Wprowadzenie do Javy - materiały na lab1

Operacje wyjścia na konsolę

W języku Java operacje wyjścia na konsolę odnoszą się głównie do wydruków tekstowych prezentowanych użytkownikowi na standardowym wyjściu (czyli zazwyczaj konsoli terminala).

Poniżej kilka podstawowych opcji:

1. **System.out.print()** i **System.out.println()**:

   - `System.out.print("Tekst")`: Drukuje podany tekst na konsoli bez przechodzenia do nowej linii.
   - `System.out.println("Tekst")`: Drukuje podany tekst na konsoli i przechodzi do nowej linii.

   Przykład:

   ```
   System.out.print("Witaj, ");
   System.out.println("świecie!");
   ```

   Wyjście:

   ```
   Witaj, świecie!
   ```

2. **System.out.printf()**: Pozwala na formatowane wyjście. Umożliwia wstawianie zmiennych w określone miejsca w tekście oraz kontrolowanie sposobu ich prezentacji.

   Przykład:

   ```
   double cena = 12.5;
   int ilosc = 5;
   System.out.printf("Cena: %.2f, Ilość: %d, Łącznie: %.2f", cena, ilosc, cena * ilosc);
   ```

   Wyjście:

   ```
   Cena: 12.50, Ilość: 5, Łącznie: 62.50
   ```

Metoda `System.out.printf()` w Javie oraz funkcja `printf()` w języku C są do siebie podobne pod względem ogólnego przeznaczenia i użycia, ponieważ obie służą do formatowanego wydruku napisów i są inspirowane tym samym konceptem. Niemniej jednak, istnieje kilka różnic w używaniu tych funkcji/metod, zarówno pod względem składni, jak i funkcjonalności, które mogą wpłynąć na to, jak są używane w obu językach.

Operacje wejścia z konsoli

Klasa `Scanner` jest często używana do wczytywania danych wejściowych z konsoli, ponieważ jest łatwa w użyciu i oferuje szeroki zakres funkcji do wczytywania różnych typów danych.

1. **Inicjalizacja**: Aby użyć klasy `Scanner`, najpierw musisz ją zaimportować z pakietu `java.util`, a następnie utworzyć jej obiekt, używając `System.in` jako źródła wejściowego.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
    }
}
```

2. **Wczytywanie danych**:

- **Wczytywanie całego wiersza tekstu**:

```
String line = scanner.nextLine();
```

- **Wczytywanie pojedynczego słowa**:

```
String word = scanner.next();
```

- **Wczytywanie danych liczbowych**:

```
int number = scanner.nextInt();       // dla int
double value = scanner.nextDouble();  // dla double
float valueFloat = scanner.nextFloat(); // dla float
```

- **Wczytywanie wartości logicznych**:

```
boolean isTrue = scanner.nextBoolean();
```

3. **Zamykanie skanera**: Po zakończeniu wczytywania danych z konsoli warto zamknąć obiekt `Scanner`, aby zwolnić zasoby.

```
scanner.close();
```

Podstawowe typy danych

Java oferuje zestaw typów podstawowych (nazywanych też typami prostymi), które są używane do reprezentacji prostych wartości, takich jak liczby całkowite, liczby zmiennoprzecinkowe, znaki czy wartości logiczne. Oto one:

1. **Liczby całkowite**:
   - `byte`: 8-bitowa wartość liczby całkowitej; zakres od -128 do 127.
   - `short`: 16-bitowa wartość liczby całkowitej; zakres od $-32,768$ do $32,767$.
   - `int`: 32-bitowa wartość liczby całkowitej; zakres od -2^{31} do $2^{31}-1$.
   - `long`: 64-bitowa wartość liczby całkowitej; zakres od -2^{63} do $2^{63}-1$.
2. **Liczby zmiennoprzecinkowe**:
   - `float`: 32-bitowa wartość liczby zmiennoprzecinkowej (jednokrotnej precyzji). Zaleca się używanie sufiksu `F` lub `f` przy inicjalizacji wartości stałych.
   - `double`: 64-bitowa wartość liczby zmiennoprzecinkowej (podwójnej precyzji). Jest to domyślny typ dla wartości zmiennoprzecinkowych w Javie.
3. **Znak**:
   - `char`: Reprezentuje pojedynczy znak w standardzie Unicode i zajmuje 16 bitów. Znaki są zamykane w pojedynczych cudzysłowach, np. `'A'`.
4. **Wartość logiczna**:
   - `boolean`: Reprezentuje wartość prawda/fałsz. Może przyjmować tylko jedną z dwóch wartości: `true` lub `false`.

Warto zwrócić uwagę na kilka rzeczy:

- Typy podstawowe w Javie są zawsze o stałej wielkości, niezależnie od platformy.
- Każdy z tych typów podstawowych (oprócz `boolean`) ma odpowiadającą mu klasę opakowującą w pakiecie `java.lang` (np. `Integer` dla `int`, `Double` dla `double` itp.). Klasy te są używane, gdy potrzebujemy reprezentować typ podstawowy jako obiekt, oraz oferują wiele pomocniczych metod do pracy z danym typem.
- Inicjalizacja zmiennych typów prostych zawsze nadaje im domyślną wartość (np. `0` dla typów liczbowych, `false` dla `boolean`), ale wartości te są ustawiane domyślnie tylko dla zmiennych klasowych lub instancyjnych, nie dla lokalnych zmiennych wewnątrz metod.

```java
public class Main {
    public static void main(String[] args) {
        int localVar; // zmienna lokalna niezainicjalizowana

        // Poniższa linia spowoduje błąd kompilacji, ponieważ próbujemy użyć zmiennej,
        // która nie została zainicjalizowana
        // System.out.println(localVar);

        localVar = 10; // inicjalizacja zmiennej lokalnej
        System.out.println(localVar); // teraz jest w porządku, ponieważ zmienna
        // została zainicjalizowana
    }
}
```

Statyczna metoda `main`

W Javie metoda `main` jest specjalną metodą, która służy jako punkt wejścia dla większości aplikacji konsolowych. To właśnie ta metoda jest wywoływana, gdy uruchamiasz program przy użyciu komendy `java NazwaKlasy`.