ситуації) генеруються автоматично, проте їх також можна генерувати «вручну», тобто спеціальними програмними методами. На перший погляд, така можливість здається зайвою і непотрібною, але це не так. Далі ми побачимо, що механізм обробки виняткових ситуацій, в тому числі штучне генерування виключень, нерідко дозволяє зробити програмний код більш компактним і елегантним, значно спрощуючи рішення складних, на перший погляд, завдань.

Для того щоб метод міг обробити виняткову ситуацію, необхідно передбачити програмний код обробки цієї ситуації - на випадок її виникнення. По-перше, потрібно виділити фрагмент коду, який повинен контролюватися на предмет генерування виняткової ситуації. По-друге, необхідно створити програмний код, безпосередньо обробляє виняткову ситуацію, тобто код, який виконується в разі виникнення виняткової ситуації.

В Java для обробки виняткових ситуацій використовується блок try-catch-finally. У блок try поміщається програмний код, який відстежується на випадок,

якщо виникне виняткова ситуація. Якщо виняткова ситуація виникає, то управління передається блоку catch. Програмний код в цьому блоці виконується, тільки якщо виникає виняткова ситуація, причому не будь-яка, а певного типу. Аргумент, що визначає, якого типу виняткові ситуації обробляються в блоці catch, вказується після ключового слова catch в круглих дужках, тобто в тому ж форматі, що і аргумент методу.

Оскільки в блоці try можуть виникати виключення різних типів, для кожного з них можна передбачити свій блок catch. Якщо блоків catch кілька, при виникненні виняткової ситуації вони перебираються послідовно до збігу типу виняткової ситуації з аргументом блоку catch.

Після блоків try і catch можна вказати блок finally з кодом, який виконується в будь-якому випадку незалежно від того, виникла виняткова ситуація чи ні.

ЗАВДАННЯ 1

1. Вивчити роботу з пакетами

2. Створити додаток, в якому:

a. Продемонструвати вміння працювати з пакетами (доступ до імен з інших пакетів, імпорт пакетів)

b. Продемонструвати вміння обробляти виняткові ситуації:

- З використанням множинного блоку catch ()
- З використанням вкладених блоків try ()
- З використанням штучного генерування виключень
- З використанням викидання винятків методами
- З використанням створення власних виключень

В якості зразка використовувати додаток «Калькулятор»

У файлі Main була написана початкова функція, створююча об’єкт NewJFrame з додатку з пакету “SomeOtherPerson”. Після цього, для запуску калькулятора викликається функція main цього об’єкту. Створення проходить у try-catch блоці, який опрацьовує будь-яке виключення. Програма працює, та її робота показана на малюнку 1.

```
package AD181.Gezha;

import SomeOtherPerson.NewJFrame;

public class Main {
    public static void main(String[] args) {
              try {
                  NewJFrame frame = new NewJFrame();
                  frame.main(args);
        } catch(Exception e) {
                  System.out.println("Error: " + e.getMessage());
        }
    }
}
```

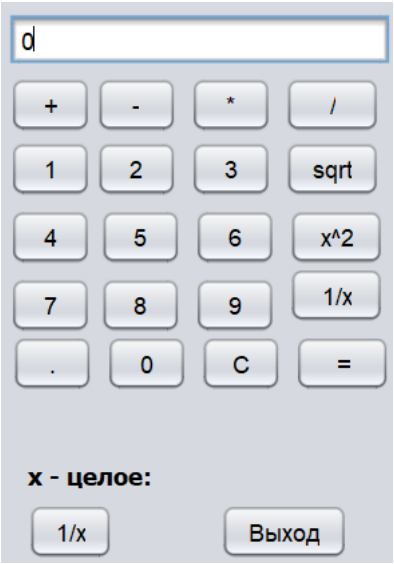

Рис. 1: запущена програма

Був написаний клас вийнятку ExceptionWithDate, зберігаючий час створення вийнятку разом з його текстом. Це дозволяє мати точну інформацію стосовно дати та часу виникнення виняткової ситуації під час роботи програми, що поліпшує дебаг програми.

```
class ExceptionWithDate extends Exception {
  private String error;

  ExceptionWithDate(String str) {
    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
    LocalDateTime now = LocalDateTime.now();
    error = str + " on date " + dtf.format(now);
  }

  public String toString() {
    return "This is a custom exception:" + error;
  }
}
```

Метод MyExDelZer, використаний у перевірці ділення числа на 0, був замінений на код приведений донизу. Новий код використовую раніше створений тип винятку для того щоб записувати час винятку. На малюнку 2 приведений приклад обробки кинутого даною функцією винятку.

```
static float MyExDelZer(float x) throws ExceptionWithDate {
  if (Math.abs(x) >= 0.00000001) return x;
```

```
  else throw new ExceptionWithDate("division by zero");
}
```

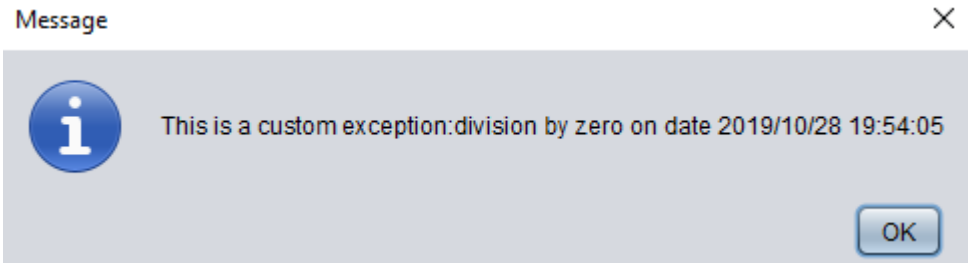

Рис. 2: обробка винятку з датою

ВИСНОВОК

У ході роботи були отримані знання та практичні навички роботи з пакетами, винятками та їх обробкою, створення власних винятків у мові програмування Java.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Форум StackOverflow
2. Приклади з лекції