Podstawy i języki programowania

Strumienie, pliki.

Sortowanie.

Wyjątki.

Operacje I/O w Javie

Serializacja

- Zapisuje całą klasę
- Plik „binarny”
- Delimiter nieokreślony
- Nie da się podglądać☺

Pliki tekstowe

- Zapisuje wybrane informacje
- Plik tekstowy
- Delimiter ustawiamy sami
- Można podejrzeć

Porównanie typów plików

Serializacja

```
¬ìsrGameCharacter
¬%gê8MÛIpowerLjava/lang/
String;[weaponst[Ljava/lang/
String;xp2tlfur[Ljava.lang.String;≠“VÁ
È{Gxptbowtswordtdustsq~»tTrolluq~tb
are handstbig axsq~xtMagicianuq~tspe
llstinvisibility
```

Plain tekst

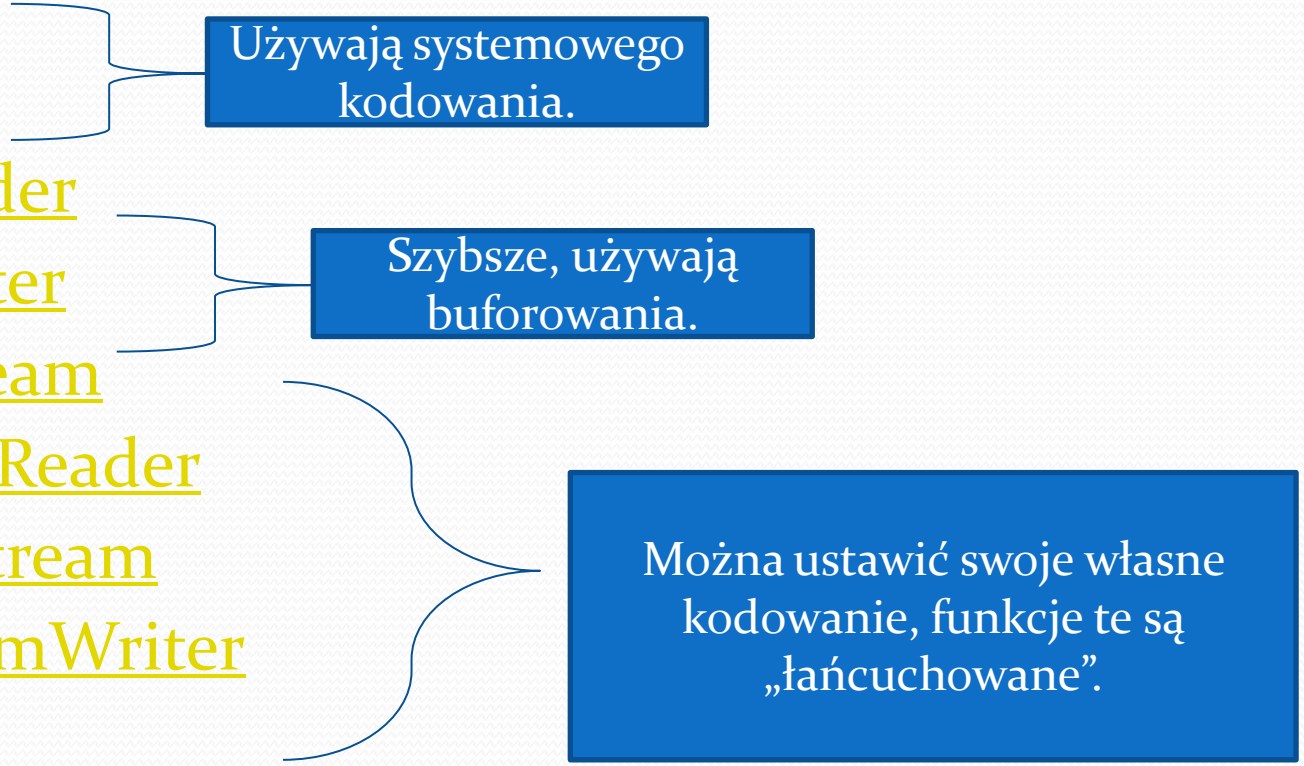
```
50,Elf,bow, sword,dust
200,Troll,bare hands,big ax
120,Magician,spells,invisibility
```

Ogólna konwencja pracy z plikami

1. Podaj ścieżkę do pliku w odpowiednim miejscu.
2. Zdefiniuj jaki plik zapisujesz/odczytujesz
3. Dokonaj operacji zapisu/odczytu
4. Zamknij plik.

W Javie wszystko jest obiektem…

- FileReader
- FileWriter
- BufferedReader
- BufferedWriter
- FileInputStream
- InputStreamReader
- FileOutputStream
- OutputStreamWriter
- File
- PrintWriter



Co to jest „łańcuchowanie”

```
BufferedReader in =
new BufferedReader(
new InputStreamReader(
„plik.txt”));
```

plik.txt jest odczytywany przez InputStreamReader; klasa ta odczytuje surowe dane (bajtowo); przekazuje bajty do kolejnej klasy – BufferedReader, która konwertuje bajty na znaki, a do tego – buforuje wejście

Try & catch

- Ryzykowne operacje zawiera się w blokach „try”
- Gdy coś złego się stanie – program wygeneruje wyjątek
- Ów wyjątek jest łapany (stąd „catch”) i może zostać „obsłużony”
- Dzięki temu program rzadziej się wysypuje☺
- Blok „finally” wykona się zawsze, niezależnie od błędów
- Literatura: http://www.javaworld.com/jw-07-1998/jw-07-exceptions.html

No to wreszcie gotowy przykład

```
1  package helloworld;
2  import java.io.*;
3
4
5  public class Main {
6
7
8  public static void main (String[] args) {
9  try {
10             File mójPlik = new File("plik_z_danymi.txt");
11             FileReader czytaczPliku = new FileReader(mójPlik);
12             BufferedReader czytacz = new BufferedReader(czytaczPliku);
13
14             String linijka = null;
15             while( (linijka = czytacz.readLine()) != null) {
16             System.out.println(linijka);
17             }
18             czytacz.close();
19             }
20 catch (Exception ex) {
21             System.out.println("Oups! Coś złego się stało");
               ex.printStackTrace();
23 }
24 finally {
25     System.out.println("Koniec programu");
26
27 }
28 }
29 }
30
```

Ćwiczenia

1. Napisz funkcję, która przyjmuje w parametrze ścieżkę do pliku, a następnie wyświetla na ekran co drugą linijkę.
2. Napisz funkcję, która odczytuje zawartość pliku, a następnie wypisuje plik w odwrotnej kolejności.
3. Ściągnij plik http://tjach.pl/downloads/parser.csv i napisz program, który wylicza średnią z podanych tam liczb.
   Podpowiedź: google: „String split java” oraz klasa Integer

Zapis pliku

```
1  package helloworld;
2  import java.io.*;
3
4  public class Main {
5  public static void main (String[] args) {
6  try {
7          File mójPlik = new File("plik_z_danymi2.txt");
8          FileWriter zapisywaczPliku = new FileWriter(mójPlik);
9          BufferedWriter zapisywacz = new BufferedWriter(zapisywaczPliku);
10         PrintWriter zapis = new PrintWriter(zapisywaczPliku);
11
12         String linijka = "Testowanie";
13         linijka+="\n";
14         linijka+="Druga";
15         linijka+=System.getProperty("line.separator");
16         zapis.write(linijka);
17         zapis.flush();
18         zapis.close();
19     }
20 catch (Exception ex) {
21         System.out.println("Oups! Coś złego się stało");
           ex.printStackTrace();
23 }
24 finally {
25     System.out.println("Koniec programu");
26
27 }
28 }
29 }
30
```

Kilka ważnych rzeczy

1. Pamiętaj, że buforowane wyjście należy czyścić, a pliki – zamykać!
2. Klasa PrintWriter , jak również wszystkie inne poznane, mają wiele użytecznych metod, warto je poznać.
3. Za pomocą klasy File możesz zmieniać strukturę katalogów: tworzyć, kasować, odczytywać.
4. FileReader może bezpośrednio odczytywać plik, ale robi to bardzo wolno.
5. Parsowanie pliku tekstowego jest zadaniem wymagającym wiedzy o tym jak on jest zbudowany.

Ćwiczenia

1. Wykorzystaj funkcje do generowanie dowolnej tablicy z poprzednich zajęć. Zapisz trzy dowolne tablice, co najmniej 5x5 do jednego pliku.
2. Zapisz tabliczkę mnożenia do pliku.
3. Za pomocą klasy File stwórz nowy katalog i skopiuj stworzony przedtem plik tekstowy do nowego katalogu.
4. Wykorzystaj plik „parser.txt”. Odczytaj wszystkie liczby z pliku, a następnie zapisz je do nowego pliku w kolejności rosnącej.

Ćwiczenia

1. W akwarium znajdują się rozwielitki (małe skorupiaki, zwane także dafniami). W chwili t = 0 ilość rozwielitek wynosiła 1000. Po każdej jednostce czasu ilość rozwielitek podwaja się. Ponieważ jednak akwarium ma ograniczone rozmiary, więc zbyt wielka ilość rozwielitek powoduje ich wymieranie. Przyjmiemy, że jeśli ilość żyjątek przekroczy 50 tysięcy, wtedy w ciągu najbliższej jednostki czasu rozwielitki nie rozmnażają się, lecz ginie 99% ich populacji.
    1. Oblicz, ile będzie rozwielitek w akwarium po upływie 100 jednostek czasu.
    2. Znajdź największą i najmniejszą ilość rozwielitek, jaka znajdowała się w akwarium w okresie rozważanym w poprzednim podpunkcie.
    3. Utwórz tabelkę obrazującą zależność liczby rozwielitek od czasu w okresie od 0 do 25 jednostek. Tabelkę umieść w pliku tekstowym.

Ćwiczenia

1. Oblicz liczbę znaków zawartych w pliku tekstowym.
2. Napisać program, który pozwoli obliczyć liczbę wierszy tekstu. Przeglądane powinny być wszystkie znaki tekstu, a obliczanie liczby wierszy powinno odbywać się dopiero w momencie zakończenia przeglądania danego wiersza.
3. Napisać program obliczający liczbę słów w pliku tekstowym zakładając, że poszczególne słowa są oddzielone spacjami, tabulatorami lub znakami końca linii.
4. Napisać program porównujący zawartość 2 plików tekstowych. Algorytm powinien polegać na sprawdzeniu równości znaków do końca krótszego pliku, a następnie w przypadku pozytywnego wyniku na sprawdzeniu długości plików.

Program na zaliczenie

Konwerter liczb

- Zamiana dziesiętnego na dwójkowy.
- Zamiana dziesiętnego na dowolny o niższej podstawie.
- Zamiana dwójkowego na dziesiętny.
- Zamiana dwójkowego na dowolny.
- Zamiana dowolnego na dziesiętny.

Wymagania:

1. Każde z zadań powinno być oddzielną funkcją.
2. Program musi umożliwiać pracę (odczyt/zapis) na plikach tekstowych oraz serializację obiektów.
3. Każda konwersja będzie nową instancją klasy.
4. Program powinien umożliwiać również wczytywanie danych z klawiatury.
5. Program ma automatycznie przy włączeniu wczytywać ustawiony wcześniej plik z konwersjami (każda konwersja będzie oznaczana jedną linijką:
   10 -> 2 \tab 5, co należy rozumieć „konwersja liczby 5 z formatu dziesiętnego na dwójkowy).
6. Wszystkie pliki z danymi mają mieć prefiks „WEJ_”, wszystkie wyjściowe „WYJ_”.

Program na zaliczenie 2

Wymagania c.d.

1. Program ma działać dopóki użytkownik tego chce, nie zamykać się po każdej konwersji.
2. Dodatkowo program ma umożliwiać przeprowadzanie prostych operacji liczbowych na liczbach o dowolnym systemie liczenia.
3. Konwersja między plikami tekstowymi a serializowanymi obiektami.
4. Obliczanie liczby wykonywanych konwersji w przypadku odczytu z pliku.
5. Zamiana wynikowego systemu liczenia dla wszystkich elementów w pliku.
6. Sortowanie pliku wyjściowego: rosnąco i malejąco.
7. Informacja o autorze programu
8. Oceniana będzie zarówno skuteczność działania, struktura i przejrzystość kodu oraz komentarze.
9. Zamiana liczb na ich odpowiedniki słowne (1 = „Jeden”, 215 = „Dwieście piętnaście”).
10. Zamiana liczb na liczby rzymskie