


SPRAWOZDANIE NR 1			
Nazwa ćwiczenia	lambda		 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki
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 lambda:

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 lambda można przekazywać logikę działania jako argumenty do metod.

Prosta Lambda – Użyta do wykonania prostej operacji drukowania tekstu

```
SimpleInterface simpleLambda = () -> System.out.println("Zrobiona prosta lambda!");
simpleLambda.display();

SimpleInterface simpleLambda = () -> System.out.println("Zrobiona prosta lambda!")
```

lambda

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

```
MathOperation addition = (a, b) -> {
    int result = a + b;
    System.out.println("Wynik dodawania: " + result);
    return result;
};
addition.operate(a: 10, b: 20);
```

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

```
// 3. Lambda przez Klasę
StringOperation toUpperCase = (str) -> str.toUpperCase();
System.out.println("Duże litery: " + toUpperCase.process(str: "hello world"));
```

Referencyjna Lambda (Metoda)

```
StringOperation reverse = LambdaExamples::reverseString;
System.out.println("Odwrócony tekst: " + reverse.process(str: "Java"));
// 5. Odwołanie do Konstruktor
```

Odwołanie do Konstruktor

```
// 3. Odwołanie do Konstruktor
ObjectCreator creator = MyData::new;
MyData myData = creator.create("Wiadomość z konstruktora");
System.out.println("Dane: " + myData.getData());
```

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, obsługiwać 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 Konstruktor
        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();
    }
}

```