Scanner

SPIS TREŚCI

Wstęp

Programy komputerowe służą do przetwarzania dostarczonych danych i generowania "informacji" użytecznych dla użytkownika. Dane te pochodzą z różnych źródeł, są różnego rodzaju i są różnej wielkości. W przypadku gdy są krótkimi tekstami lub pojedynczymi wartościami mogą być wprowadzane ręcznie za pomocą klawiatury, natomiast gdy danych jest więcej muszą być one odczytywane bezpośrednio z zasobów w których są składowane. Istnieją różnorodne sposoby przechowywania danych, najprostszym z nich — wbrew pozorom powszechnie stosowanym — jest przechowywanie danych w zwykłych plikach tekstowych. Co więcej wiele systemów przechowywania danych umożliwia eksportowanie danych do plików tekstowych, np. w formacie CSV, z którym możemy je pobierać do analizy. Pliki te mogą być dostępne zarówno na lokalnym komputerze, jak i zdalnie przez sieć komputerową.

Java traktuje wszystkie dane jako strumienie danych. Obsługa tych strumieni nie jest sprawą trywialną dla osób początkujących, dlatego też dobrą alternatywą jest użycie dostępnych metod klasy Scanner, które wykonają to za użytkownika. Tak więc można traktować tę klasę jako narzędzie upraszczające programowanie obsługi wejścia, czyli pobierania danych do programów z zewnętrznych źródeł.

Wczytywanie pojedynczych wartości z klawiatury

Aby skorzystać z dostępnych metod klasy Scanner należy w pierwszej kolejności stworzyć obiekt tej klasy wskazując źródło z którego dane będą odczytywane.

```
Scanner sc = new Scanner(System.in);
```

W tym przypadku źródłem będzie klawiatura komputera. System.in oznacza standardowe wejście, czyli podstawowy kanał komunikacji programów z systemem operacyjnym.

Tak więc aby pobrać z klawiatury ciąg znaków i przypisać do jakiejś zmiennej w programie — w tym przypadku s — należy użyć metody next().

```
String s = sc.next();
```

Z kolei aby pobrać z konsoli liczbę całkowitą należy użyć metody nextInt();

```
int rok = sc.nextInt();
```

W tym przypadku pobrany z konsoli ciąg znaków konwertowany jest do zmiennej typu int.

Z konsoli oczywiście można też pobierać wartości zmiennoprzecinkowe, np. typu double. Służy do tego metoda nextDouble().

```
double cena = sc.nextDouble();
```

Działa to podobnie jak we wcześniejszych przypadkach, należy w tym miejscu tylko zwrócić uwagę na to, że domyślnie separatorem części całkowitej i ułamkowej w tym przypadku będzie znak przecinka, w przeciwieństwie do literału znakowego reprezentującego wartość zmiennoprzecinkową, w którym separatorem części całkowitej i ułamkowej jest znak kropki.

Klasa skaner posiada także metody pozwalające na pobieranie danych innych typów podstawowych. Są to m. in. nextByte(), nextShort(), nextLong(), nextBoolean(), nextFloat(), nextBigInteger(), nextBigDecimal().

Warto też zwrócić uwagę na metodę nextLine() — pozwalającą na wczytanie całych linii tekstu łącznie z białymi znakami — z których można wydzielić indywidualne wartości, określonych typów.

Wczytywanie wielu wartości z pliku

Większe ilości danych można szybko wczytać z pliku. W tym celu w pierwszej kolejności należy stworzyć obiekt klasy Scannera powiązany z konkretnym plikiem na dysku.

```
File plik = new File("plik.txt");

Scanner sc = new Scanner(plik);
```

Od tej chwili można rozpocząć wczytywanie danych w sposób zasadniczo nie różniący się od tego co napisano powyżej. Faktyczna różnica pojawia się w związku z chęcia odczytania nie pojedynczej ale bardzo wielu wartości. W tym celu konieczne z jednej strony jest

zadeklarowanie pętli w której będzie następowało odczytywanie danych, a z drugiej strony w której odczytane dane będą zapisywane do pamięci komputera.

Wygodnym sposobem deklarowania pętli powiązanej z obiektem klasy Scanner jest deklaracja while. Za jej pomocą można wczytać wszystkie wartości zawarte w pliku.

```
File plik = new File("plik.txt");
Scanner sc = new Scanner(plik);

String s;

while(sc.hasNext())

{
     s = sc.next();

     // dalsze operacje
}
```

W tym przypadku odczytywane są z pliku wszystkie ciągi znaków i zapisywane w zmiennej s, która następnie jest dalej przetwarzana.

W podobny sposób można odczytywać wartości liczbowe, np. całkowite.

```
File plik = new File("plik.txt");
Scanner sc = new Scanner(plik);

int liczba;

while(sc.hasNextInt())

{
     liczba = sc.nextInt();

     // dalsze operacje
}
```

W rzeczywistości pliki tekstowe w pojedynczych liniach zapisują wartości różnego typu, tj. zarówno ciągi znaków jak i wartości liczbowe. Aby odczytać i prawidłowo zinterpretować takie wartości należy w pierwszej kolejności odczytuje się całe linie, następne dzieli się linie na indywidualne wartości, a następnie konwertuje się je na konkretne typu.

Zakładając, że w pojedynczej linii pliku plik.txt zapisane są trzy wartości — w tym jeden ciąg znaków, jedna wartość całkowita i jedna wartość typu double — można je pobrać i odczytać np. w następujący sposób.

```
File plik = new File("plik.txt");
Scanner sc = new Scanner(plik);
```

```
String linia;

while(sc.hasNextLine())

{
     linia = sc.nextLine();

     String[] dane = linia.split();

     int liczbaC = Integer.parseInt(dane[1]);

     double liczbaD = Double.parseDouble(dane[2]);

     // dalsze operacje
}
```

Problem przecinka czyli kropki

W języku polskim inaczej niż w języku angielskim w zapisie liczb rzeczywistych separatorem części dziesiętnej i części ułamkowej jest znak przecinka. Aby poprawnie przekonwertować łańcuch znaków na typ zmiennoprzecinkowy za pomocą metody Double.parseDouble(String s) najprościej zamienić występujący w łańcuchu przecinek na kropkę za pomocą metody replace(String s1, String2) klasy String.

```
String s = "123,456";

s = s.replace(",",".");

double liczba = Double.parseDouble(s);
```

Delimiter czyli separator

Domyślnym separatorem wartości (tokenów) odczytywanych z pliku są znaki końca linii lub znaki spacji. Skaner za każdym razem odczytuje tylko jeden, następny "token" który może znajdować się w tej samej linii oddzielony od poprzedniego znakiem spacji lub następnej. Jednak w wielu przypadkach dane zapisane w plikach tekstowych nie są oddzielone znakami spacji lecz innymi — zwykle są to znaki przecinka[1] albo średnika. Aby odczytać poprawnie takie pliki należy poinformować o tym Skaner czego dokonuje sie poprzez użycie metody useDelimiter(String d). Na przykład aby zmienić standardową spację na znak przecinka należy napisać.

```
File plik = new File("plik.txt");
Scanner sc = new Scanner(plik);

sc.useDelimiter(",");
```

[1] W języku angielskim przecinek nie jest separatorem w między częścią całkowitą a ułamkową w liczbach rzeczywistych.

A następnie postępować jak poprzednio.

Odczytywanie danych z Sieci

Scanner pozwala na odczytywanie danych dostępnych przez sieć komputerową, ze względu na fakt, że strumienie sieciowe są znakowe.

Poniższy przykład pokazuje jak łatwo można pobierać z Sieci np. całe strony internetowe,

```
URL url = new URL("https://onet.pl");

Scanner sc = new Scanner(url.openStream());

while(sc.hasNext()) System.out.println(sc.next());
```

które po pobraniu można np. przeszukiwać pod kątem występowania określonych informacji albo odnośników do innych zasobów — stron albo multimediów — w Sieci.

Zamykanie Scannera

Skaner zasadniczo należy zamknąc po uzyciu. Dokonujemy tego przez użycie metody close().

```
File plik = new File("plik.txt");
Scanner sc = new Scanner(plik);

sc.close();
```

Podsumowanie

Klasa Scanner jest uniwersalnym i prostym w użyciu narzędziem do pobierania danych do tworzonych przez nas programów.

Ciekawostki

1. System.in z punktu widzenia języka Java to obiekt klasy BufferedInputStream.

Biblioteka

1. **Oficjalna dokumentacja klasy Scanner**
   Scanner (Java SE 14 & JDK 14)

2. **Standardowe wejście, wyjście i kanał błędów**
   Standard Input and Standard Output | Linux Shell