**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. 7**

**„Internaționalizarea”**

**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](#_Toc149210512)

[Ce este internaționalizarea? 2](#_Toc149210513)

[Cod Java – Clasa Main 2](#_Toc149210514)

[Concluzii 4](#_Toc149210515)

[Referințe bibliografice 4](#_Toc149210516)

Scop. Condiția problemei

Scopul acestei lucrări de laborator legată de internationalizare în Java este familiarizarea cu conceptele și tehnicile utilizate în gestionarea localizării și a internaționalizării în dezvoltarea de aplicații Java. Acest lucru implică înțelegerea modului în care pot fi create și utilizate resursele pentru diferite limbi și țări, astfel încât aplicația să poată oferi o experiență adecvată utilizatorilor din diverse regiuni geografice.

De asemenea se presupune crearea unei aplicații de bază în Java care să fie capabilă să afișeze text și interfețe în mai multe limbi, în funcție de preferințele utilizatorului. Aceasta va implica gestionarea resurselor pentru fiecare limbă, precum traducerea textelor și formatarea corectă a datelor pentru fiecare localizare.

Ce este internaționalizarea?

Internationalizarea (i18n) reprezintă procesul de adaptare a unei aplicații software sau a unui produs pentru a putea fi utilizat în diferite țări și regiuni din întreaga lume, fără a necesita modificări majore ale codului sursă. Scopul principal al internationalizării este de a face ca o aplicație să fie multilingvă și multiculturală, astfel încât utilizatorii din diverse medii culturale și lingvistice să poată interacționa cu ea în mod eficient.

Cod Java – Clasa Main

import java.text.NumberFormat;

import java.util.\*;

import java.nio.charset.Charset;

import java.text.DateFormat;

public class Main {

public static void main(String[] args) {

// Afiseaza codificare implicită

System.out.println("Codificare implicită:"+System.getProperty("file.encoding"));

Double number = 674.2;

//Formatul numerelor

Locale localizare= new Locale("ro","MD");

NumberFormat formatare=NumberFormat.getNumberInstance(localizare); // Formatul numărului corespunzător localizării

String rezultat= formatare.format(number); // Convertește un număr într-un șir în funcție de formatul numărului

System.out.println(rezultat+" "+localizare.toString()); // Afișează rezultatul

System.out.println("\nLocalizări disponibile:");

// Lista localurilor disponibile

Locale lista[]=Locale.getAvailableLocales(); // Obține toate localitățile disponibile

for (int i=0;i<lista.length;i++){ //Buclă toate localitățile disponibile

System.out.println(lista[i].toString());

}

System.out.println("\nFormatul datei");

//Afiseaza data

Date azi= new Date(); // Primește data de azi

DateFormat dformatare= DateFormat.getDateInstance(DateFormat.DEFAULT,localizare); // Obține formatul de dată corespunzător localizării (franceză).

String data = dformatare.format(azi); //Convertește o dată într-un șir în funcție de formatul de dată al acestei localități

System.out.println(data+"\t"+localizare); // Afișează rezultatul

System.out.println("\nSeturi de caractere disponibile: ");

// Afișează toate codificările disponibile

Iterator<String> $it=

Charset.availableCharsets().keySet().iterator(); // Obținerea unui iterator să itereze prin toate codificările disponibile

while($it.hasNext()){ // Afisarea tuturor codificărilor

System.out.println($it.next());

}

// Localizare standard

Locale loc = Locale.getDefault(); // Obține localitatea implicită (depinde de sistemul în care rulează programul)

System.out.println("\nLocalizare implicită: "+loc); // Nume local (ru\_RU)

System.out.println(loc.getLanguage()); // Codul numelui limbii locale (ru)

System.out.println(loc.getCountry()); // Cod de țară locală (RU)

System.out.println(loc.getDisplayLanguage()); //Numele limbii locale: rusă

System.out.println(loc.getDisplayCountry()); // Nume țara locală: Rusia

System.out.println(loc.getDisplayLanguage(Locale.ENGLISH)); //Numele limbii locale într-o altă limbă locală (engleza): russe

System.out.println(loc.getDisplayCountry(Locale.ENGLISH)); //Numele țării localității în limba altei locații (în engleza): Russie

}

}

Concluzii

În concluzie, această lucrare de laborator a avut ca scop explorarea și înțelegerea conceptelor legate de internationalizare în dezvoltarea de software. Am descoperit importanța adaptării aplicațiilor pentru diverse culturi și limbi, precum și modul în care această abordare contribuie la extinderea utilizării globale a produselor software.

De asemenea, am învățat să lucrăm cu direcția textului și să gestionăm seturile de caractere într-un mod eficient. Toate aceste cunoștințe dobândite în cadrul acestei lucrări ne-au echipat cu abilitățile necesare pentru a crea aplicații care pot fi utilizate cu succes pe scară globală.

Referințe bibliografice

1. *Internationalization and Localization in Java* [online] [vizitat 27.10.2023]. Disponibil: <https://www.javatpoint.com/internationalization-in-java>
2. *Internationalization and Localization in Java* *8* [online] [vizitat 27.10.2023]. Disponibil: <https://www.baeldung.com/java-8-localization>
3. *Java Se 8 Internationalization support* [online] [vizitat 28.10.2023]. Disponibil: <https://docs.oracle.com/javase/8/docs/technotes/guides/intl/index.html>