Wprowadzenie do Javy - materiały na lab2

Generowanie liczb pseudolosowych

lasa `Random` w Javie służy do generowania strumieni pseudolosowych liczb. Jest ona częścią pakietu `java.util`.

Oto kilka najczęściej używanych metod klasy `Random`:

1. **nextInt()**: Zwraca losową liczbę całkowitą (typu `int`).
   - `nextInt()`: zwraca dowolną wartość `int`.
   - `nextInt(int bound)`: zwraca wartość od 0 (włącznie) do podanej wartości (wyłącznie).
2. **nextLong()**: Zwraca losową liczbę całkowitą typu `long`.
3. **nextDouble()**: Zwraca losową liczbę zmiennoprzecinkową typu `double` z zakresu od 0.0 (włącznie) do 1.0 (wyłącznie).
4. **nextFloat()**: Zwraca losową liczbę zmiennoprzecinkową typu `float` z zakresu od 0.0 (włącznie) do 1.0 (wyłącznie).
5. **nextBoolean()**: Zwraca losową wartość `boolean`, czyli `true` lub `false`.
6. **nextBytes(byte[] bytes)**: Wypełnia podaną tablicę bajtów losowymi bajtami.
7. **nextGaussian()**: zwraca losową liczbę zmiennoprzecinkową typu `double` wygenerowaną według standardowego rozkładu normalnego (rozkładu Gaussa)o średniej równej 0 i odchyleniu standardowym równym 1.
8. **setSeed(long seed)**: Ustawia ziarno generatora liczb losowych. Jeśli użyjesz tego samego ziarna dla dwóch różnych instancji klasy `Random`, będą one generować te same sekwencje liczb. To jest przydatne, gdy chcesz uzyskać powtarzalne wyniki dla testów czy symulacji.

```
import java.util.Random;
```

```
public class Example {
    public static void main(String[] args) {
        Random random = new Random();
        System.out.println("Losowa liczba całkowita: " + random.nextInt());
    }
}
```

Tablice jednowymiarowe

Tablice jednowymiarowe to struktury danych, które przechowują wiele wartości tego samego typu w jednym obiekcie.

Deklaracja: Aby zadeklarować tablicę, używa się typu danych, jaki ma być przechowywany, a następnie nawiasów kwadratowych. Na przykład, aby zadeklarować tablicę liczb całkowitych, używa się:

```
int[] mojaTablica;
```

Inicjalizacja: Można zainicjować tablicę w kilka sposobów: - Przy pomocy operatora `new` oraz określenia rozmiaru tablicy:

```
mojaTablica = new int[5]; // tablica z 5 elementami typu int
```

- Przy pomocy literału tablicowego:

```
int[] innaTablica = {1, 2, 3, 4, 5}; // tablica zawierająca 5 liczb całkowitych
```

Dostęp do elementów: Aby uzyskać dostęp do elementu tablicy, używa się indeksu (numerowane od 0) w nawiasach kwadratowych:

```
int pierwszyElement = mojaTablica[0]; // uzyskanie dostępu do pierwszego elementu
```

Modyfikacja elementów: Można modyfikować zawartość tablicy, odnosząc się do konkretnego indeksu:

```
mojaTablica[0] = 42; // przypisanie wartości 42 do pierwszego elementu tablicy
```

Długość tablicy: Aby uzyskać długość tablicy, używa się atrybutu `length`:

```
int dlugosc = mojaTablica.length;
```

Ograniczenia:

- Tablice w Javie mają stałą długość, co oznacza, że po ich utworzeniu nie można zmieniać ich rozmiaru.
- Wszystkie elementy tablicy muszą być tego samego typu.

Przypisanie:

W Javie możesz przypisać jedną tablicę do innej zmiennej tablicowej, ale ważne jest zrozumienie, co się wtedy dzieje.

Gdy przypisujesz tablicę do innej zmiennej tablicowej, nie tworzysz nowej kopii tej tablicy. Zamiast tego tworzysz drugą referencję do tej samej tablicy w pamięci. Oznacza to, że obie zmienne wskazują na tę samą tablicę, a zmiany dokonane za pomocą jednej zmiennej będą widoczne przy użyciu drugiej zmiennej.

```
int[] tablica1 = {1, 2, 3};
int[] tablica2 = tablica1;

tablica2[0] = 42;

System.out.println(tablica1[0]); // wydrukuje "42", a nie "1"
```

W powyższym przykładzie, po przypisaniu `tablica1` do `tablica2`, obie zmienne wskazują na tę samą tablicę. Dlatego zmiana wartości w `tablica2` wpływa również na `tablica1`.

Jeśli chcesz mieć dwie różne kopie tej samej tablicy, musisz skopiować zawartość jednej tablicy do drugiej, na przykład za pomocą pętli lub używając metody `System.arraycopy()`:

```
int[] tablica1 = {1, 2, 3};
int[] tablica2 = new int[tablica1.length];
System.arraycopy(tablica1, 0, tablica2, 0, tablica1.length);
```

Teraz `tablica2` jest osobną kopią `tablica1`, a zmiany w jednej z nich nie wpłyną na drugą.

Przekazywanie tablic do metody:

W Javie argumenty są zawsze przekazywane przez wartość. Mamy do czynienia z kopią referencji.

```
public class TestTablicy {

    public static void main(String[] args) {
        int[] tablica = {1, 2, 3};
        modyfikujTablice(tablica);
```

```
        System.out.println(tablica[0]); // Wydrukuje "42", a nie "1"
    }

    public static void modyfikujTablice(int[] arr) {
        arr[0] = 42;
    }
}
```

Metody "systemowe" do obsługi tablic

Dokumentacja: https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Arrays.html

W Javie do obsługi tablic dostępna jest klasa `java.util.Arrays`, która dostarcza szereg przydatnych metod. Oto niektóre z nich:

1. **`sort`**: Sortuje elementy tablicy. Może być używane do sortowania całej tablicy lub tylko jej części.
2. **`binarySearch`**: Wyszukuje określony element w posortowanej tablicy, używając wyszukiwania binarnego.
3. **`equals`**: Sprawdza, czy dwie tablice są równe pod względem zawartości.
4. **`fill`**: Ustala wszystkie elementy tablicy na określoną wartość.
5. **`copyOf`**: Kopiuje określoną liczbę elementów tablicy źródłowej do nowej tablicy.
6. **`copyOfRange`**: Kopiuje zakres elementów z tablicy źródłowej do nowej tablicy.
7. **`deepEquals`**: Sprawdza, czy dwie tablice wielowymiarowe są równe pod względem zawartości.
8. **`deepHashCode`**: Oblicza kod hash dla tablicy, biorąc pod uwagę wszystkie elementy w tablicach wielowymiarowych.
9. **`deepToString`**: Konwertuje tablicę wielowymiarową na łańcuch reprezentujący jej zawartość (w sposób głęboki).
10. **`hashCode`**: Oblicza hash dla tablicy.
11. **`stream`**: Dostarcza sekwencyjny strumień zawierający elementy tablicy, co pozwala na używanie operacji strumieniowych (od Javy 8).
12. **`setAll`**: Ustala wartości wszystkich elementów tablicy, używając dostarczonej funkcji generującej.

13. **`parallelSetAll`**: Podobnie jak **`setAll`**, ale używa wielowątkowości do jednoczesnego ustalania wartości dla wielu elementów.
14. **`parallelSort`**: Sortuje tablicę używając wielowątkowości.
15. **`mismatch`**: Znajduje i zwraca indeks pierwszego niezgodnego elementu między dwiema tablicami (od Javy 9).
16. **`toString`**: Konwertuje tablicę wielowymiarową na łańcuch reprezentujący jej zawartość (w sposób płytki).

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] tablica = {34, 12, 5, 78, 2, 10, 8};
        Arrays.sort(tablica);
        System.out.println(Arrays.toString(tablica));
    }
}
```

```java
import java.util.Arrays;

public class SortowanieNierosnace {
    public static void main(String[] args) {
        int[] tablica = {34, 12, 5, 78, 2, 10, 8};
        Arrays.sort(tablica);
        for (int i = 0; i < tablica.length / 2; i++) {
            int temp = tablica[i];
            tablica[i] = tablica[tablica.length - 1 - i];
            tablica[tablica.length - 1 - i] = temp;
        }
        System.out.println(Arrays.toString(tablica));
    }
}
```

```java
import java.util.Arrays;

public class PorownanieTablic {
    public static void main(String[] args) {
        int[] tablica1 = {1, 2, 3, 4, 5};
        int[] tablica2 = {1, 2, 3, 4, 5};
        int[] tablica3 = {5, 4, 3, 2, 1};
```

```
        boolean czyRowne1i2 = Arrays.equals(tablica1, tablica2); // true
        boolean czyRowne1i3 = Arrays.equals(tablica1, tablica3); // false
        System.out.println("Czy tablica1 jest równa tablica2? " + czyRowne1i2);
        System.out.println("Czy tablica1 jest równa tablica3? " + czyRowne1i3);
    }
}
```

```
import java.util.Arrays;

public class WypelnienieTablicy {
    public static void main(String[] args) {
        int[] tablica = new int[5];
        Arrays.fill(tablica, 10);
        System.out.println(Arrays.toString(tablica));
    }
}
```

```
import java.util.Arrays;

public class KonwersjaTablicyNaString {
    public static void main(String[] args) {
        int[] tablica = {1, 2, 3, 4, 5};
        String reprezentacjaTablicy = Arrays.toString(tablica);
        System.out.println(reprezentacjaTablicy);
    }
}
```

```
import java.util.Arrays;

public class KopiowanieTablicy {
    public static void main(String[] args) {
        int[] oryginalnaTablica = {1, 2, 3, 4, 5};
        int[] nowaTablica = Arrays.copyOf(oryginalnaTablica, 3);
        System.out.println(Arrays.toString(nowaTablica));
    }
}
```