

# Podstawy i języki programowania

## Laboratorium 9 - wprowadzenie do obsługi plików tekstowych

mgr inż. Krzysztof Szwarz

krzysztof@szwarz.net.pl

Sosnowiec, 4 i 11 maja 2020

## 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 (typWyjatkku obiekt) { kod obsługujący  
    wyjątek }  
catch (innyTypWyjatkku obiekt) { kod  
    obsługujący wyjątek }  
  
:  
  
finally { kod, który wykona się niezależnie od  
    wystąpienia wyjątku }
```

# Przykład

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

# Przykład

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

# Przykład - niezalecany

```
int variable = 0;
try {
    variable = reader.nextInt();
}
catch (Exception ex) {
    System.out.print("Cos poszlo zle -
        "+ex.toString());
    variable = 5;
}
finally {
    System.out.print(variable); /* 5
        lub wpisana wartosc/*
}
```

- 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`).

## Tworzenie instancji klasy File

Klasa **File** wykorzystywana jest do przechowywania danych o pliku. Sposób 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
<code>exists()</code>	Zwraca wartość typu Boolean reprezentującą informację czy plik istnieje
<code>createNewFile ()</code>	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();
        }
    }
}
```

## 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`.

## Odczyt pliku

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

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

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

## 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**).

```
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();
}
```

- 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 FileReader

## Opis

Aby odczytać zawartość pliku tekstowego z wykorzystaniem **FileReader** należy go zaimportować:

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

Klasa traktuje plik jako strumień znaków.

# Przykład dla FileReader

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

- 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 `FileReader`.
- 3 Napisz metodę przyjmującą jako parametr ścieżkę pliku i zwracającą liczbę znaków znajdujących się w nim. Skorzystaj z `FileReader`.

# 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 character;
    while ((character = reader.read()) !=
        -1 && (char)character != 'k') {
        System.out.print((char) character);
    }
} 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 bufferedReader = new
        BufferedReader(reader);
    System.out.print(bufferedReader
        .readLine());
    bufferedReader.close();
}
```

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

## Inicjalizacja obiektu klasy `PrintWriter`

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

```
PrintWriter writer = 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));
    writer.println("test");
    writer.flush();
    writer.close();
}
```

# Przegląd najważniejszych metod

Metoda	Opis
<code>println(tekst)</code>	Wpisanie do pliku tekstu i zakończenie go znakiem nowej linii
<code>print(wartosc)</code>	Wpisanie do pliku wartości <code>wartosc</code>
<code>close()</code>	Zamknięcie pliku
<code>flush()</code>	Przesunięcie buforowanych danych do strumienia

## 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);
    writer.write("tekst");
    writer.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 bufferedWriter = new
        BufferedWriter(writer);
    bufferedWriter.write("tekst");
    bufferedWriter.flush();
    bufferedWriter.close();
}
```

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` `colle`. Niech metoda otwiera wskazany plik i pobierze jego  $n$ -tą linię ( $n=colle$ ). Następnie tekst ma zostać odwrócony (podpowiedź: skorzystaj ze `StringBuilder`) i wypisany na ekran.



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).

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