|  |  |  |  |
| --- | --- | --- | --- |
| ***SPRAWOZDANIE NR 1*** | | | |
| Nazwa ćwiczenia | lambda | | Obraz zawierający tekst  Opis wygenerowany automatycznie |
| Przedmiot | Zawansowane programowanie obiektowe | |
| Student  grupa | Paweł Jońca gr 7 122348 | |
| Data ćwiczeń | 24.03.2025r | 24.03.2025r | Data oddania sprawozdania |

**Wyrażenia lambda** w Javie to krótkie, uproszczone sposoby na definiowanie funkcji w postaci anonimowej. Służą głównie do implementacji interfejsów funkcyjnych, czyli takich, które mają tylko jedną metodę abstrakcyjną.

Zastosowanie lambdy:

Kompaktowość i czytelność - Lambdy pozwalają pisać bardziej zwięzły kod, eliminując konieczność tworzenia osobnych klas anonimowych.

Przetwarzanie kolekcji - Świetnie sprawdzają się przy użyciu strumieni (Stream API) w operacjach takich jak filtrowanie, mapowanie czy sortowanie.

Programowanie funkcyjne - Lambdy ułatwiają stosowanie wzorców programowania funkcyjnego, które promuje użycie funkcji jako obiektów pierwszej klasy.

Obsługa zdarzeń -Mogą być używane w aplikacjach GUI (np. JavaFX) do obsługi kliknięć czy innych zdarzeń.

Przekazywanie logiki jako parametr – Dzięki lambdom można przekazywać logikę działania jako argumenty do metod.

Prosta Lambda – Użyta do wykonania prostej operacji drukowania tekstu

Obraz zawierający tekst, zrzut ekranu, Czcionka, Oprogramowanie multimedialne

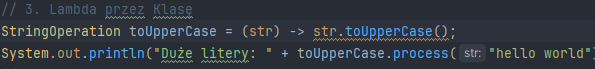
Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Blokująca lambda - Zawiera wieloliniowy blok kodu, wykonuje operację dodawania i wypisuje wynik

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Lambda przez Klasę – Przekształca tekst na wielkie litery (toUpperCase.process("hello world")).



Referencyjna Lambda (Metoda)



Odwołanie do Konstruktora

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Wnioski:

Wyrażenia lambda w Javie są przydatne, ponieważ pozwalają na skrócenie i uproszczenie kodu, szczególnie przy implementacji interfejsów funkcyjnych. Dzięki nim łatwiej można wykonywać operacje na danych, obsług iwać zdarzenia i korzystać z nowoczesnych rozwiązań programowania funkcyjnego.

Kod:

interface SimpleInterface {  
 void display();  
}  
interface MathOperation {  
 int operate(int a, int b);  
}  
interface StringOperation {  
 String process(String str);  
}  
  
interface ObjectCreator {  
 MyData create(String data);  
}  
class MyData {  
 private String data;  
  
 MyData(String data) {  
 this.data = data;  
 }  
 public String getData() {  
 return data;  
 }  
}  
public class LambdaExamples {  
 public static void main(String[] args) {  
 // 1. Prosta Lambda  
 SimpleInterface simpleLambda = () -> System.out.println("Zrobiona prosta lambda!");  
 simpleLambda.display();  
 // 2. Blokująca Lambda  
 MathOperation addition = (a, b) -> {  
 int result = a + b;  
 System.out.println("Wynik dodawania: " + result);  
 return result;  
 };  
 addition.operate(10, 20);  
 // 3. Lambda przez Klasę  
 StringOperation toUpperCase = (str) -> str.toUpperCase();  
 System.out.println("Duże litery: " + toUpperCase.process("hello world"));  
 // 4. Referencyjna Lambda (Metoda)  
 StringOperation reverse = LambdaExamples::reverseString;  
 System.out.println("Odwrócony tekst: " + reverse.process("Java"));  
 // 5. Odwołanie do Konstruktora  
 ObjectCreator creator = MyData::new;  
 MyData myData = creator.create("Wiadomość z konstruktora");  
 System.out.println("Dane: " + myData.getData());  
 }  
 public static String reverseString(String str) {  
 return new StringBuilder(str).reverse().toString();  
 }  
}