МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Практична робота №5**

з дисципліни «Розробка крос-платформного програмного забезпечення»

тема: «Асинхронне виконання дій. Малювання на полотні. Візуалізація медичних даних»

**Виконав:**

Студент гр. БС-03

Затуловський Г. А.

**Перевірела:**

ст.викл. Сердаковський В. С.

ac. Рисін. С. В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

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

(підпис викладача)

Київ-2023

**Мета:** уяснити принципи дійсної та псевдопаралельності процесів та можливості організації асинхронної організації дій; усвідомити принципи відображення графічної інформації надвовимірному просторі; ознайомитися з представленням зображень у растровому та векторному вигляді; навчитися використовувати засоби для малювання на полотні та виконання асинхронних процесів, розібратися з використанням класів BufferedImage, Canvas, Timer та TimerTask; получити уявлення про стандарт DICOM та інформацію, з якою він дозволяє працювати; організувати показ відеоряду зображень з набору DICOM шляхом періодичного відображення послідовних кадрів.

## **Завдання:**

1. Отримати з віддаленного сервера перелік файлів DICOM у наборі згідно з варіантом.

2. Завантажити з файлів наданого набору зображення з використанням класу BufferedImage та їх метадані.

2.1. Оскільки процес виборки файлів з зображеннями потребує деякий час, організувати завантаження таким чином, щоб при читанні набору відображалась інформація про файл, що завантажується в даний момент.

2.2. Для уникнення колізій на час, доки відбувається завантаження набору даних, інтерфейс користувача має бути заблоковано.

3. Відсортувати зображення по положенню зрізів, на яких вони отримані, та послідовності їх отримання.

4. Відобразити відсортований список всіх зрізів набору з вказанням положення кожного зрізу та кількості зображень на ньому.

5. При виборі довільного зрізу зі списку забезпечити показ зображень з цього зрізу з використанням класу Canvas.

6. Організувати показ відеоряду шляхом послідовноі зміни зображень

## **Лістинг програми:**

**Controller.java:**

import javafx.application.Platform;  
import javafx.embed.swing.SwingFXUtils;  
import javafx.fxml.FXML;  
import javafx.scene.canvas.Canvas;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.ListView;  
import javafx.scene.text.Text;  
import javafx.scene.image.Image;  
  
import java.io.IOException;  
import java.util.\*;  
  
public class Controller {  
  
 @FXML  
 public Canvas CANVAS\_WIDGET;  
 @FXML  
 public Button DOWNLOAD\_button;  
 @FXML  
 public Button LOOP\_button;  
 @FXML  
 public Label IMAGE\_VIEW;  
 @FXML  
 public Text SHOW\_IMAGE\_INFO;  
  
 @FXML  
 public ListView<String> slices\_COLLECT\_View;  
 private FTP\_Connection ftpConnection;  
 private DICOMImage\_collection dicom\_Image\_Collect;  
 private List<String> FILENAMES;  
 private Map<Double, Integer> slices;  
  
 private int CURRENT\_Image\_ID = 0;  
 private List<DICOMImage> CURRENT\_Images = new ArrayList<>();  
  
 String local\_dir;  
 String remote\_dir;  
  
 boolean isLoop = false;  
  
 TimerTask timerTask;  
  
 @FXML  
 public void initialize(String url, int port, String user, String pass, String localDir, String remoteDir)  
 throws IOException {  
 LOOP\_button.setDisable(true);  
 ftpConnection = new FTP\_Connection(url, port, user, pass);  
 dicom\_Image\_Collect = new DICOMImage\_collection();  
 this.local\_dir = localDir;  
 this.remote\_dir = remoteDir;  
  
 }  
  
 @FXML  
 public void Download\_DICOM() {  
 DOWNLOAD\_button.setDisable(true);  
 try {  
 FILENAMES = ftpConnection.GET\_file\_names(remote\_dir);  
 var thread = new Thread(new Runnable() {  
 String FILENAME;  
 int downloadingIndex;  
  
 @Override  
 public void run() {  
 Runnable textUpdater = () -> SHOW\_IMAGE\_INFO.setText(  
 "Завантаження: " + (downloadingIndex + 1) + "/" + FILENAMES.size() + " " + FILENAME );  
 for (int i = 0; i < FILENAMES.size(); i++) {  
 FILENAME = FILENAMES.get(i);  
 downloadingIndex = i;  
 try {  
 ftpConnection.DOWNLOAD\_file(FILENAME , local\_dir, remote\_dir);  
 dicom\_Image\_Collect.ADD\_Image(local\_dir, FILENAME );  
 Platform.*runLater*(textUpdater);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 dicom\_Image\_Collect.Sort\_collections();  
 slices = dicom\_Image\_Collect.GET\_Slices();  
 for (Double sliceLocation : slices.keySet()) {  
 Platform.*runLater*(() -> slices\_COLLECT\_View.getItems()  
 .add("Місцезнаходження: " + sliceLocation +", Розмір: " + slices.get(sliceLocation)));  
 }  
 SHOW\_IMAGE\_INFO.setText("Завантаження файлів завершено!");  
 }  
 });  
 thread.setDaemon(true);  
 thread.start();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 public void Looping\_Slice() {  
 if (isLoop) {  
 timerTask.cancel();  
 LOOP\_button.setText("Розпочати цикл");  
 } else {  
 timerTask = new TimerTask() {  
 @Override  
 public void run() {  
 Platform.*runLater*(() -> Show\_Current\_Image());  
 }  
 };  
 var timer = new Timer(true);  
 timer.scheduleAtFixedRate(timerTask, 0, 50);  
 LOOP\_button.setText("Зупинити цикл");  
 }  
 isLoop = !isLoop;  
 }  
  
 private void Show\_Current\_Image() {  
 DICOMImage dicomImage = CURRENT\_Images.get(CURRENT\_Image\_ID);  
 IMAGE\_VIEW.setText("Назва: " + dicomImage.FileName + "\nНомер серії: "  
 + dicomImage.SeriesNumber + "\nНомер знаття: "  
 + dicomImage.AcquNumber + "\nНомер екземпляру: " + dicomImage.InstNumber);  
  
 Image image = SwingFXUtils.*toFXImage*(dicomImage.Image, null);  
 var ctx = CANVAS\_WIDGET.getGraphicsContext2D();  
 CANVAS\_WIDGET.setWidth(image.getWidth());  
 CANVAS\_WIDGET.setHeight(image.getHeight());  
 ctx.drawImage(image, 0, 0);  
  
 CURRENT\_Image\_ID = CURRENT\_Image\_ID < CURRENT\_Images.size() - 1 ? ++CURRENT\_Image\_ID : 0;  
 }  
  
 public void Object\_Select() {  
 Integer index = slices\_COLLECT\_View.getSelectionModel().getSelectedIndex();  
 CURRENT\_Images = dicom\_Image\_Collect.GET\_Images\_Slide((double) slices.keySet().toArray()[index]);  
 CURRENT\_Image\_ID = 0;  
 Show\_Current\_Image();  
 if (LOOP\_button.isDisable())  
 LOOP\_button.setDisable(false);  
  
 }  
}

**DICOMImage.java:**

import java.awt.image.BufferedImage;  
import java.io.FileNotFoundException;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.\*;  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
public class DICOMImage implements Comparable <DICOMImage>  
{  
 String FileName;  
 int Wd;  
 int Ht;  
 double SliceLocation;  
 int SeriesNumber;  
 int AcquNumber;  
 int InstNumber;  
 double AcquTime;  
 double ContTime;  
 double TrigTime;  
 BufferedImage Image;  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 static private long skip ( InputStream IS, int Size ) throws IOException {  
 long skipped = 0;  
 for ( ; Size > 0; ){  
 long \_skipped = IS.skip(Size);  
 if ( \_skipped <= 0 )  
 break;  
 skipped += \_skipped;  
 Size -= \_skipped;  
 }  
  
 return skipped;  
 }  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 static private int read (InputStream IS, byte Buff [ ], int Size)throws IOException{  
 byte \_buff [ ] = new byte [ Size ];  
 int read = 0;  
 for ( ; Size > 0; ){  
 int \_read = IS.read(\_buff, 0, Size);  
 if ( \_read < 0 )  
 break;  
 for ( int byteNo = 0; byteNo < \_read;)  
 Buff[read++] = \_buff[byteNo++];  
  
 Size -= \_read;  
 }  
  
 return read;  
 }  
  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 public DICOMImage ( String PathName, String FileName )throws FileNotFoundException, IOException, Exception{  
 File initialFile = new File(PathName + "/" + FileName);  
 InputStream fis = new FileInputStream(initialFile);  
  
 // InputStream fis = (new URL("file:///" + PathName + "/" + FileName)).openStream();  
  
 // skipping 'File Preamble'  
 *skip*(fis, 0x80);  
  
 // site for reading  
 byte buf4 [ ] = new byte [ 4 ];  
  
 // checking 'DICOM Prefix'  
 int read = *read*(fis, buf4, 4);  
 if (! new String(buf4).equals("DICM")){  
 fis.close();  
 throw new Exception("Файлів DICOM не знайдено");  
 }  
  
 //looks like, going on  
 this.FileName = FileName;  
 SeriesNumber = AcquNumber = InstNumber = 0;  
 SliceLocation = AcquTime = ContTime = TrigTime = 0;  
 Image = null;  
  
 // DICOM tag  
 short tag2 [ ] = new short [ 2 ];  
  
 // initial datatype assuming explicit  
 boolean expl = true;  
  
 // c'mon iterating metadata  
 for ( ; ; ){  
 // reading tag bytes  
 if ( (read = *read*(fis, buf4, 4)) <= 0 )  
 break;  
  
 // getting tag  
 tag2[0] = (short) (((buf4[1] & 0xFF) << 8) | (buf4[0] & 0xFF));  
 tag2[1] = (short) (((buf4[3] & 0xFF) << 8) | (buf4[2] & 0xFF));  
  
 // site for datatype  
 String dt = "";  
 // site for length  
 int len4 = 0;  
 // site for text  
 String text = "";  
  
 if ( expl || tag2[0] == 0x0002 ){  
 // explicit datatype  
 // reading datatype bytes  
 read = *read*(fis, buf4, 2);  
 // getting datatype  
 dt = new String(buf4, 0, 2);  
  
 // getting length  
 if (dt.equals("OB") ||   
 dt.equals("OW") ||   
 dt.equals("OF") ||   
 dt.equals("SQ") ||   
 dt.equals("UN")){  
 // 'Other Byte' || 'Other Word' || 'Other Float' || 'Sequence of Items' || 'Unknown'  
 *skip*(fis, 2);  
 read = *read*(fis, buf4, 4);  
 len4 = (int) (((buf4[3] & 0xFF) << 24) | ((buf4[2] & 0xFF) << 16) | ((buf4[1] & 0xFF) << 8) | (buf4[0] & 0xFF));  
 }  
 else{  
 read = *read*(fis, buf4, 2);  
 len4 = (int) (((buf4[1] & 0xFF) << 8) | (buf4[0] & 0xFF));  
 }  
 } else {  
 // implicit datatype  
 dt = " ";  
 // getting length  
 read = *read*(fis, buf4, 4);  
 len4 = (int) (((buf4[3] & 0xFF) << 24) |   
 ((buf4[2] & 0xFF) << 16) |   
 ((buf4[1] & 0xFF) << 8) |   
 (buf4[0] & 0xFF));  
 }  
  
 if (tag2[1] == 0x0000){  
 // group  
 // getting length  
 if (tag2[0] == 0x0002 ){  
 read = *read*(fis, buf4, 4);  
 len4 = (int) (((buf4[3] & 0xFF) << 24) |   
 ((buf4[2] & 0xFF) << 16) |   
 ((buf4[1] & 0xFF) << 8) |   
 (buf4[0] & 0xFF));  
 }  
 else *skip*(fis, len4);  
 } else if ( tag2[0] == 0x0002 && tag2[1] == 0x0001){  
 // version  
 short vers;  
 read = *read*(fis, buf4, 2);  
 vers = (short) (((buf4[1] & 0xFF) << 8) | (buf4[0] & 0xFF));  
 } else if ( tag2[0] == 0x0008 && (tag2[1] == 0x1110 ||   
 tag2[1] == 0x1111 || tag2[1] == 0x1115 || tag2[1] == 0x1120)){  
 // 'Referenced Sequences'  
 byte buff [ ] = new byte [ 12 ];  
 read = *read*(fis, buff, 8);  
 } else if ( tag2[0] == 0x0028 && (tag2[1] == 0x0010 || tag2[1] == 0x0011)){  
 // width || height  
 byte ch [ ] = new byte [ len4 ];  
 read = *read*(fis, ch, len4);  
 if ( tag2[1] == 0x0010 )  
 Ht = (int) (((ch[1] & 0xFF) << 8) | (ch[0] & 0xFF));  
 else  
 Wd = (int) (((ch[1] & 0xFF) << 8) | (ch[0] & 0xFF));  
  
 } else if ( tag2[0] == 0x7FE0 && tag2[1] == 0x0010 ){  
 // pixel data  
 Image = new BufferedImage(Wd, Ht, BufferedImage.*TYPE\_3BYTE\_BGR*);  
 int bytes = Wd \* Ht \* 2;  
 // reading 'pixels'  
 byte data [ ] = new byte [ bytes ];  
 read = *read*(fis, data, bytes);  
 // + scaling 'pixels' to '0-255' range  
 int min = 65535, max = 0;  
 for (int pxlNo = 0; pxlNo < bytes; pxlNo += 2){  
 int val = (((int)data[pxlNo + 1] & 0xFF) << 8) | ((int)data[pxlNo] & 0xFF);  
 if ( val > max ) max = val;  
 if ( val < min ) min = val;  
 }  
  
 for (int pxlNo = 0, yy = 0; yy < Ht; yy++)  
 for (int xx = 0; xx < Wd; xx++, pxlNo += 2){  
 int val = (max != 0 ? ( (((int)data[pxlNo + 1] & 0xFF) << 8) | ((int)data[pxlNo] & 0xFF) ) \* 255 / max : 0);  
 Image.setRGB(xx, yy, (val << 16) | (val << 8) | val);  
 }  
 // - scaling 'pixels' to '0-255' range  
 break;  
 } else if ( dt.equals("SQ") ) {  
 byte data [ ] = new byte [ 8 ];  
 read = *read*(fis, data, 8);  
 } else if (dt.equals("OB") ||   
 dt.equals("FL") ||   
 dt.equals("FD") ||   
 dt.equals("SS") ||   
 dt.equals("US") ||   
 len4 > 0 ){  
 // 'Other Byte' || 'Floating Point Single' || 'Floating Point Double' || 'Signed Short' || 'Unsigned Short'  
 byte data [ ] = new byte [ len4 ];  
 read = *read*(fis, data, len4);  
 if ( len4 > 0 )  
 text = new String(data, 0, len4).trim();  
 }  
  
 // getting needed data  
 if ( (tag2[0] == 0x0008 && (tag2[1] == 0x0032 || tag2[1] == 0x0033)) || (tag2[0] == 0x0018 && tag2[1] == 0x1060)){  
 // 'Acquisition Time' || 'Content Time' || 'Trigger Time'  
 double time = Double.*parseDouble*(text);  
 if ( tag2[0] == 0x0018 )  
 TrigTime = time;  
 else if (tag2[1] == 0x0032)  
 AcquTime = time;  
 else  
 ContTime = time;  
 } else if (tag2[0] == 0x0020 && (tag2[1] == 0x0011 ||   
 tag2[1] == 0x0012 || tag2[1] == 0x0013 || tag2[1] == 0x1041)){  
 // 'Series Number' || 'Acquisition Number' || 'Instance Number' || 'Slice Location'  
 switch ( tag2[1] )  
 {  
 case 0x0011 : SeriesNumber = Integer.*parseInt*(text); break;  
 case 0x0012 : AcquNumber = Integer.*parseInt*(text); break;  
 case 0x0013 : InstNumber = Integer.*parseInt*(text); break;  
 case 0x1041 : SliceLocation = Double.*parseDouble*(text); break;  
 }  
 }  
  
 if ( tag2[0] == 0x0002 && tag2[1] == 0x0010 && ! text.equals("1.2.840.10008.1.2.1") )  
 // implicit datatypes format ('Explicit VR Little Endian Transfer Syntax UID')  
 expl = false;  
 }  
 fis.close();  
 }   
  
 @Override  
 public int compareTo(DICOMImage i) {  
 return Comparator.*comparing*((DICOMImage i2) ->  
 i2.SliceLocation, Comparator.*reverseOrder*())  
 .thenComparingInt((DICOMImage i2) ->  
 i2.SeriesNumber)  
 .thenComparingInt((DICOMImage i2) ->  
 i2.AcquNumber)  
 .thenComparingInt((DICOMImage i2) ->  
 i2.InstNumber)  
 .compare(this, i);  
  
 }  
  
}

**DICOMImage\_collection.java:**

import java.util.\*;  
import java.util.stream.Collectors;  
  
public class DICOMImage\_collection {  
 public ArrayList<DICOMImage> images = new ArrayList<>();  
  
 private Map<Double, Integer> slices = new HashMap<>();;  
  
 public void ADD\_Image(String dir, String fileName) {  
 try {  
 DICOMImage dicomImage = new DICOMImage(dir, fileName);  
 images.add(dicomImage);  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
 }  
   
 public void Sort\_collections() {  
 Collections.*sort*(images);  
 }  
  
 public Map<Double, Integer> GET\_Slices() {  
 if (slices.size() == 0 && images.size() > 0) {  
 slices = new HashMap<>();  
 images.stream().collect(Collectors.*groupingBy*(image -> image.SliceLocation)).forEach((k, v) -> slices.put(k, v.size()));  
 }  
 return slices;  
 }  
  
 public List<DICOMImage> GET\_Images\_Slide(double sliceLocation) {  
 return images.stream().filter(i -> i.SliceLocation == sliceLocation).collect(Collectors.*toList*());  
 }  
}

**FTP\_Connection.java:**

import org.apache.commons.net.ftp.\*;  
import org.apache.commons.net.ftp.FTPClient;  
import java.io.IOException;  
import java.util.\*;  
import java.util.stream.Collectors;  
import java.io.File;  
import java.io.FileOutputStream;  
  
public class FTP\_Connection {  
 private final FTPClient FTP\_Client;  
 private FTPFile[] FTP\_Files;  
  
 public FTP\_Connection(String url, int port, String username, String password) throws IOException {  
 FTP\_Client = new FTPClient();  
 FTP\_Client.connect(url, port);  
 FTP\_Client.login(username, password);  
 FTP\_Client.enterLocalPassiveMode();  
 }  
 public List<String> GET\_file\_names(String dir) throws IOException {  
 this.SET\_ftp\_files(dir);  
 return Arrays.*stream*(FTP\_Files).map(FTPFile::getName).collect(Collectors.*toList*());  
 }  
  
 public void SET\_ftp\_files(String dir) throws IOException {  
 this.FTP\_Files = FTP\_Client.listFiles(dir);  
 }  
  
 public void DOWNLOAD\_file(String fileName, String localDir, String remoteDir) throws IOException  
 {  
 for (FTPFile ftpFile: FTP\_Files) {  
 if (ftpFile.getName().equals(fileName)) {  
 new File(localDir).mkdirs();  
 File file = new File(localDir + "/" + fileName);  
 file.createNewFile();  
 FileOutputStream fos = new FileOutputStream(localDir + "/" + fileName, false);  
 FTP\_Client.retrieveFile(remoteDir+ "/" +fileName, fos);  
 return;  
 }  
 }  
  
 }  
}

**Main.java:**

import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
 private static String *url*;  
 private static int *port*;  
 private static String *username*;  
 private static String *password*;  
 private static String *local\_dir*;  
 private static String *remote\_dir*;  
  
 public static void main(String[] args) {  
 *url* = "195.3.158.22";  
 *port* = 8000;  
 *username* = "03-06";  
 *password* = "P1Fqjp";  
 *local\_dir* = "images";  
 *remote\_dir* = "lab5";  
  
 *launch*(args);  
 }  
  
 @Override  
 public void start(Stage stage) throws Exception {  
 FXMLLoader loaderMain = new FXMLLoader(getClass().getClassLoader().getResource("Main\_window.fxml"));  
 Parent rootMain = loaderMain.load();  
 Scene sceneMain = new Scene(rootMain, 750, 420);  
 Controller controller = loaderMain.getController();  
 controller.initialize(*url*, *port*, *username*, *password*, *local\_dir*, *remote\_dir*);  
 stage.setTitle("5 ЛАБОРОТОРНА РОБОТА - Затуловський");  
  
 stage.setScene(sceneMain);  
  
 stage.show();  
 }  
  
}

**Main\_window.fxml:**

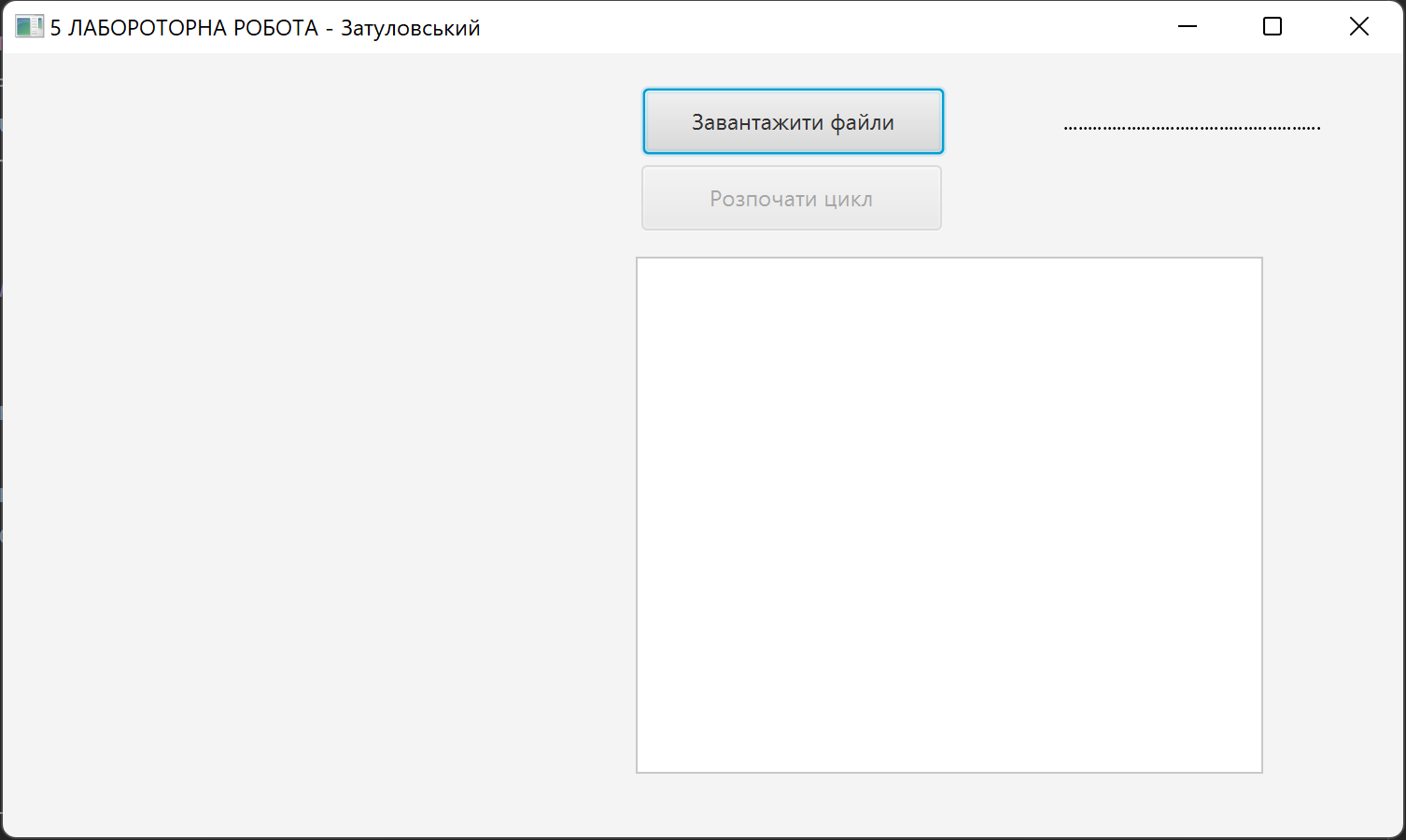
<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.canvas.Canvas?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.ListView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Text?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="455.0" prefWidth="756.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="Controller">  
 <children>  
 <Button fx:id="DOWNLOAD\_button" layoutX="343.0" layoutY="19.0" mnemonicParsing="false" onAction="#Download\_DICOM" prefHeight="35.0" prefWidth="161.0" styleClass="outline-button" text="Завантажити файли" />  
  
 <Button fx:id="LOOP\_button" layoutX="342.0" layoutY="60.0" mnemonicParsing="false" onAction="#Looping\_Slice" prefHeight="35.0" prefWidth="161.0" styleClass="outline-button" text="Розпочати цикл" />  
  
 <Text fx:id="SHOW\_IMAGE\_INFO" layoutX="568.0" layoutY="41.0" text="....................................................." wrappingWidth="145.111328125" />  
  
 <VBox alignment="CENTER" layoutX="24.0" layoutY="19.0" prefHeight="411.0" prefWidth="298.0">  
 <Canvas fx:id="CANVAS\_WIDGET" height="258.0" width="278.0" />  
   
 <Label fx:id="IMAGE\_VIEW" alignment="BOTTOM\_CENTER" contentDisplay="CENTER" wrapText="true">  
 </Label>  
 </VBox>  
 <ListView fx:id="slices\_COLLECT\_View" layoutX="339.0" layoutY="109.0" onMouseClicked="#Object\_Select" prefHeight="277.0" prefWidth="336.0" />  
 </children>  
</AnchorPane>

**DICOMImage::compareTo():**

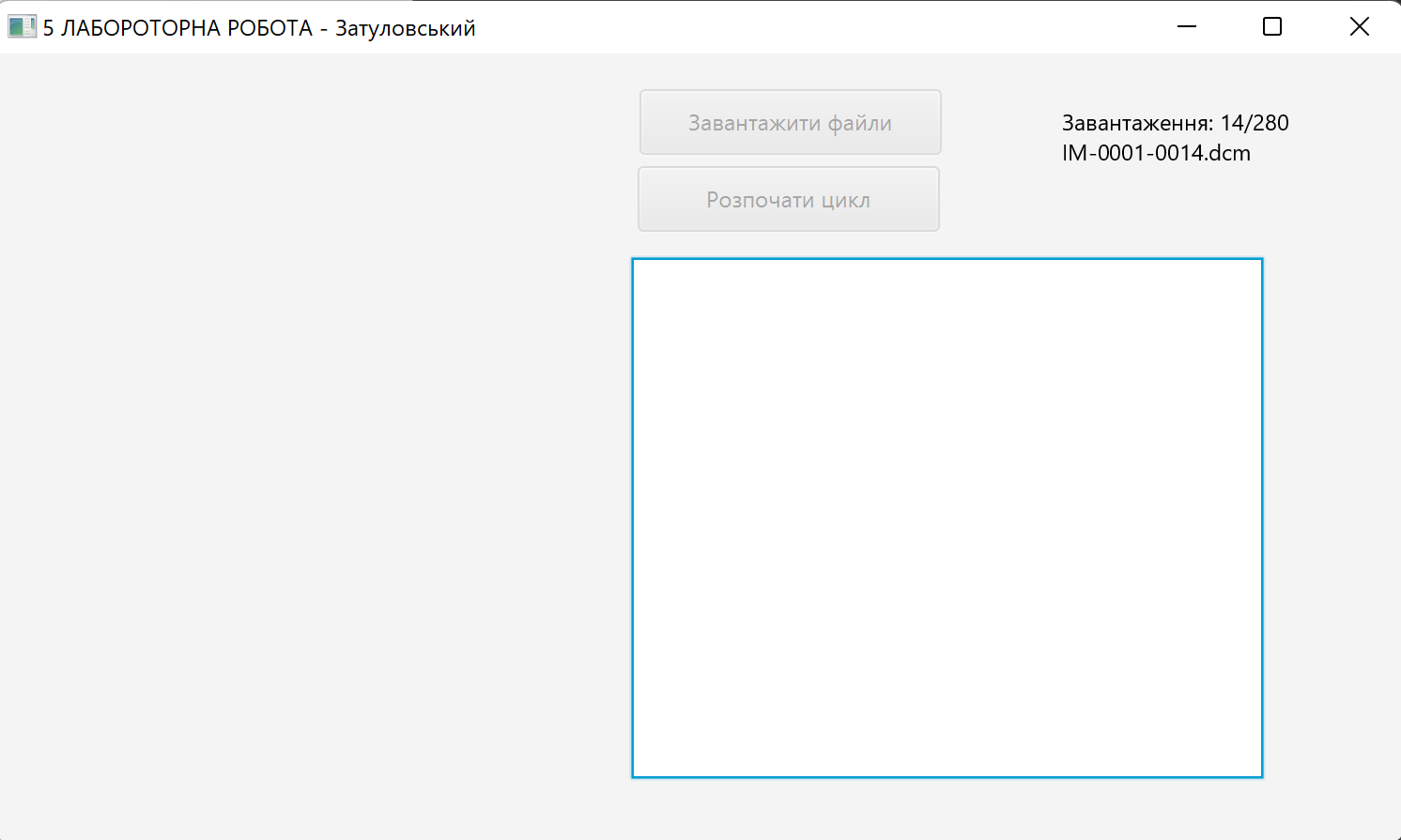
@Override  
public int compareTo(DICOMImage i) {  
 return Comparator.*comparing*((DICOMImage i2) ->  
 i2.SliceLocation, Comparator.*reverseOrder*())  
 .thenComparingInt((DICOMImage i2) ->  
 i2.SeriesNumber)  
 .thenComparingInt((DICOMImage i2) ->  
 i2.AcquNumber)  
 .thenComparingInt((DICOMImage i2) ->  
 i2.InstNumber)  
 .compare(this, i);  
  
}

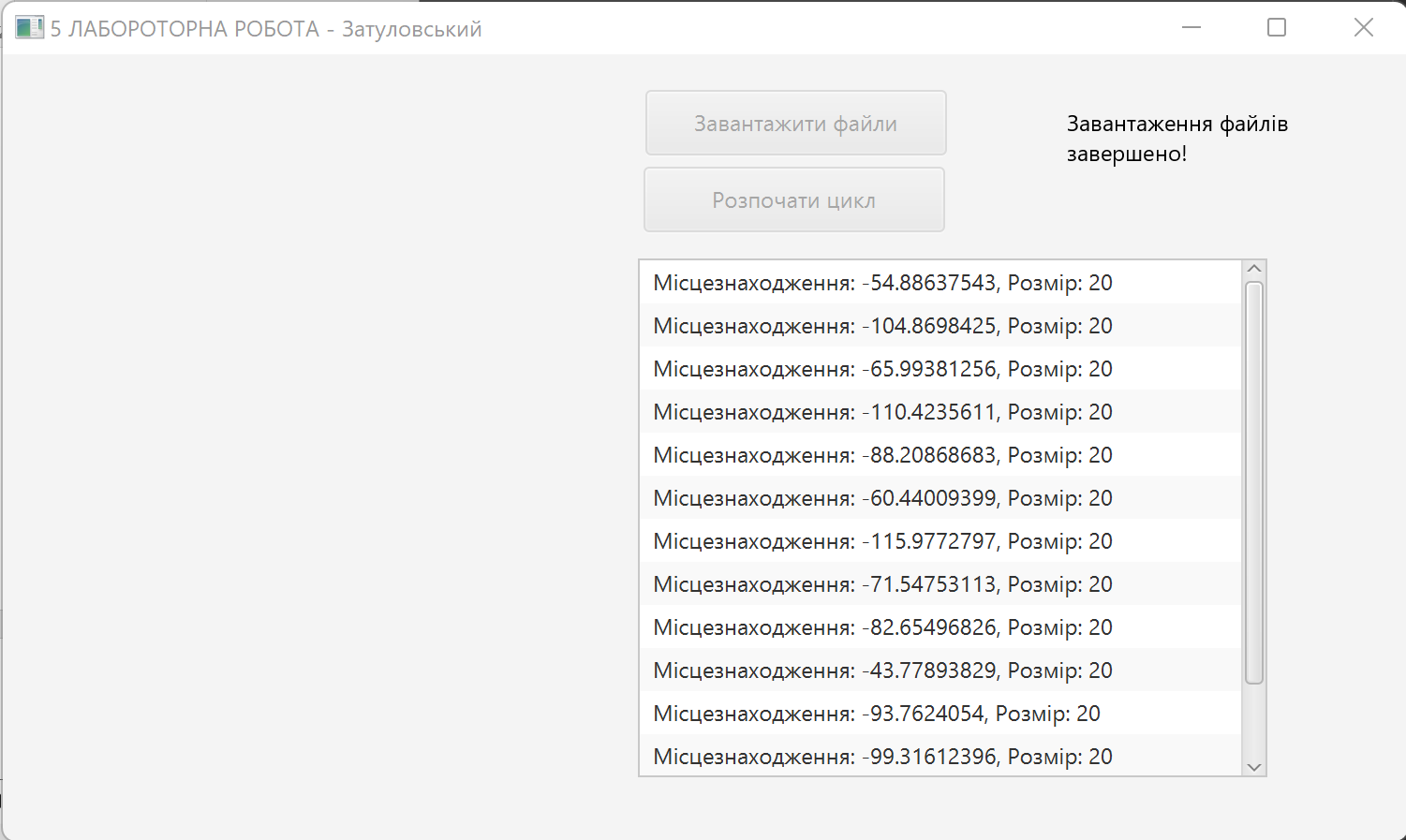
**Результати:**

Запуск програми:

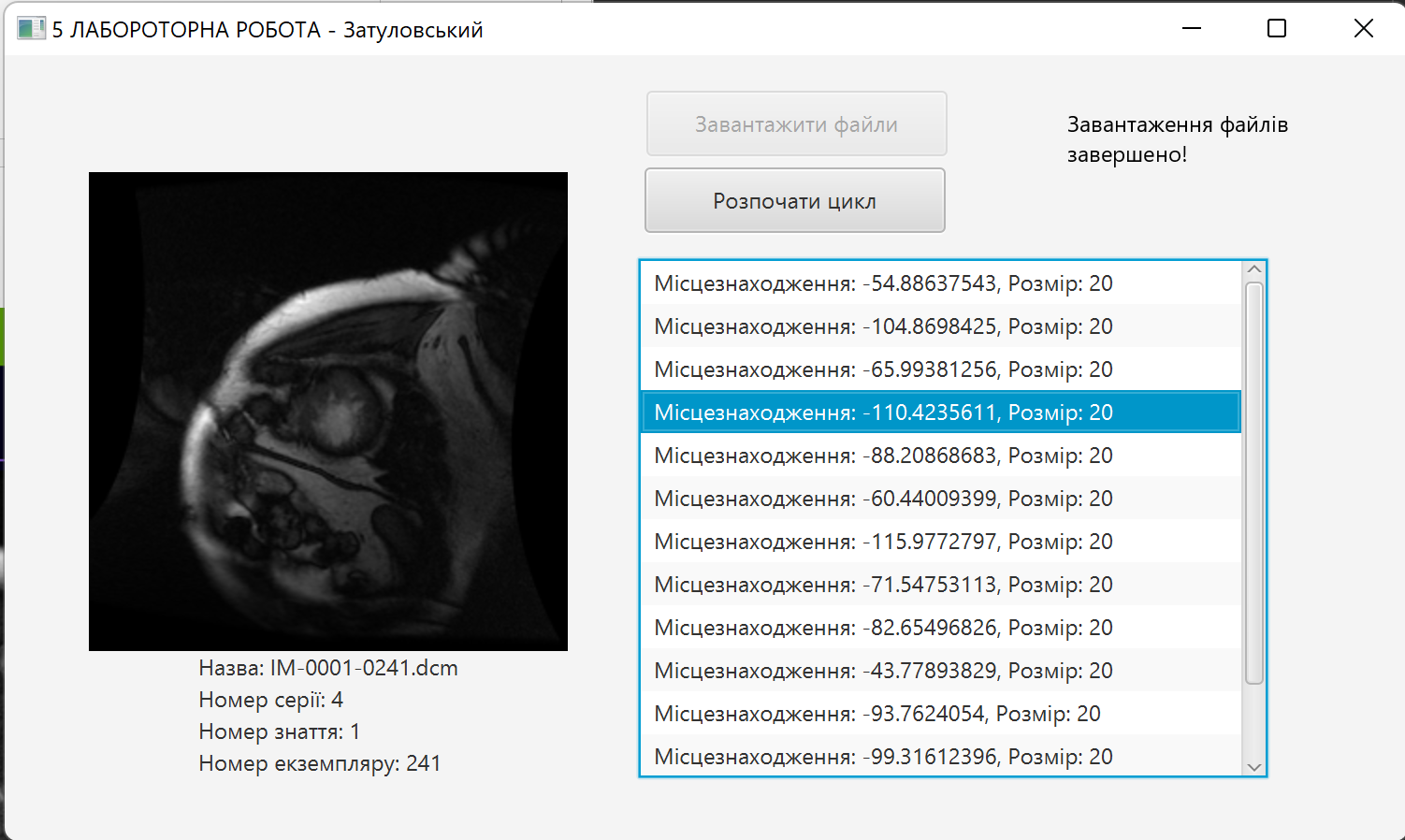


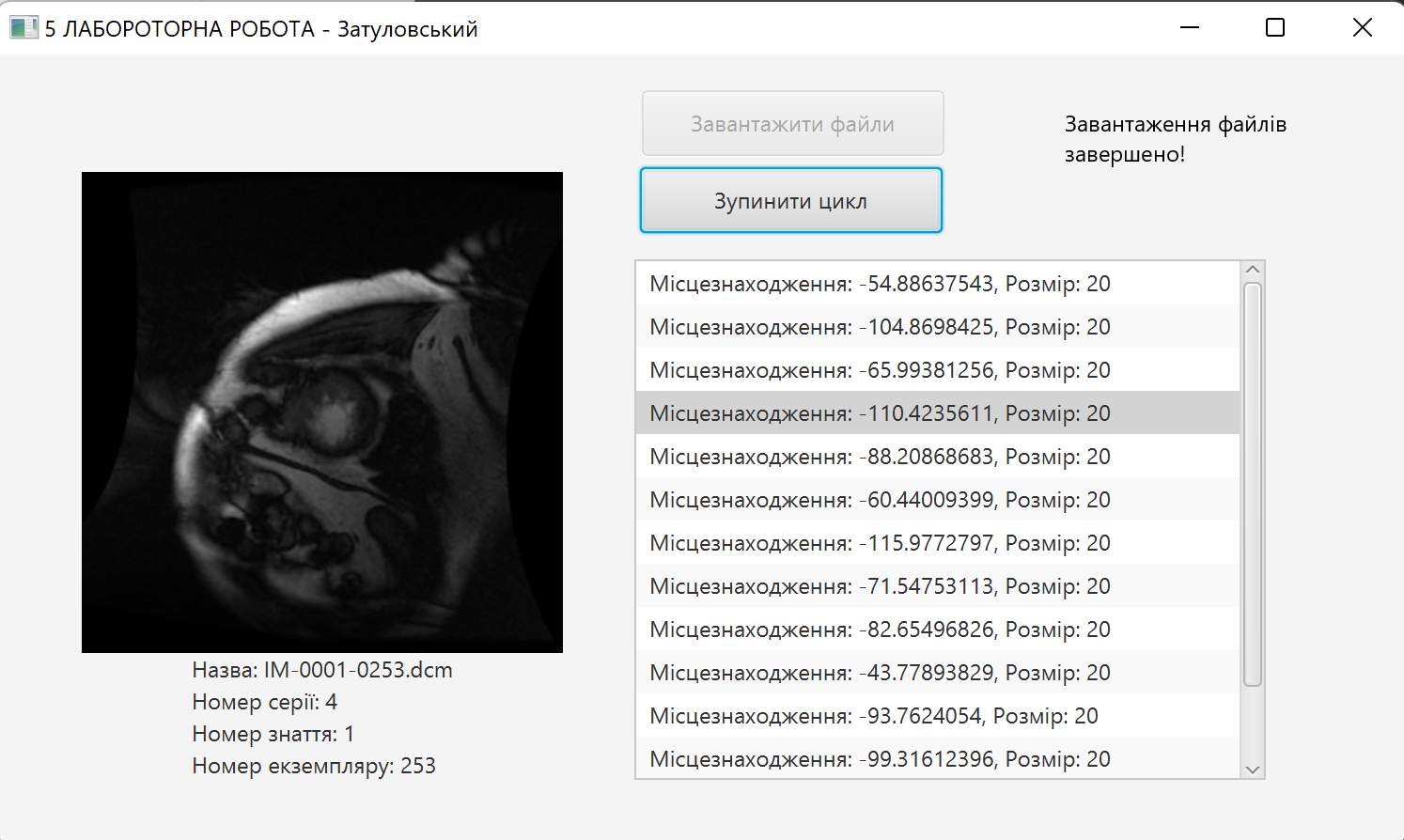
Завантаження DICOM файлів:



Завершення завантаження DICOM файлів:

Цикл DICOM:





**Висновок:** Ми уяснили принципи дійсної та псевдопаралельності процесів та можливості організації асинхронної організації дій; усвідомили принципи відображення графічної інформації надвовимірному просторі; ознайомилися з представленням зображень у растровому та векторному вигляді; навчилися використовувати засоби для малювання на полотні та виконання асинхронних процесів, розібратися з використанням класів BufferedImage, Canvas, Timer та TimerTask; получили уявлення про стандарт DICOM та інформацію, з якою він дозволяє працювати; організували показ відеоряду зображень з набору DICOM шляхом періодичного відображення послідовних кадрів.

## **Контрольні запитання:**

1. **Опишіть класи FTPClient та BufferedImage.**

FTPClient - з’єднує із віддаленими серверами через FTP протоколи.

BufferedImage - використовується для обробки та керування даними зображення

1. **Для чого призначені методи paint() та update() класу Canvas та в чому полягає взаємозв'язок між ними?**

paint() - малюває на віджеті Canvas, за заданим контекстом

update() - оновлює зображення

Вони взаємодіють між собою. Після виклику методу paint(), для обновлення зображення, треба викликати метод update()

1. **Які задачі і яким чином дозволяє вирішувати клас Timer? Що скасовує його метод cancel(), а що – ні?**

Timer використовується для створення таймеру на певні задачі, під час цього створює власний потік.

Метод сancel() скасовує потік запланованих задач, але не перериває поточну

1. **Надайте характеристику класу TimerTask та його властивостям.**

TimerTask – запланована до виконання задача, яка необхідна бути екземпляром TimerTask. Спочатку створюється TimerTask, а потім плануєтьсяйого запуск за допомогою класу Timer. TimerTask – запланована до виконання задача, яка необхідна бути екземпляром TimerTask. Спочатку створюється TimerTask, а потім плануєтьсяйого запуск за допомогою класу Timer.

1. **Розкрийте сутність асинхронної організації виконання дій та вкажіть, в чому полягяють її переваги над синхронною.**

Асинхронна організація виконання дій відноситься до парадигми програмування, де виконання завдань не обмежується послідовним виконанням кроків, а здійснюється паралельно або непослідовно, без очікування результату кожної дії перед переходом до наступної.

1. **Чим відрізняється дійсна паралельність виконання від псевдопаралельності та при якій апаратній конфігурації перша з них неможлива і чому?**

Різниця полягає в тому, що дійсна паралельність – дійсно робить паралельно завдання, а псевдопаралельність – імітує дане виконання, насправді виконуючи задачу послідовно. Під час паралельності процесорний час може надаватися декільком додаткам, вона неможлива у випадку використання одноядерного процесору з одним потоком, мова йде про дійсну паралельність.

1. **В чому полягають переваги багатопотоковості над послідовним виконанням дій та які є дві основні проблеми її використання? Що означає термін «потокобезпечний» («thread-safe»)?**

Переваги багатопотоковості над послідовним виконанням дій:

* Покращена продуктивність. Використання багатопотоковості дозволяє виконувати кілька завдань одночасно, розділяючи їх на окремі потоки. Це забезпечує краще використання доступних ресурсів системи, таких як процесорний час і пам'ять, і зменшує час виконання загального завдання.
* Покращена відповідність. Багатопотокові програми можуть бути більш реактивними і швидше реагувати на події або запити.

Дві проблеми :

* синхронізація даних між потоками;
* можливість одночасного доступу до ресурсів.

Термін "потокобезпечний" (thread-safe) вказує на те, що певний код або об'єкт може бути безпечно використовуваний одночасно кількома потоками без виникнення проблем змагання за ресурси чи синхронізації. Потокобезпечність досягається шляхом правильного управління доступом до спільних даних та ресурсів, таким чином, щоб уникнути конфліктів і неправильних результатів в багатопотоковому середовищі.

**8. Які засоби та механізми надає Java для паралельного виконання дій та яким класом вони забезпечуються?**

Java надає різні механізми Therad, Runnable, Callable для паралельного виконання дій. Головний потік (точка входу в програму та весь її код, що не звертається до классів Thread) виконується одним потоком. Але головний потік також може створити нові потоки за допомгою Thread

**9. Що являє собою стандарт DICOM та для роботи з якою інформацією він призначений?**

Стандарт DICOM (Digital Imaging and Communications in Medicine) розробленим для обміну, зберігання та обробки медичних зображень та відповідної інформації для використання в різних медичних пристроях.

**10. Сформулюйте своє розуміння щодо відповідального відношення до розробки технологій, які призначені для використання в медицині та інших важливих галузях, на прикладі ситуації з апаратом «Therac-25» та аналогічних випадків, що Вам відомі.**

Відповідальне відношення до розробки технологій, призначених для використання в медицині та інших важливих галузях, є надзвичайно важливим і повинно бути покладене в основу всього процесу розробки. У таких сферах, де життя та здоров'я людей залежать від технологій, незаперечно потрібно приділяти велику увагу безпеці, надійності та ефективності. Ситуація з апаратом «Therac-25» може служити ілюстрацією того, що може трапитися, якщо не враховувати відповідальність під час розробки медичних технологій. Therac-25 був медичним пристроєм для лікування раку, який в 1980-х роках став причиною серії смертельних помилок. Ці помилки виникали через недоліки в проектуванні та програмному забезпеченні апарата, що призводило до передачі небезпечно великих доз радіації пацієнтам.