**MINISTERUL EDUСAŢIEI ŞI СERСETĂRII AL REPUBLIСII MOLDOVA**

**UNIVERSITATEA DE STAT „ALEСU RUSSO” DIN BĂLŢI**

**FAСULTATEA DE ŞTIINŢE REALE, EСONOMIСE ȘI ALE MEDIULUI**

**СATEDRA DE MATEMATIСĂ ȘI INFORMATIСĂ**

**LUCRARE DE LABORATOR NR. 4**

**„Lambda Expresii”**

**Curs „Platforma JAVA Enterprise II”**

**Coordonator:**

**Olga CERBU**

dr.,conf.univ

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**Autor:**

Studentul grupei AW21M

**Dumitru BAZAOСHI**

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**BĂLȚI, 2023**

**CUPRINS**

[Scop. Condiția problemei 2](#_Toc149056436)

[Ce sunt expresiile Lambda? 2](#_Toc149056437)

[Caracteristicile expresiilor lambda 2](#_Toc149056438)

[Sintaxa expresiilor și exemplu 2](#_Toc149056439)

[Cod Java – Clasa Main 3](#_Toc149056440)

[Cod Java – Clasa LambdaApp 3](#_Toc149056441)

[Cod Java – Clasa ReferinteApp 4](#_Toc149056442)

[Rezultate Consolă 5](#_Toc149056443)

[Concluzii 5](#_Toc149056444)

[Referințe bibliografice 5](#_Toc149056445)

Scop. Condiția problemei

Scopul acestei lucrării de laborator legată de expresiile lambda este de familiarizarea cu conceptul de expresii lambda în programarea orientată obiect, în special în contextul limbajelor de programare care suportă funcționalități lambda. De asemenea de a obține o înțelegere profundă și practică a expresiilor lambda și pregătirea în utilizarea acestui concept în proiecte de dezvoltare software.

Ce sunt expresiile Lambda?

Expresiile lambda, cunoscute și sub numele de funcții lambda sau lambda abstracții, sunt o caracteristică fundamentală a programării funcționale și a unor limbaje de programare orientate către funcții. Acestea reprezintă funcții anonime, adică funcții fără un nume specific. Expresiile lambda pot fi definite și utilizate în linie, fără a fi necesară o definiție separată a funcției.

Caracteristicile expresiilor lambda

Printre caracteristicile expresiilor lambda sunt:

* **Anonimă**: Expresiile lambda nu au un nume asociat și pot fi definite și utilizate în același loc în care sunt necesare.
* **Scurte și Concise**: Sunt de obicei scrise într-o sintaxă concisă și ușor de înțeles, concentrându-se pe esențialul funcției.
* **Utilizare Flexibilă**: Pot fi folosite pentru a înlocui funcțiile tradiționale atunci când este necesară o funcționalitate simplă și nu este necesar un nume pentru funcție.
* **Parametrii și Corpul Funcției**: Include parametrii funcției și un corp care definește operațiile efectuate de funcție.

Sintaxa expresiilor și exemplu

**Sintaxa**: (parametri) -> { corpul functiei }

**Exemplu**: (x, y) -> x + y

În acest exemplu, expresia lambda primește doi parametri (x și y) și returnează suma lor.

Cod Java – Clasa Main

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<Integer> numbers = new ArrayList<Integer>(); // Crearea unei liste de numere întregi

numbers.add(5); // Adăugarea numărului 5 în listă

numbers.add(9); // Adăugarea numărului 9 în listă

numbers.add(8); // Adăugarea numărului 8 în listă

numbers.add(1); // Adăugarea numărului 1 în listă

// Iterarea prin elementele listei și afișarea acestora folosind o expresie lambda

numbers.forEach( (n) -> { System.out.println(n); } );

}

}

Cod Java – Clasa LambdaApp

interface Cab {

double bookCab(String source, String destination);

}

public class LambdaApp {

public static void main(String[] args) {

// Crearea unei instanțe a interfeței "Cab" utilizând expresia lambda

Cab cab = (source, destination) -> {

// Afisarea unui mesaj cu sursa și destinația

System.out.println("UberX Booked from " + source + " to " + destination + " Arriving Soon !!");

// Returnarea unui preț fictiv (în acest caz, 200)

return 200;

};

// Apelarea metodei bookCab și atribuirea rezultatului la variabila "pret"

double pret = cab.bookCab("Chisinau", "Balti");

// Afișarea prețului

System.out.println("Pret: " + pret + " lei.");

}

}

Cod Java – Clasa ReferinteApp

// Definirea interfeței funcționale "Calculator" cu metoda "add" care adună doi intei.

interface Calculator {

void add(int num1, int num2);

}

// Clasa "Calc" cu o metodă "letsAdd" care adună și afișează rezultatul.

class Calc {

public void letsAdd(int num1, int num2) {

System.out.println(num1 + " plus " + num2 + " este egal cu " + (num1 + num2));

}

}

// Clasa "ReferinteApp" pentru demo-ul utilizării referințelor către metode.

public class ReferinteApp {

public static void main(String[] args) {

// Crearea unei instanțe a clasei "Calc"

Calc calc = new Calc();

// Crearea unei referințe la metoda "letsAdd" din clasa "Calc"

Calculator cRef = calc::letsAdd;

// Apelarea metodei "add" prin intermediul referinței

cRef.add(12, 23);

}

}

Rezultate Consolă

*Rezultatele obținute în Consolă sunt:*

*Clasa Main:*

5

9

8

1

*Clasa Lambda App:*

UberX Booked from Chisinau to Rezina Arriving Soon !!

Pret: 200.0 lei.

*Clasa ReferinteApp:*

12 plus 23 este egal cu 35

Concluzii

În concluzie, această lucrare de laborator a avut ca scop să ne ofere o înțelegere profundă și să ne ofere experiență practică în utilizarea expresiilor lambda în programarea Java. Prin această lucrare, am explorat și aplicat conceptele fundamentale ale expresiilor lambda, reprezentând o paradigmă modernă și puternică în dezvoltarea de aplicații.

Expresiile lambda ne-au permis să scriem cod mai concis și mai clar, contribuind la eficiența și flexibilitatea programelor noastre. Am învățat să lucrăm cu interfețe funcționale și să definim funcții anonime într-un mod simplificat, fără a sacrifica precizia sau performanța codului.

Referințe bibliografice

1. *Java Lambda Expressions* [online] [vizitat 17.10.2023]. Disponibil: <https://www.w3schools.com/java/java_lambda.asp>
2. *Java 8 Tutorial* [online] [vizitat 17.10.2023]. Disponibil: <https://winterbe.com/posts/2014/03/16/java-8-tutorial/>
3. *Lambda Expressions* [online] [vizitat 18.10.2023]. Disponibil: <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>