# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №3**

**«Побудова програм потоків за допомогою стандартних бібліотек Java»**

**з дисципліни «Програмне забеспечення мережевих технологій»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

ст. викл.Сенів М.М.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

ЛЬВІВ – 2014

**Тема роботи**: Побудова програм потоків за допомогою стандартних бібліотек Java.

**Мета роботи:** Освоїти елементи програмування мовою JAVA, а також засвоїти класи і методи графічної бібліотеки awt (swing).

## 1. Теоретичні відомості

Один потік - це одна одиниця виконання коду. Кожен потік послідовно виконує інструкції процесу , якому він належить , паралельно з іншими потоками цього процесу.

Слід окремо обговорити фразу «паралельно з іншими потоками». Відомо, що на одне ядро процесора, в кожен момент часу, доводиться одна одиниця виконання. Тобто одноядерний процесор може обробляти команди тільки послідовно, по одній за раз (у спрощеному випадку). Однак запуск декількох паралельних потоків можливий і в системах з одноядерними процесорами. У цьому випадку система буде періодично перемикатися між потоками, по черзі даючи виконуватися то одному, то іншому потоку. Така схема називається псевдо-паралелізмом. Система запам'ятовує стан (контекст) кожного потоку, перед тим як переключитися на інший потік, і відновлює його з повернення до виконання потоку. У контекст потоку входять такі параметри, як стек, набір значень регістрів процесора, адреса виконуваної команди та інше ...

Простіше кажучи, при псевдопаралельному виконанні потоків процесор перемикається між виконанням декількох потоків, виконуючи по черзі частину кожного з них.

Кожен процес має хоча б один виконується потік. Той потік , з якого починається виконання програми, називається головним. У мові Java, після створення процесу, виконання головного потоку починається з методу main(). Потім, у міру необхідності , в заданих програмістом місцях, і при виконанні заданих ним же умов, запускаються інші, побічні потоки.

У мові Java потік представляється у вигляді об'єкта - нащадка класу Thread. Цей клас інкапсулює стандартні механізми роботи з потоком.

Запустити новий потік можна двома способами:

**Спосіб 1.**

Створити об'єкт класу Thread, передавши йому в конструкторі щось, що реалізує інтерфейс Runnable. Цей інтерфейс містить метод run(), який буде виконуватися в новому потоці. Потік закінчить виконання, коли завершиться його метод run().

class SomeThing / / Щось, що реалізує інтерфейс Runnable

implements Runnable / / (що містить метод run ())

{

     public void run () / / Цей метод буде виконуватися в побічному потоці

     {

         System.out.println ("Привіт з побаченого потоку!");

     }

}

public class Program / ​​/ Клас з методом main ()

{

     static SomeThing mThing; / / mThing - об'єкт класу, що реалізовує інтерфейс Runnable

     public static void main (String [] args)

     {

         mThing = new SomeThing ();

         Thread myThready = new Thread (mThing); / / Створення потоку "myThready"

         myThready.start (); / / Запуск потоку

         System.out.println ("Головний потік завершено ...");

     }

}

**Спосіб 2.**

Створити нащадка класу Thread і перевизначити його метод run():

class AffableThread extends Thread

{

     @ Override

     public void run () / / Цей метод буде виконаний в побічному потоці

     {

         System.out.println ("Привіт з побічного потоку!");

     }

}

public class Program

{

     static AffableThread mSecondThread;

     public static void main (String [] args)

     {

         mSecondThread = new AffableThread (); / / Створення потоку

         mSecondThread.start (); / / Запуск потоку

         System.out.println ("Головний потік завершено ...");

     }

}

## 2. Хід роботи

**Варіант 5.** Створити клас потоку Movie, який в методі run містить код малювання ліній, фігур, зображень з файлу тощо. Вхідними параметрами потоку встановити координати центру, час затримки або інші атрибути малюнків. Запустити декілька потоків щоб їх робота відображала рух об’кта по екрану.

В середовищі netbeans створимо новий проект наберемо текст JAVA-аплета:

import java.applet.Applet;

import java.awt.\*;

import java.util.Calendar;

import java.util.Random;

//нарямки руху фігур

enum direction {HORIZONTAL, VERTICAL, DIAGONAL};

public class MovieApplet extends Applet {

//кількість потоків

private int countThreads = 0;

private Random rand;

@Override

public void init() {

rand = new Random(Calendar.getInstance().getTimeInMillis());

setBackground(Color.YELLOW);

//створюємо два потоки

new Movie(50, 20, 0xff5555, direction.DIAGONAL, 50).start();

new Movie(30, 70, 0x5555ff, direction.VERTICAL, 100).start();

}

@Override

public boolean mouseDown(Event evt, int x, int y) {

//при натиску клавіші миші створюємо новий потік

new Movie(x, y,

rand.nextInt(0xffffff),

direction.values()[rand.nextInt(direction.values().length)] ,

50 + countThreads \* 10).start();

return true;

}

class Movie extends Thread {

//останні координати фігури

private int lastX,lastY;

//поточні координати фігури

private int x,y;

//час затримки фігури на екрані

private int pause;

//напрямок руху по екрану

private direction dir;

//колір

private Color col;

Movie(int xStart, int yStart, int col, direction dir, int pause) {

countThreads++;

//присвоюємо початкові значення

this.col = new Color(col);

this.pause = pause;

this.dir = dir;

x = xStart; y = yStart;

lastX = x; lastY = y;

}

@Override

public void run() {

while(true) {

//запам'ятовуємо старі координати

lastX = x; lastY = y;

//в залежності від напряму змінюємо поточні

//координати

switch(dir) {

case DIAGONAL: x += 10; y += 10; break;

case VERTICAL: y += 10; break;

default: x += 10;

}

//якщо вийшли за видиму область

if(x > 1000) x = 20;

if(y > 500) y = 20;

//малюємо

mypaint();

try {

//затримка зображення

Movie.sleep(pause);

}

catch (InterruptedException e){}

}

}

public void mypaint() {

Graphics g = getGraphics();

if(g == null) return;

//замальовуємо "старе" зображення кольором фону

g.setColor(Color.YELLOW);

g.drawOval(lastX, lastY, 20, 20);

//малюємо "нове" зображення

g.setColor(col);

g.drawOval(x, y, 20, 20);

}

}

}

Відредагуємо файл MovieApplet.html, який було автоматично створено середовищем netbeans до наступного вигляду:

<HTML>

<HEAD>

<TITLE>Лабораторна робота №3</TITLE>

</HEAD>

<BODY>

<H3><HR WIDTH="100%">Побудова програм потоків за допомогою стандартних бібліотек Java<HR WIDTH="100%"></H3>

<P>

<APPLET codebase="classes" code="org/me/movie/MovieApplet.class" width=1000 height=500></APPLET>

</P>

<HR WIDTH="100%"><FONT SIZE=-1><I>Автор: Тарас Гринчук</I></FONT>

</BODY>

</HTML>

Запустимо даний документ у браузері (рис. 2.1):



Рис. 2.1. Вигляд html-документа у браузері

Як бачимо у вікні атпета, рухаються дві фігури, одна – по вертикалі, а друга – по діагоналі. Зробимо декілька кліків у формі вікна (рис. 2.2):



Рис. 2.2

Як бачимо, з кожним кліком з’яввляється нова фігура у місці це знаходився курсор миші. Фігури рухаються по вертикалі, горизонталі та діагоналі, кожна наступна рухається повільніше.

## ВИСНОВКИ

На даній лабораторній роботі я вивчив основні принципи побудови багатопоточних JAVA-програм, освоїв елементи програмування мовою JAVA при розробленні аплетів та потоків, а також засвоїв класи і методи бібліотек java.awt, java.applet.