# Podstawy programowania

## Laboratorium 10 - wprowadzenie do obsługi plików tekstowych i wyjątków

mgr inż. Krzysztof Szwarc

krzysztof@szwarc.net.pl

Sosnowiec, 24 maja 2021

# Wyjątki

## Opis

Jeżeli w programie pojawi się operacja niedozwolona (np. przypisanie do zmiennej typu int ciągu znaków) zgłasza on określony wyjątek i kończy swoje działanie. Aby nasza aplikacja nie finalizowała działania, w momencie pojawienia się błędu, możemy zastosować obsługę wyjątków polegającą na złapaniu błędu (catch) i obsłużeniu go. Części składowe obsługi wyjątków:

1. Blok try (obowiązkowy).
2. Blok catch (opcjonalny/obowiązkowy).
3. Blok finally (opcjonalny/obowiązkowy).

# Ogólna struktura obsługi wyjątków

```
try {
     kod mogący spowodować wystąpienie błędu
}
catch (typWyjątku obiekt) {
     kod obsługujący wyjątek
}
catch (innyTypWyjątku obiekt) {
     kod obsługujący wyjątek
}

⋮

finally { kod, który wykona się niezależnie od
   wystąpienia wyjątku }
```

# Przykład

```
int var = 0;
try {
    var = reader.nextInt();
}
catch (InputMismatchException ex) {
    System.out.print("Wpisałeś złą wartość");
    var = 5;
}
finally {
    System.out.print(var); // 5 lub wpisana
       wartość
}
```

# Przykład

```
int var = 0;
try {
    var = reader.nextInt();
}
catch (InputMismatchException ex) {
    System.out.print("Wpisałeś złą wartość");
    var = 5;
}
System.out.print(var); // 5 lub wpisana
   wartość
```

# Przykład - niezalecany

```
int var = 0;
try {
    var = reader.nextInt();
}
catch (Exception ex) {
    System.out.print("Błąd - "+ ex.toString());
    var = 5;
}
finally {
    System.out.print(var); // 5 lub wpisana wartość
}
```

# Zadanie

1. Napisz program pobierający od użytkownika liczbę całkowitą, a następnie dzielący przez nią liczbę 5. Obsłuż możliwość wpisania przez użytkownika liczby $0$.
2. Dla powyższej aplikacji wykorzystaj metodę printStackTrace przechwyconego wyjątku (w bloku catch).

# Klasa File

## Tworzenie instancji klasy File

Klasa File wykorzystywana jest do przechowywania danych o pliku. Spssób utworzenia jej instancji jest następujący:

```
File file = new File("sciezkaPliku");
```

Aby móc korzystać z klasy musimy ją zaimportować:

```
import java.io.File;
```

Dla operacji mogących spowodować wystąpienie wyjątku wymagane jest zaimportowanie klasy:

```
import java.io.IOException;
```

# Przegląd najważniejszych metod

| Metoda | Opis |
| --- | --- |
| exists() | Zwraca wartość typu Boolean reprezentującą informację czy plik istnieje |
| createNewFile () | Tworzy nowy plik |

# Przykład

```
public static void main(String[] args) {
    File file = new File("sciezkaPliku");
    if (!file.exists()) {
        try {
            file.createNewFile();
        }
        catch (IOException ex) {
            System.out.print("Błąd");
            ex.printStackTrace();
        }
    }
}
```

# Uwaga

## Opis

W celu uproszczenia zapisu na kolejnych slajdach nie będą obsługiwane błędy (zastosowano instrukcję throws, która zgłosi wystąpienie wyjątku wyżej w hierarchii). Proszę posługiwać się zapisem korzystającym z bloku try, catch oraz finally.

# Klasy do obsługi IO

## Odczyt pliku

1. FileReader (korzysta z systemowego kodowania).
2. BufferedReader (korzysta z buforowania).

Łańcuchowanie umożliwia wykorzystanie obiektu danej klasy jako parametru konstruktora innej np.

```
BufferedReader wej = new
   BufferedReader(new
   FileReader("plik"));
```

# Klasy do obsługi IO

## Zapis pliku

1. FileWriter (korzysta z systemowego kodowania).
2. BufferedWriter (korzysta z buforowania).
3. PrintWriter (korzysta z buforowania).

# Odczyt pliku tekstowego z użyciem Scannera

## Odczyt pliku tekstowego

Aby odczytać zawartość pliku tekstowego możemy skorzystać z klasy Scanner w następujący sposób:

```
File file = new File("sciezkaPliku");
Scanner reader = new Scanner(file);
```

Należy pamiętać, aby zamknąć plik po zakończeniu żądanych operacji (wykorzystując metodę close).

# Przykład

```
public static void main(String[] args)
    throws IOException {
      File file = new File("sciezkaPliku");
      if (!file.exists()) {
          file.createNewFile();
      }
      Scanner reader = new Scanner(file);
      String firstLine = reader.nextLine();
      System.out.println(firstLine);
      reader.close();
}
```

# Zadanie

1. Przerób przykładowy program tak, aby nie pojawiał się wyjątek (zabezpiecz go przed sytuacją, gdy w pliku nie ma tekstu).

# Odczyt pliku tekstowego z użyciem FileReadera

## Opis

Aby odczytać zawartość pliku tekstowego z wykorzystaniem FileReadera należy go zaimportować:

```
import java.io.FileReader;
```

Klasa traktuje plik jako strumień znaków.

# Przykład dla FileReadera

```
public static void main(String[] args) throws
    IOException {
     File file = new File("sciezkaPliku");
     if (!file.exists()) {
         file.createNewFile();
     }
     FileReader reader = new FileReader(file);
     int var;
     while((var = reader.read())!=-1) {
          System.out.print((char)var);
     }
     reader.close();
}
```

# Zadania

1. Utwórz plik tekstowy zawierający tekst „Ala ma kota”. Korzystając z klasy FileReader napisz metodę wypisującą wszystkie znaki znajdujące się w pliku do momentu znalezienia litery 'k'. Niech jako parametr metody będzie przekazywana ścieżka do pliku.
2. Utwórz plik tekstowy zawierający dwie linijki tekstu. Napisz metodę przyjmującą jako parametr ścieżkę pliku i zwracającą jego pierwszą linijkę. Wykorzystaj FileReadera.
3. Napisz metodę przyjmującą jako parametr ścieżkę pliku i zwracającą liczbę znaków znajdujących się w nim. Skorzystaj z FileReadera.

# Try-with-resources

```
File file = new File("plik.txt");
if (!file.exists()) {
    try {
        file.createNewFile();
    } catch (IOException ex) {
    }
}
try (FileReader reader = new FileReader(file)) {
    int var;
    while ((var = reader.read()) != -1 &&
        (char)var!='k')  {
        System.out.print((char) var);
    }
} catch (IOException ex) {
}
```

# Odczyt pliku tekstowego z użyciem BufferedReadera

## Opis

Aby odczytać zawartość pliku tekstowego z wykorzystaniem BufferedReadera należy go zaimportować:

```
import java.io.BufferedReader;
```

# Przykład dla BufferedReadera

```
public static void main(String[] args) throws
    IOException {
     File file = new File("sciezkaPliku");
     if (!file.exists()) {
        file.createNewFile();
     }
     FileReader reader = new FileReader(file);
     BufferedReader buffReader = new
        BufferedReader(reader);
     System.out.print(buffReader.readLine());
     buffReader.close();
}
```

# Zadanie

1. Napisz metodę „porownanie”, która będzie pobierała jako parametry ścieżki dwóch plików. Ma ona wypisać wszystkie linijki, które nie są identyczne w obu plikach. Wykorzystaj BufferedReadera.

# Zapis do pliku tekstowego

## Inicjalizacja obiektu klasy PrintWriter

Aby zapisać wartość do pliku tekstowego możemy skorzystać z klasy PrintWriter, której obiekt można zainicjować w następujący sposób:

```
PrintWriter wr = new PrintWriter("nazwaPliku");
```

Aby móc korzystać z klasy musimy ją zaimportować:

```
import java.io.PrintWriter;
```

Należy pamiętać, aby zamknąć plik po zakończeniu żądanych operacji wykorzystując metodę close.

# Przykład z nadpisywaniem zawartości pliku

```
public static void main(String[] args) throws
    IOException {
    PrintWriter writer = new PrintWriter("test");
    writer.println("test");
    writer.flush();
    writer.close();
}
```

# Przykład 2

```
public static void main(String[] args)
    throws IOException {
    Boolean append = true;
    File file = new File("test");
    PrintWriter writer = new
        PrintWriter(new
        FileOutputStream(file, append));
    file.println("test");
    file.flush();
    file.close();
}
```

# Przegląd najważniejszych metod

| Metoda | Opis |
|---|---|
| ptintln(tekst) | Wpisanie do pliku tekstu i zakończenie go znakiem nowej linii |
| print(wartosc) | Wpisanie do pliku wartości wartosc |
| close() | Zamknięcie pliku |
| flush() | Przesunięcie buforowanych danych do strumienia |

# Zapis do pliku tekstowego z użyciem FileWritera

## Zapis pliku tekstowego

Aby zapisać wartość do pliku tekstowego z wykorzystaniem FileWritera należy go zaimportować:

```
import java.io.FileWriter;
```

# Przykład dla FileWritera

```
public static void main(String[] args)
    throws IOException {
      Boolean append = true;
      File file = new File("sciezkaPliku");
      FileWriter writer = new
         FileWriter(file, append);
      file.write("tekst");
      file.close();
}
```

# Zapis do pliku tekstowego z użyciem BufferedWritera

## Opis

Aby zapisać wartość do pliku tekstowego z wykorzystaniem BufferedWritera należy go zaimportować:

```
import java.io.BufferedWriter;
```

# Przykład dla BufferedWritera

```
public static void main(String[] args)
    throws IOException {
      Boolean append = true;
      File file = new File("sciezkaPliku");
      FileWriter writer = new
         FileWriter(file, append);
      BufferedWriter buffWriter = new
         BufferedWriter(writer);
      buffWriter.write("tekst");
      buffWriter.flush();
      buffWriter.close();
}
```

# Zadania

Niech wszystkie programy zawierają informacje o swoim przeznaczeniu i będą zabezpieczone przed sytuacją, gdy plik nie istnieje.

1. Napisz program zawierający metodę przyjmującą w parametrze ścieżkę do pliku, która wypisze na ekran jego zawartość.
2. Napisz program zawierający metodę przyjmującą w parametrze ścieżkę do pliku oraz zmienną typu int offset. Niech metoda otwiera wskazany plik i pobierze jego co $n$-tą linię ($n{=}offset$). Nastepnie tekst ma zostać odwrócony (podpowiedź: skorzystaj ze StringBuildera) i wypisany na ekran.

# Zadania

Niech wszystkie programy zawierają informacje o swoim przeznaczeniu i będą zabezpieczone przed sytuacją, gdy plik nie istnieje.

3. Napisz program zawierający dwie opcje - odczyt oraz zapis. Jeżeli użytkownik wybierze odczyt należy wypisać informacje przechowywane w pliku „dane.txt” (jeżeli plik jest pusty lub nie istnieje należy wyświetlić odpowiedni komunikat). Dla zapisu niech użytkownik ma możliwość wpisania imienia oraz nazwiska, po czym obie wartości powinny zostać zapisane do pliku „dane.txt” (jeżeli plik nie istnieje należy go utworzyć).

# Zadania

Niech wszystkie programy zawierają informacje o swoim przeznaczeniu i będą zabezpieczone przed sytuacją, gdy plik nie istnieje.

4. Napisz program tworzący plik tekstowy zawierający 100 wylosowanych liczb całkowitych z przedziału od 0 do 10.
5. Napisz program, który odczytuje plik utworzony przez aplikację z zadania 4 i wypisuje na ekran posortowane rosnąco wartości.

# Dziękuję za uwagę