Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»



ПРАКТИКА НАВЧАЛЬНА

Звіт з практичного завдання «Графічний редактор»

Підготував студент

1-го року навчання

Факультету інформатики

Спеціальність «Інженерія програмного забезпечення»

1 група

Накитняк Вадим Ігорович

Викладач

Пєчкурова Оксана Миколаївна

Київ – 2019

Зміст

[Постановка задачі 3](#_Toc9915110)

[Структура програми 4](#_Toc9915111)

[Опис методів та класів 5](#_Toc9915112)

[Інструкція користувача 20](#_Toc9915113)

[Висновок 29](#_Toc9915114)

[Посилання 29](#_Toc9915115)

[Лістинг коду 30](#_Toc9915116)

Список ілюстрацій

[Рисунок 1. Приклад виконання 2](#_Toc9940332)

[Рисунок 2. Структура класів програми 3](#_Toc9940333)

[Рисунок 3. Методи та поля класа GEFrame 5](#_Toc9940334)

[Рисунок 4. Методи класу CanvasPanel 8](#_Toc9940335)

[Рисунок 5. Поля та методи класу ToolbarPanel 11](#_Toc9940336)

[Рисунок 6. Методи та поля класу ScrollSidePanel 12](#_Toc9940337)

[Рисунок 7. Поля та методи класу PreviewPanel 13](#_Toc9940338)

[Рисунок 8. Enum Tool 14](#_Toc9940339)

[Рисунок 9. Interface Drawable 15](#_Toc9940340)

[Рисунок 10. Поля та методи класа DrawnCircle 15](#_Toc9940341)

[Рисунок 11. Поля та методи класа DrawnImage 16](#_Toc9940342)

[Рисунок 12. Поля та методи класу DrawnLine 17](#_Toc9940343)

[Рисунок 13. Поля та методи класу DrawnRectangle 17](#_Toc9940344)

[Рисунок 14. Поля та методи класу DrawnText 18](#_Toc9940345)

[Рисунок 15. Поля та методи класу DrawnTriangle 19](#_Toc9940346)

[Рисунок 16. Методи та поля класу ImageTransfarable 20](#_Toc9940347)

[Рисунок 17. Стартове вікно 21](#_Toc9940348)

[Рисунок 18. Панель ToolbarPanel 21](#_Toc9940349)

[Рисунок 19. Панель ScrollSidePanel 22](#_Toc9940350)

[Рисунок 20. Панель CanvasPanel 22](#_Toc9940351)

[Рисунок 21. Вставка зображення 23](#_Toc9940352)

[Рисунок 22. Зміна фону 23](#_Toc9940353)

[Рисунок 23. Вставка тексту 24](#_Toc9940354)

[Рисунок 24. Гумка 24](#_Toc9940355)

[Рисунок 25. Кнопка Pipette 25](#_Toc9940356)

[Рисунок 26. Піпетка 25](#_Toc9940357)

[Рисунок 27. Фігури, пензель, колір та штрих 26](#_Toc9940358)

[Рисунок 28. PreviewPanel 26](#_Toc9940359)

[Рисунок 29. Зміна масштабу 1 27](#_Toc9940360)

[Рисунок 30. Зміна масштабу 2 27](#_Toc9940361)

[Рисунок 31. MenuBar Edit 28](#_Toc9940362)

[Рисунок 32. Виділення зони 28](#_Toc9940363)

[Рисунок 33. Вирізання та вставка 29](#_Toc9940364)

[Рисунок 34. MenuBar File 29](#_Toc9940365)

Постановка задачі

**Графічний редактор**

Створити (мінімум 15 дій)

Обов'язкові функції:

1) Меню

2) Відкриття файлу

3) Збереження файлу

4) Скрол зображення

5) Збільшення та зменшення шрифту

6)Вставка фото, тексту, …

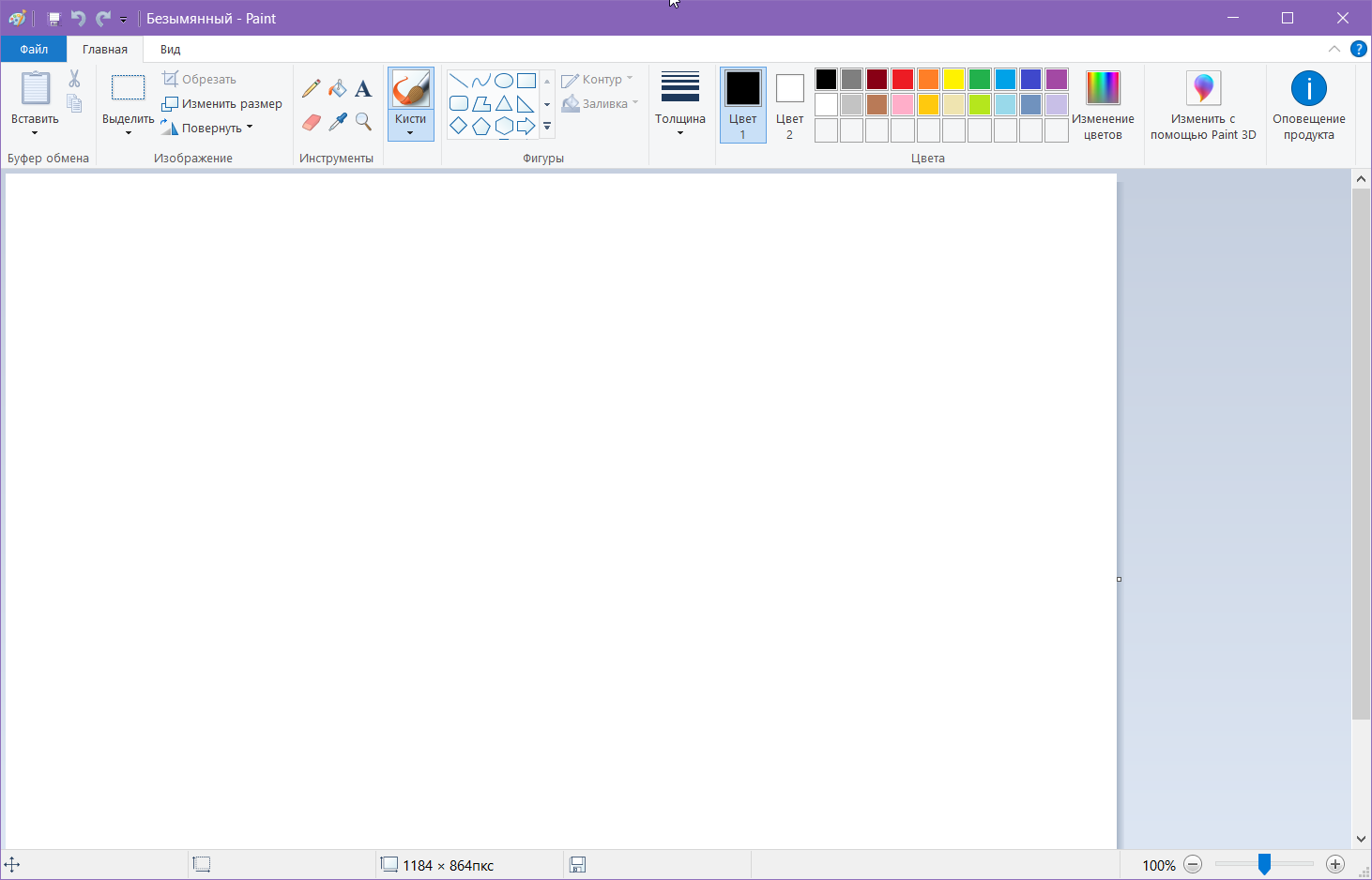


Рисунок 1. Приклад виконання

Структура програми

Для створення графічного інтерфейсу програми було використано стандартні бібліотеки **Java: java.swing та java.awt**. Як середу розробки я обрав **Apache NetBeans IDE 10.0**, завдяки чому набагато спростив собі створення інтерфейсу програми.

Рисунок 2. Структура класів програми

Програма складається з 14 класів (12 class, 1 enum, 1 interface). **GEFrame** – головний фрейм, на якому розміщено 3 панелі, а також меню бар. На **ToolbarPanel** в свою чергу розміщено ще одну панель **PreviewPanel**. А в класі **CanvasPanel** використовуються наступні класи : **DrawnCircle**, **DrawnImage**, **DrawnRectangle**, **DrawnText**, **DrawnTriangle**, DrawnLine – які реалізують інтерфейс **Drawable**. В класі(enum) Tool є перерахування всіх інструментів для малювання, які використовуються в класі **CanvasPanel**, та переключаються в класі **ToolbarPanel**.

Опис методів та класів

Головним класом програми є public class **EGFrame**(extends **JFrame**) – фрейм, на якому розміщено всі інші панелі, такі як CanvasPanel, ToolbarPanel, **ScrollSidePanel**. Цей клас зберішає посилання на ці панелі, а також значення x та y для визначення розміщення миші на фреймі. Щодо методів :

**public GEFrame()** – Конструктор, в якому створюються і додаються панелі, а також ініціалізуються змінні.;

**initComponents()** – метод, що ініціалізує об’єкти на панелі;

**private void formMouseClicked(java.awt.event.MouseEvent evt)** – Метод, який спрацювує при натисканні на фрейм мишкою. Зчитує координати мишки, а встановлює колір на полотні(**CanvasPanel**) в залежності від кольору пікселя(За умови, що вімкнено **Tool.PIPETTE**);

**private void backItemActionPerformed(java.awt.event.ActionEvent evt)** – Метод-слухач для пункту меню **Edit**, який скасовує минулий об’єкт, доданий користувачем на полотно. Встановлено мнемоніку Ctrl + Z;

**private void forwardItemActionPerformed(java.awt.event.ActionEvent evt)** – Метод-слухач для пункту меню **Edit**, який відновлює скасований користувачем об’єкт. Встановлено мнемоніку Ctrl + Y;

**private void openItemActionPerformed(java.awt.event.ActionEvent evt)–** Метод-слухач для пункту меню **File**, який відкриває нове зображення на полотні. Встановлено мнемоніку Ctrl + O;

**private void saveItemActionPerformed(java.awt.event.ActionEvent evt)** –Метод-слухач для пункту меню **File**, який зберігає зображення полотна у директорію **savedPictures**. Користувач має ввести назву файлу перед тим як зберегти. Встановлено мнемоніку Ctrl + S;

**private void saveAsItemActionPerformed(java.awt.event.ActionEvent evt)** –Метод-слухач для пункту меню **File**, який зберігає зображення полотна у вибраний користувачем файл. Встановлено мнемоніку Ctrl + A;

**private void newPictureItemActionPerformed(java.awt.event.ActionEvent evt)** – Метод-слухач для пункту меню **File**, який відкриває нове полотне, при цьому стираючи старе. Встановлено мнемоніку Ctrl + N;

**private void aboutItemActionPerformed(java.awt.event.ActionEvent evt)** –Метод-слухач для пункту меню **File**, який виводить користувачу коротку інформацію про програму;

**private void exitItemActionPerformed(java.awt.event.ActionEvent evt)** – Метод-слухач для пункту меню **File**, який примусово закриває програму. Встановлено мнемоніку Ctrl + E.

**private void insertItemActionPerformed(java.awt.event.ActionEvent evt)** – Метод-слухач для пункту меню **Edit**, який вставляє хображення на полотно із буферу обміну. Встановлено мнемоніку Ctrl + V;

**private void cutMenuActionPerformed(java.awt.event.ActionEvent evt)** - Метод-слухач для пункту меню **Edit**, встановлю поточним інструментом **CUT\_ZONE**. Встановлено мнемоніку Ctrl + X;

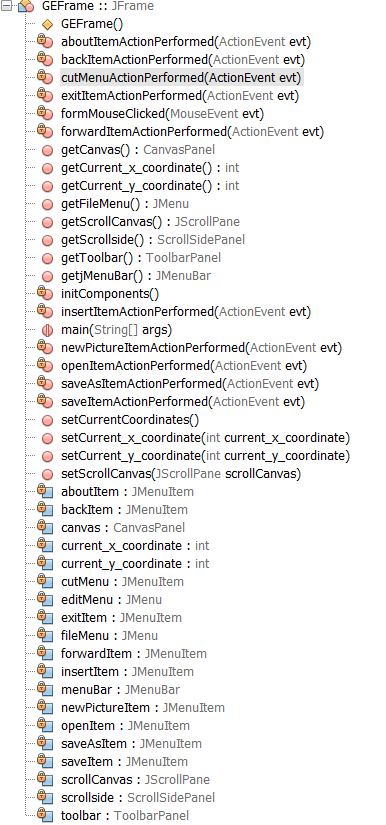


Рисунок 3. Методи та поля класа GEFrame

**public class CanvasPanel(extends JPanel)**. Панель, на якій розміщено полотно для малювання. Клас перенасичено всілякими змінними, які допомагають вималювати бажану картинку. В колекцію **shapes**(ArrayList<Drawable>) кожен раз зберігаються створені користувачем фігури, текст, зображення. Потім, в методі **paintComponent(Graphics g)** з колекцію циклічно дістається кожен об’єкт класу, що наслідую інтерфейс **Drawable**, та вимальовується згідно параметрам. Щодо методів :

**public CanvasPanel(GEFrame frame)** – Конструктор, який приймає посилання на фрейм та ініціалізує всі змінні;

**initComponents()** – метод, що ініціалізує об’єкти на панелі;

**private void formMousePressed(java.awt.event.MouseEvent evt)** – Метод, який викликається при натисканні мишкою по панелі. Він вималювує початковий контур фігур, а також присвоєю змінним типу boolean **isMouseClicked**, **isMouseOnImage**, **isMouseOnText** значення true;

**private void formMouseDragged(java.awt.event.MouseEvent evt)** – Метод, який викликається при перетаскуванні зажатої мишки по панелі. Він продовжує вимальовувати фігуру, в залежності від змінної tool. Я реалізував це за допомогою оператора множинного вибору switch case;

**private void formMouseReleased(java.awt.event.MouseEvent evt)** – Метод, який спрацьовує при відтисканні зажатої мишки. Змінній типу boolean **isMouseClicked** присваюється значення false, а також спрацьовує метод **reset();**

**public void reset()** – Метод, який обнуляє всі тимчасово створені змінні типу **Drawable** **shape**, **text** та **image**, а також присвоєю змінним типу boolean **isMouseOnImage**, **isMouseOnText** значення false;

**protected void paintComponent(Graphics g)** – Метод, завдяки якому здійснюється малювання на панелі. Після настройки кольору та ширини штриху, викликаються методи **drawAllShapes(g2d), drawShape(g2d), drawImage(g2d), drawText(g2d)** для промальовки об’єктів;

**public void drawAllShapes(Graphics2D g2d)** – Метод, який промальовує всі об’єкти з колекції **shapes**;

**public void drawZone(Graphics2D g2d) –** Метод, який вимальовує зону видалення;

**public void drawShape(Graphics2D g2d)** – Метод, який промальовує тимчасово створену фігуру;

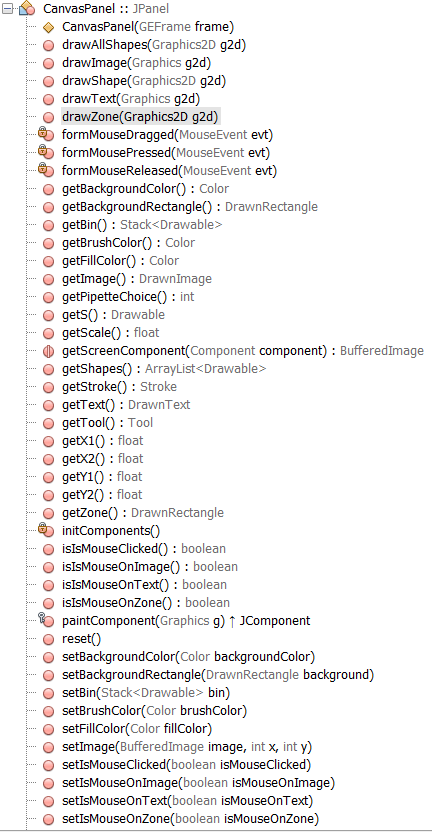
**public void drawImage(Graphics g2d) –** Метод, який промальовує тимчасово створене зображення;

**public void drawText(Graphics g2d)** – Метод, який промальовує тимчасово створений текст;

**public static BufferedImage getScreenComponent(Component component) –** Статичний метод , який робить зображення компоненту, та вертає об’єкт типу BufferedImage.

**isIsMouseClicked(), getShapes(), getX1(), getY1(), getX2(), getY2(), getS(), getBackgroundRectangle(), getBrushColor(), getBackgroundColor(), getFillColor(), getStroke(), getTool(), getScale(), getPipetteChoice(), getImage(), getText(), getBin(), isIsMouseOnImage(), isIsMouseOnText(), public DrawnRectangle getZone(), public boolean isIsMouseOnZone()** – Методи-геттери;

**setIsMouseClicked(boolean isMouseClicked), setShapes(ArrayList<Drawable> shapes), setX1(int x1), setY1(int y1), setX2(int x2), setY2(int y2), setS(Drawable s), setBackgroundRectangle(DrawnRectangle background), setBrushColor(Color brushColor), setBackgroundColor(Color backgroundColor), setFillColor(Color fillColor), setStroke(Stroke stroke), setTool(Tool tool), setScale(float scale), setPipetteChoice(int pipetteChoice), setImage(BufferedImage image, int x, int y), setText(DrawnText text), setIsMouseOnImage(boolean isMouseOnImage), setIsMouseOnText(boolean isMouseOnText), setBin(Stack<Drawable> bin), public void setZone(DrawnRectangle zone, setIsMouseOnZone(boolean isMouseOnZone)** – Методи-сеттери;



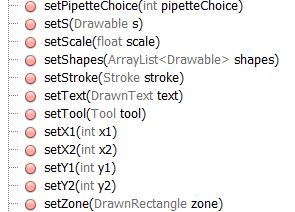


Рисунок 4. Методи класу CanvasPanel

**public class ToolbarPanel(extends JPanel).** Панель для інструментів(Вставка зображення, зміна фону, вставка тексту, пензель, гумка, піпетка, вставка фігури, зміна кольору та ширини штриху) Методи:

**public ToolbarPanel(GEFrame frame)** - Конструктор, який приймає посилання на фрейм та ініціалізує всі змінні;

**initComponents()** – метод, що ініціалізує об’єкти на панелі;

**private void brushButtonMouseReleased(java.awt.event.MouseEvent evt)** – Метод- слухач, що спрацьовує при натисканні на кнопку **Brush**. Встановлює поточний інструмент як **Brush**;

**private void eraserButtonMouseReleased(java.awt.event.MouseEvent evt)** – Метод-слухач, що спрацьовує при натисканні на кнопку **Eraser**. Встановлює поточний інструмент як **Eraser**;

**private void chooseColorButtonMouseReleased(java.awt.event.MouseEvent evt)** – Метод-слухач, що спрацьовує при натисканні на кнопку **Choose Color**. Встановлює поточний колір пензля за бажанням користувача;

**private void pipetteButtonMouseReleased(java.awt.event.MouseEvent evt)** – Метод-слухач, що спрацьовує при натисканні на кнопку **Pipette**. Дає вибір типу кольору користувачу. Встановлює поточний інструмент як **Pipette**;

**private void formMouseClicked(java.awt.event.MouseEvent evt)** – Метод-слухач, що реагує на натискання по панелі. Встановлює колір в залежності від пікселя, на який натиснув користувач;

**private void setBackgroundButtonMouseClicked(java.awt.event.MouseEvent evt)** – Метод-слухач, що спрацьовує при натисканні на кнопку **Set Background**. Дає вибір кольору фону і встановлює його;

**private void strokeSpinnerStateChanged(javax.swing.event.ChangeEvent evt)** - Метод-слухач, що спрацьовує при зміненні значення лічільника **strokeSpinner**. Встановлює ширину штриху в залежності від значення;

**private void fillColorButtonMouseReleased(java.awt.event.MouseEvent evt) -** Метод-слухач, що спрацьовує при натисканні на кнопку **Fill Color**. Встановлює колір заливки фігур;

**private void rectangleButtonMouseReleased(java.awt.event.MouseEvent evt)** - Метод-слухач, що спрацьовує при натисканні на кнопку з чотирикутником. Встановлює поточний інструмент як **Shape\_Rectangle**;

**private void insertPictureButtonMouseReleased(java.awt.event.MouseEvent evt)** - Метод-слухач, що спрацьовує при натисканні на кнопку **Insert Picture**. Користувач вибирає, яке зображення він хоче розмістити на полотно;

**private void insertTextButtonMouseReleased(java.awt.event.MouseEvent evt)** - Метод-слухач, що спрацьовує при натисканні на кнопку **Insert Picture.** Пропонує користувачу на вибір зредагувати текст та вивести його на єкран;

**private void circleButtonMouseReleased(java.awt.event.MouseEvent evt)** - Метод-слухач, що спрацьовує при натисканні на кнопку з колом. Встановлює поточний інструмент як **Shape\_Circle**;

**private void triangleButtonMouseReleased(java.awt.event.MouseEvent evt)** - Метод-слухач, що спрацьовує при натисканні на кнопку з трикутником. Встановлює поточний інструмент як **Shape\_Triangle**;

**private void lineButtonMouseReleased(java.awt.event.MouseEvent evt**) - Метод-слухач, що спрацьовує при натисканні на кнопку з лінією. Встановлює поточний інструмент як **Shape\_Line**;

**private void fillCheckBoxMouseReleased(java.awt.event.MouseEvent evt)** - Метод-слухач, що спрацьовує при натисканні на прапорець **Fill**. В залежності від значенні заливає, або ні фігури;

**public void updatePreview()** – Метод, який оновлює PreviewPanel в залежності від поточних параметрів у класі **CanvasPanel**;

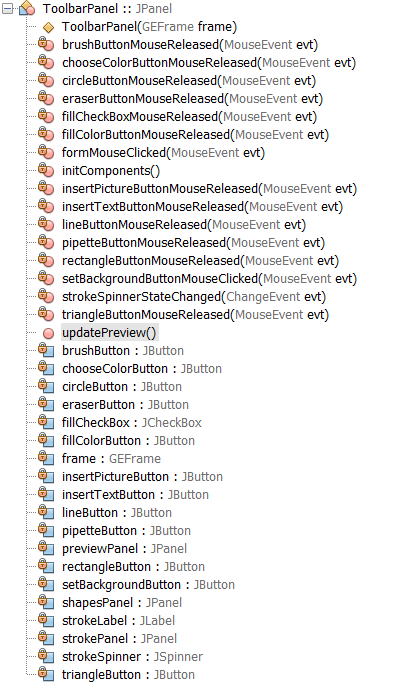


Рисунок 5. Поля та методи класу ToolbarPanel

**public class ScrollSidePanel(extends JPanel)**. Панель, на якій розміщено смугу прокрутки для зміни масштабу зображення, а також позначки з назвою програми та іменем автора. Методи:

**public ScrollSidePanel(GEFrame frame)** - Конструктор, який приймає посилання на фрейм та ініціалізує всі змінні;

**initComponents()** – метод, що ініціалізує об’єкти на панелі;

**private void formMouseClicked(java.awt.event.MouseEvent evt)** - Метод-слухач, що реагує на натискання по панелі. Встановлює колір в залежності від пікселя, на який натиснув користувач;

**private void sliderStateChanged(javax.swing.event.ChangeEvent evt)** – Метод-слухач, який спрацьовує при зміні значення смуги прокрутки. Встановлює масштаб полотна в залежності віз значення;

**private void sliderMouseWheelMoved(java.awt.event.MouseWheelEvent evt)** – Метод-слухач, що реагує на прокрутку колеса. Встановлює значення смуги прокрутки в залежності від дій користувача.

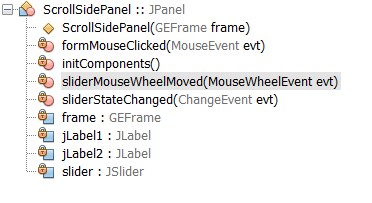


Рисунок 6. Методи та поля класу ScrollSidePanel

**public class PreviewPane(extends JPanel)**. Панель для предперегляду того, який інструмент наразі обраний користувачем. Методи:

**public PreviewPanel(Tool tool, Stroke stroke, Color color, Color fillColor)** – Конструктор, який приймає параметри для налаштування малювання і ініціалізує ними змінні;

**initComponents()** – метод, що ініціалізує об’єкти на панелі;

**protected void paintComponent(Graphics g)** – Метод, який вимальовує потрібний інструмент на маленькомй чотирикутничку в куточку;

**public void update(Tool tool, Stroke stroke, Color color, Color fillColor)** – Метод, який оновлює параметри для малювання;

**getTool(), getStroke(), getColor(), getFillColor()** – Методи-геттери;

**setTool(Tool tool), setStroke(Stroke stroke), setColor(Color color), setFillColor(Color fillColor)** – Методи-сеттери.

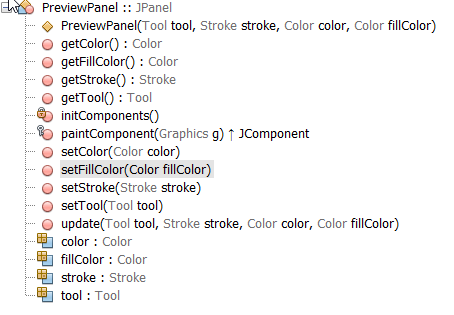


Рисунок 7. Поля та методи класу PreviewPanel

**public enum Tool**. Перерахування всіх інструментів, які використані в програмі.

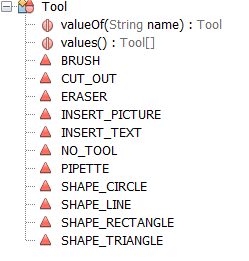


Рисунок 8. Enum Tool

**public interface Drawable**. Інтерфейс для всіх об’єктів, які малюються на полотні. Перелік методів :

**Stroke getStroke()** – метод вертає штрих;

**Color getColor()** – метод вертає колір пензля;

**Color getFillColor()** – метод вертає колір заливки;

**void setStroke(Stroke stroke)** – метод привласнює полю stroke значення;

**void setColor(Color color)** - метод привласнює полю color значення;

**void setFillColor(Color color)** - метод привласнює полю fillColor значення.

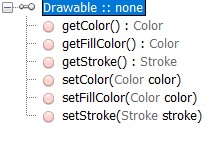


Рисунок 9. Interface Drawable

**public class DrawnCircle extends Ellipse2D.Double implements Drawable**. Клас кола, що розширює **Ellipse2D.Double** та реалізує інтерфейс **Drawable**(з усіма його методами. див. вище).

**public DrawnCircle(Stroke stroke, Color color, Color fillColor, double x, double y, double w, double h)** – Конструктор кола;

+ implemented **Drawable** methods(див. вище).

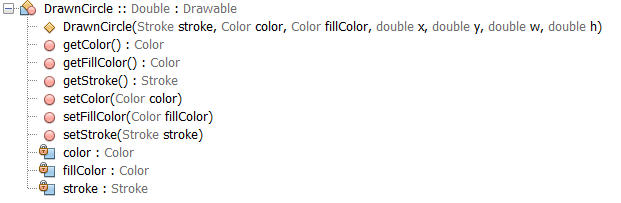


Рисунок 10. Поля та методи класа DrawnCircle

**public class DrawnImage extends BufferedImage implements Drawable**. Клас зображення, що розширює **BufferedImage** та реалізує інтерфейс **Drawable**.

**public DrawnImage(BufferedImage image, int x, int y),**

**public DrawnImage(int width, int height, int imageType)** – Конструктори зображення.

**getImage(), getX(), getY()** – Методи-геттери;

**setImage(BufferedImage image), setX(int x), setY(int y)** – Методи-сеттери;

+ implemented **Drawable** methods(див. вище).

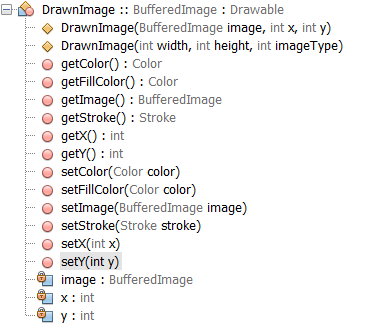


Рисунок 11. Поля та методи класа DrawnImage

**public class DrawnLine extends Line2D.Double implements Drawable.** Клас лінії, що розширює **Line2D.Double** та реаліхує інтерфейс **Drawable**.

**public DrawnLine(Stroke stroke, Color color, double x1, double y1, double x2, double y2)** – Конструктор лінії;

+ implemented **Drawable** methods(див. вище).

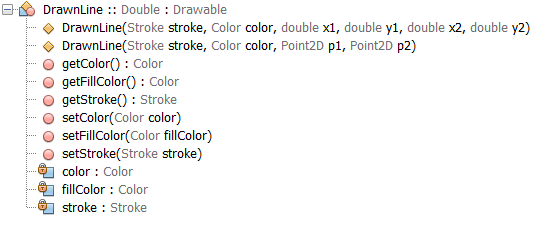


Рисунок 12. Поля та методи класу DrawnLine

**public class DrawnRectangle extends Rectangle implements Drawable.**Клас чотирикутника, що рощширює **Rectangle** та реалізує інтерфейс **Drawable**.

**public DrawnRectangle(Stroke stroke, Color color, Color fillColor, int x, int y, int width, int height)** – Конструктор чотирикутника;

+ implemented **Drawable** methods(див. вище).

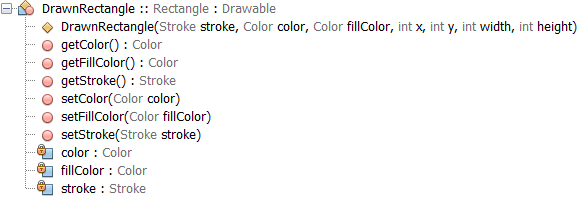


Рисунок 13. Поля та методи класу DrawnRectangle

**public class DrawnText implements Drawable**. Клас тексту, що реалізує інтерфейс **Drawable**.

**public DrawnText(Font font, String text, Color color, int x, int y)** – Конструктор тексту;

**getX(), getY(), getFont(), getText()** – Методи-геттери;

**setX(int x), setY(int y), setFont(Font font), setText(String text)** – Методи-сеттери;

+ implemented **Drawable** methods(див. вище).

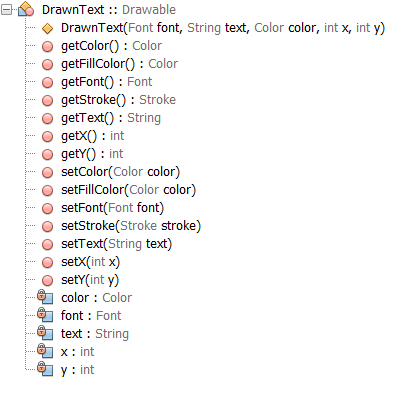


Рисунок 14. Поля та методи класу DrawnText

**public class DrawnTriangle extends Path2D.Double implements Drawable**. Клас трикутника, що розширює **Path2D.Double** та реалізує інтерфейс **Drawable**.

public DrawnTriangle(Stroke stroke, Color color, Color fillColor, double x1, double y1, double x2, double y2, double x3, double y3) – Конструктор трикутника;

+ implemented **Drawable** methods(див. вище).

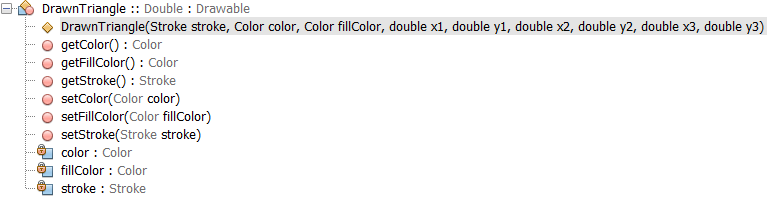


Рисунок 15. Поля та методи класу DrawnTriangle

**public class ImageTransfarable implements Transferable.** Клас, який реалізує інтерфейс Transfarable та перевизначає низку його методів. Я створив його, для того, щоб можна було вирізати та зберігати зображення в буфері обміну. Цей клас якраз і визначає тип зображень, які можна переміщати в та із буферу обміну. Методи:

public ImageTransfarable(Image image) – Конструктор, який приймає зображення, та на його основі створює новий єкземпляр **ImageTransfarable;**

public DataFlavor[] getTransferDataFlavors() – Метод вертає масив об’єктів типу DataFlovar, а сама об’єкт new DataFlavor[]{DataFlavor.imageFlavor};

\*DataFlovar - A DataFlavor provides meta information about data. DataFlavor is typically used to access data on the clipboard, or during a drag and drop operation.

public boolean isDataFlavorSupported(DataFlavor flavor) – Метод вертає true або false, в залежності від того, чи підтримується flavor в даному класі;

public Object getTransferData(DataFlavor flavor) throws UnsupportedFlavorException, IOException – Метод, який вертає об’єкт типу Image, якщо flavor підтримується в даному класі.

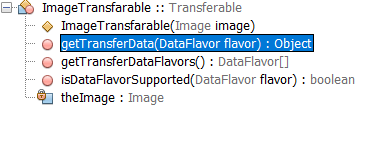


Рисунок 16. Методи та поля класу ImageTransfarable

Інструкція користувача

При запуску програму перед користувачем відкривається наступне вікно:

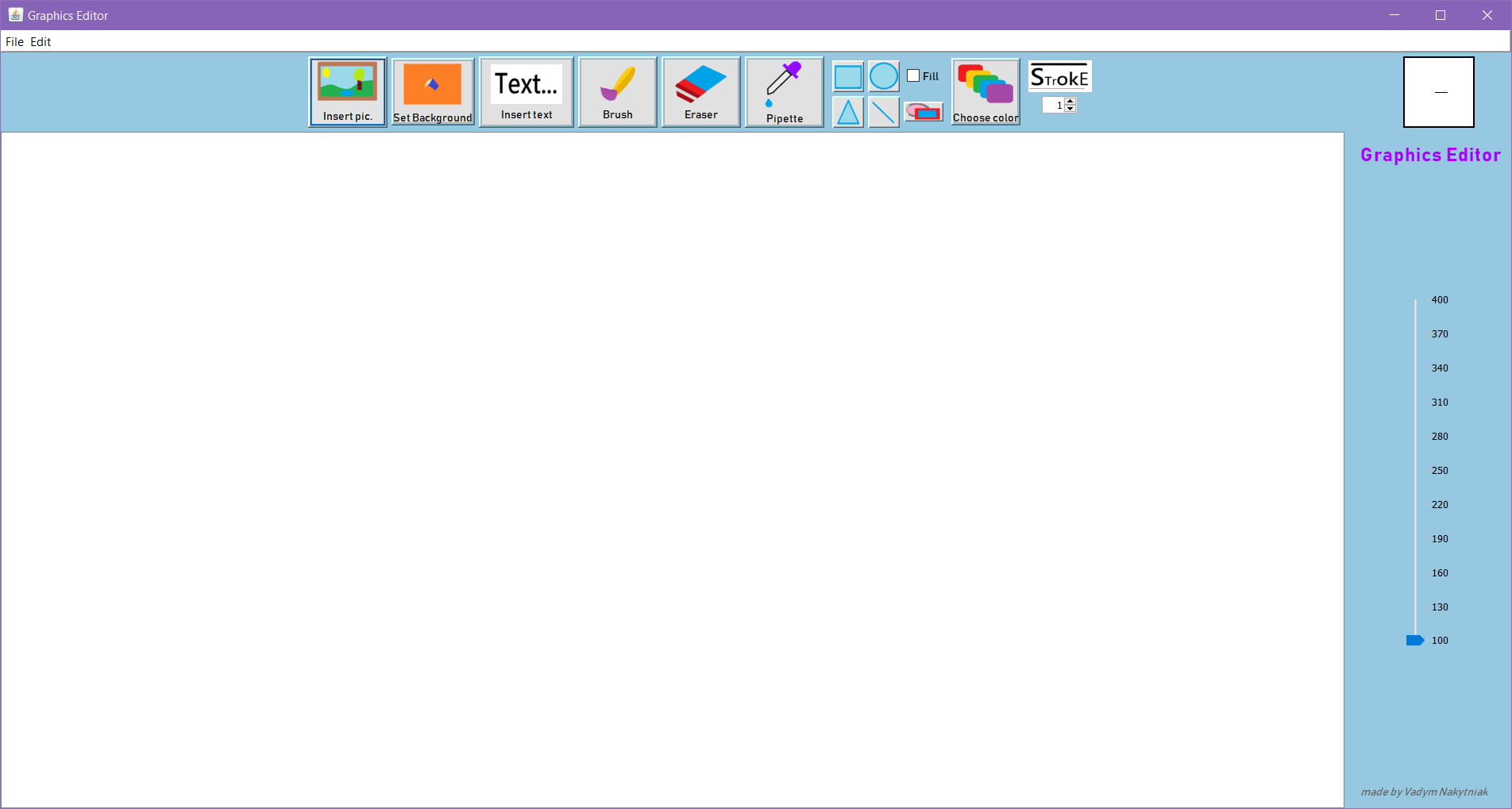


Рисунок 17. Стартове вікно

За бажанням, можна зменшувати та збільшувати вікно, але не менше за розмір панелі **ToolbarPanel**, що у верхній частині єкрану:

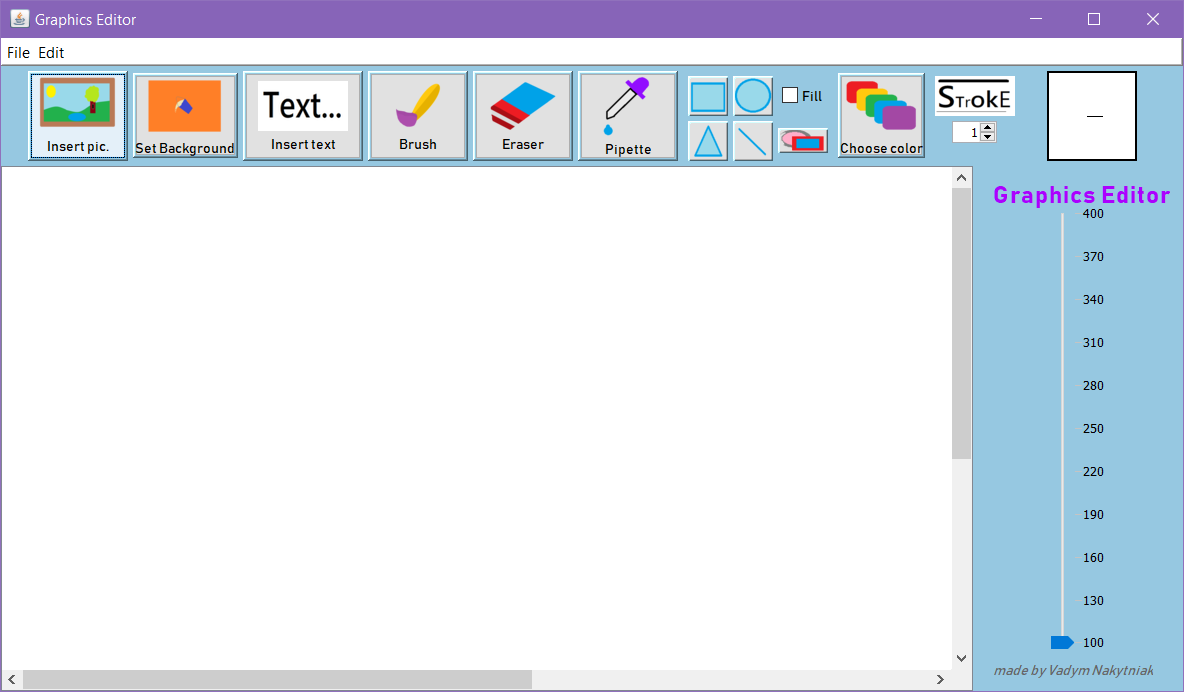


Рисунок 18. Панель ToolbarPanel

Справа розміщено панель **ScrollSidePanel** зі смугою прокрутки та позначками імені автора й назви програми:

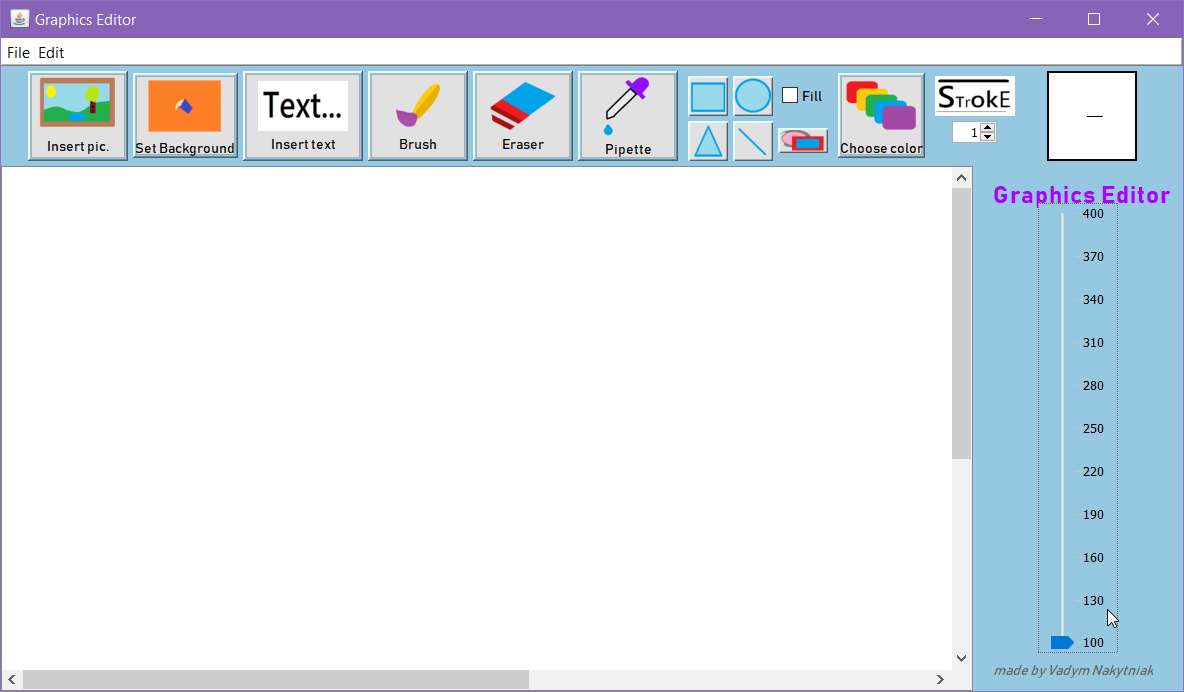


Рисунок 19. Панель ScrollSidePanel

А ось вже по центру розміщено полотно(панель **CanvasPanel**), на якій і будуть вимальовуватися об’єкти:

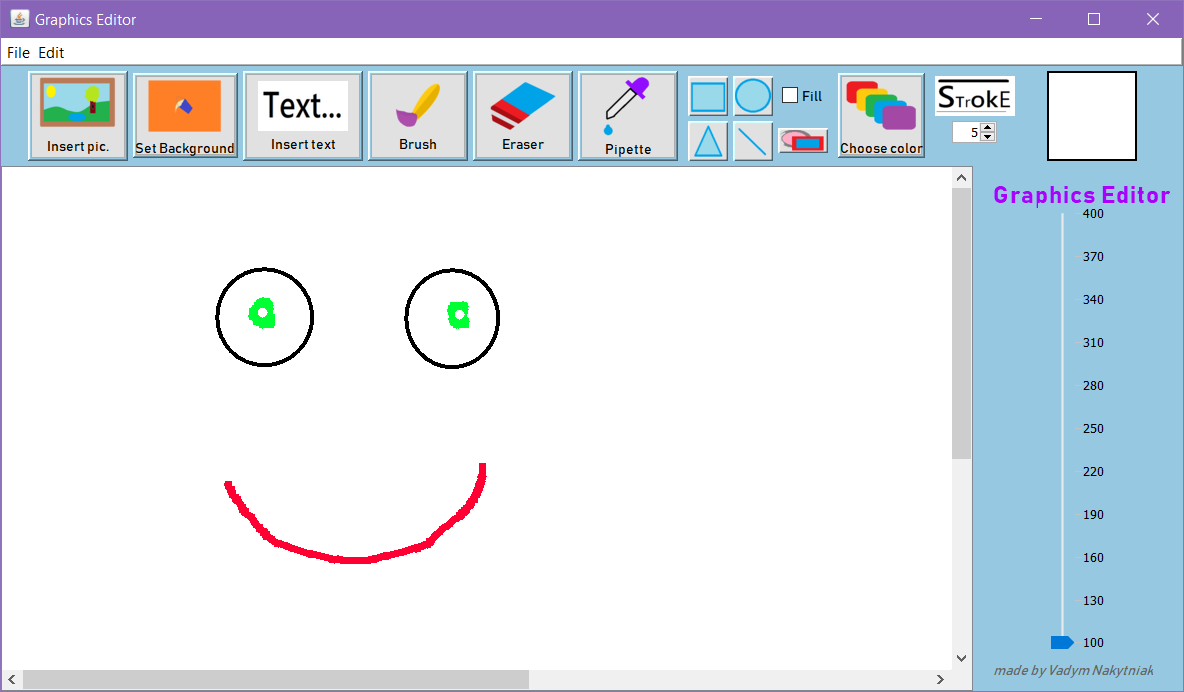


Рисунок 20. Панель CanvasPanel

Тепер окремо по кожному інструменту. Щойно вже було продемонстровано здібності пензля та вставки кола. А як щодо вставки зображення ?:

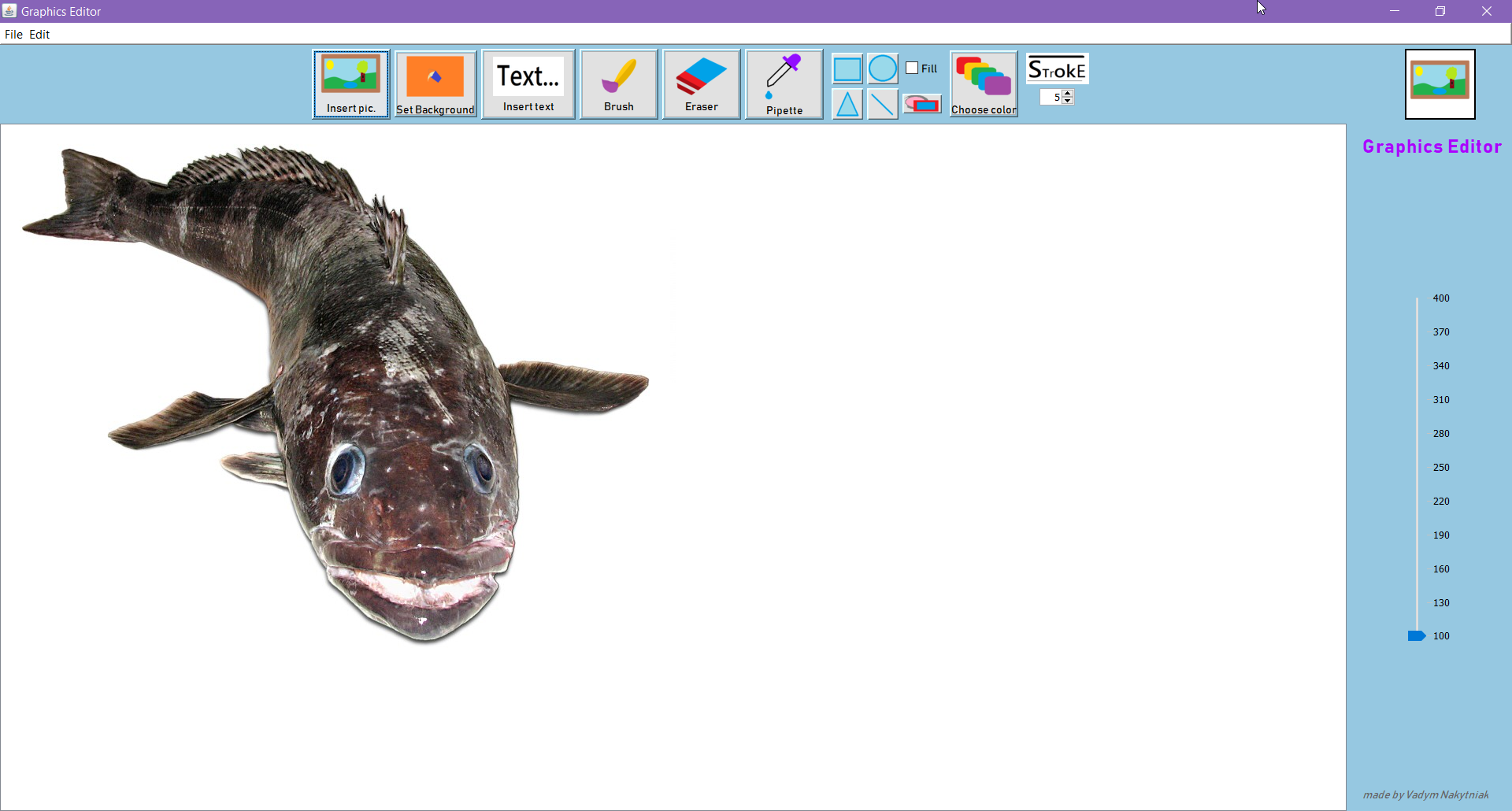


Рисунок 21. Вставка зображення

Зміна фону:

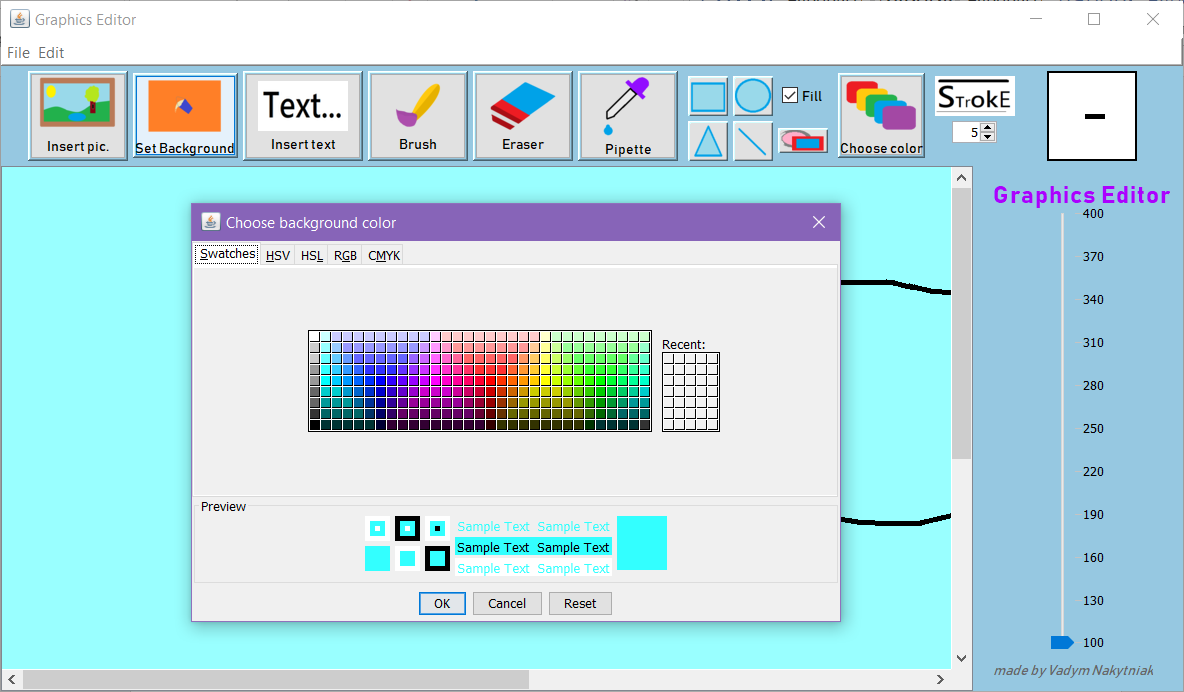


Рисунок 22. Зміна фону

Вставка тексту :

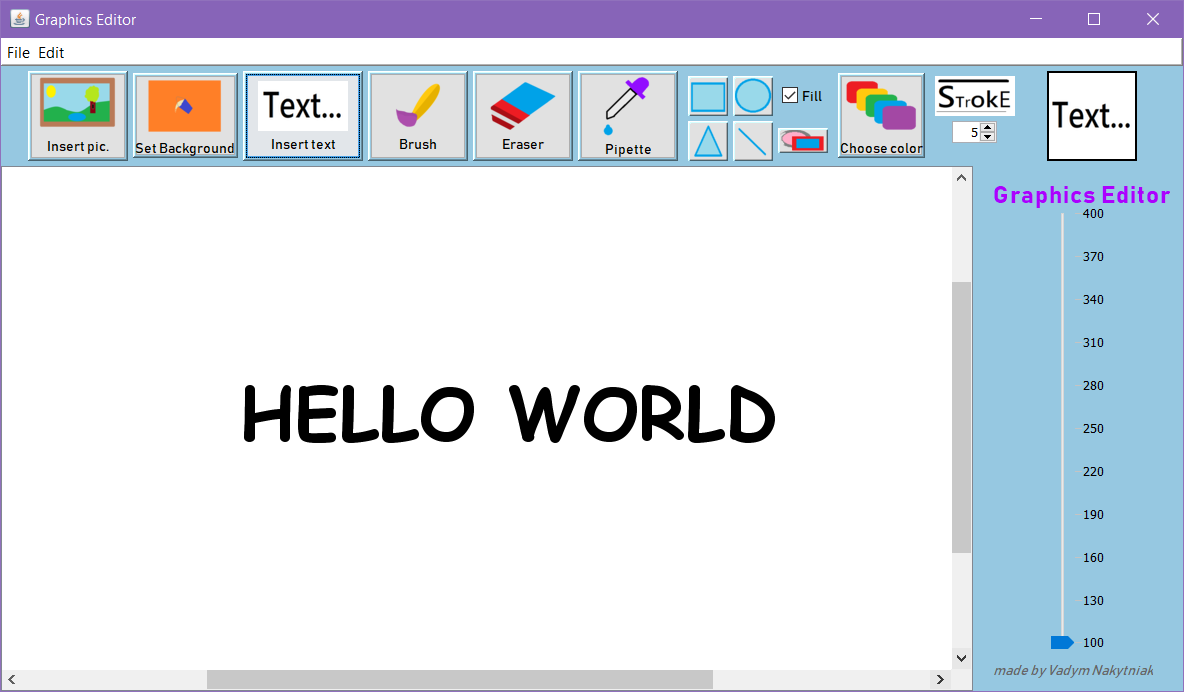


Рисунок 23. Вставка тексту

Гумка:

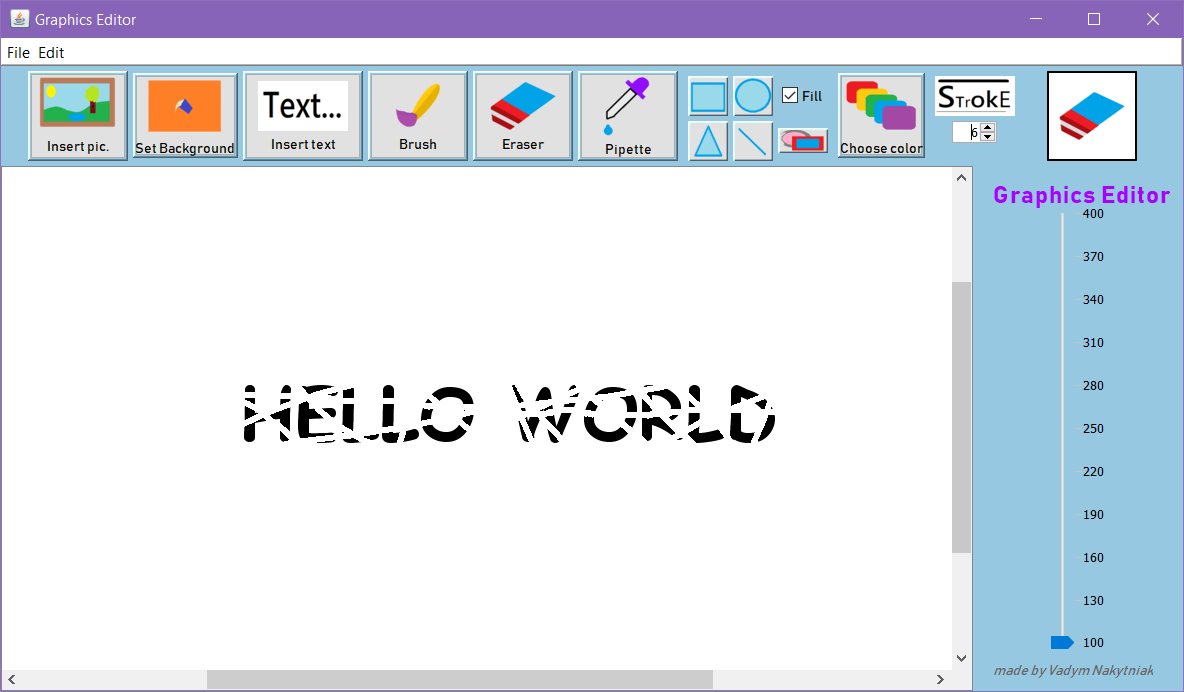


Рисунок 24. Гумка

При натисканні на піпетку, пропонується обрати, колір саме чого змінити:

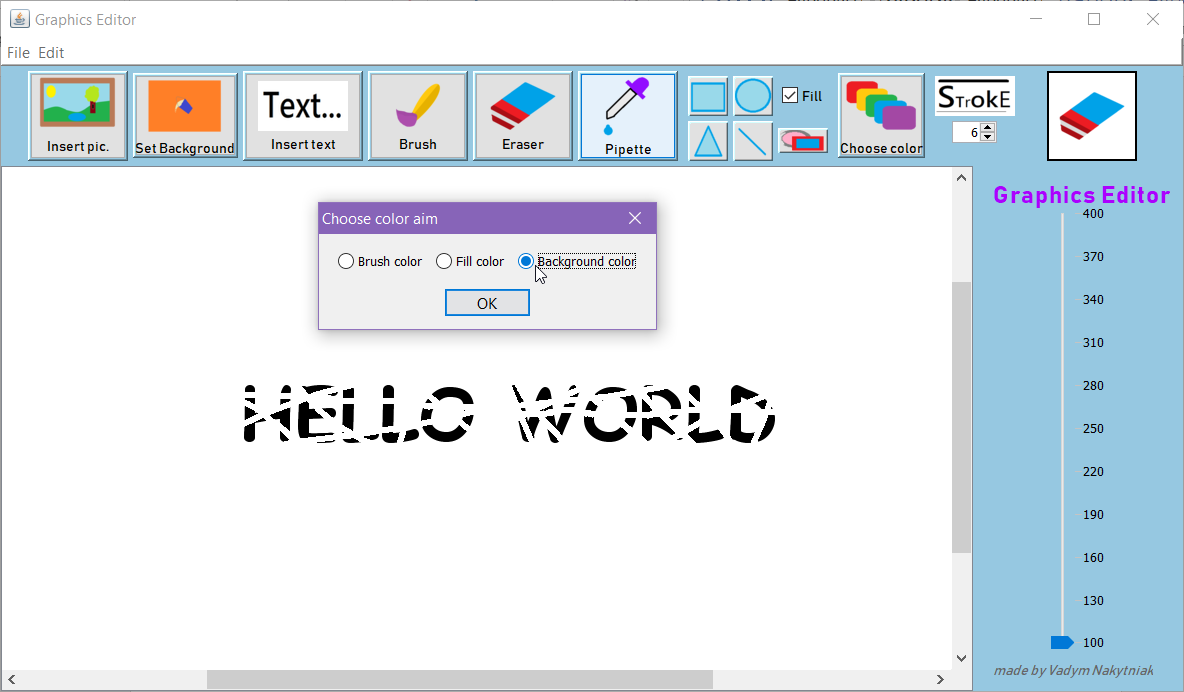


Рисунок 25. Кнопка Pipette

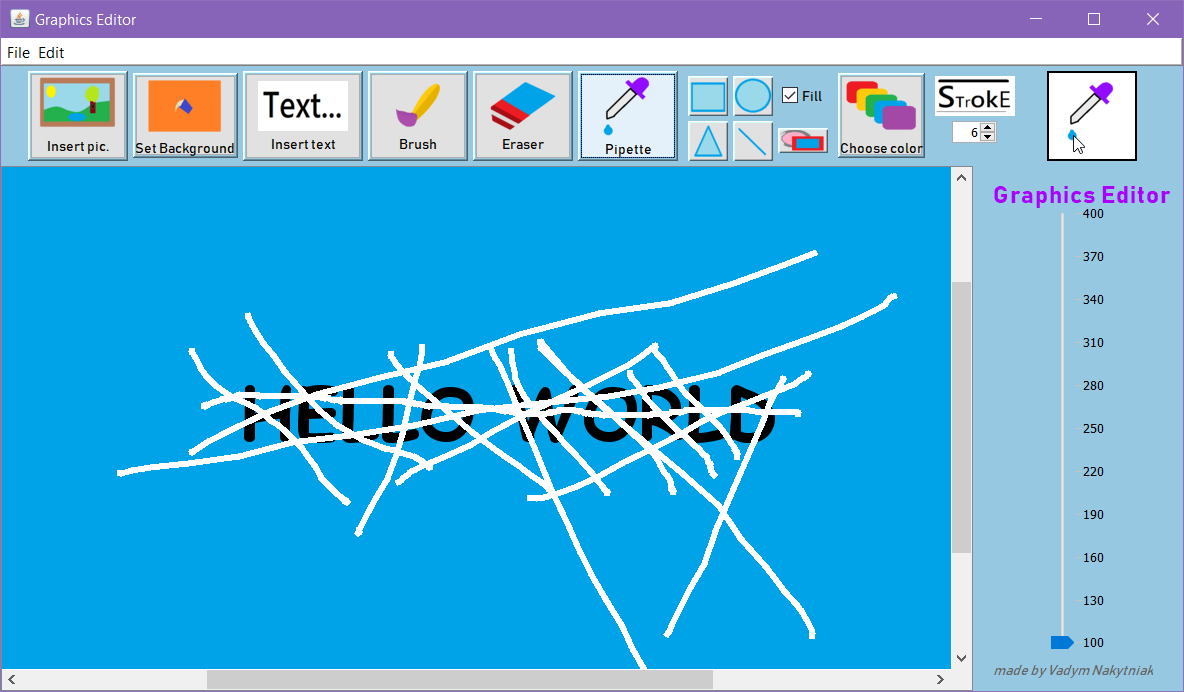


Рисунок 26. Піпетка

Вставка фігур та пензель(+ зміна кольору і штриху):

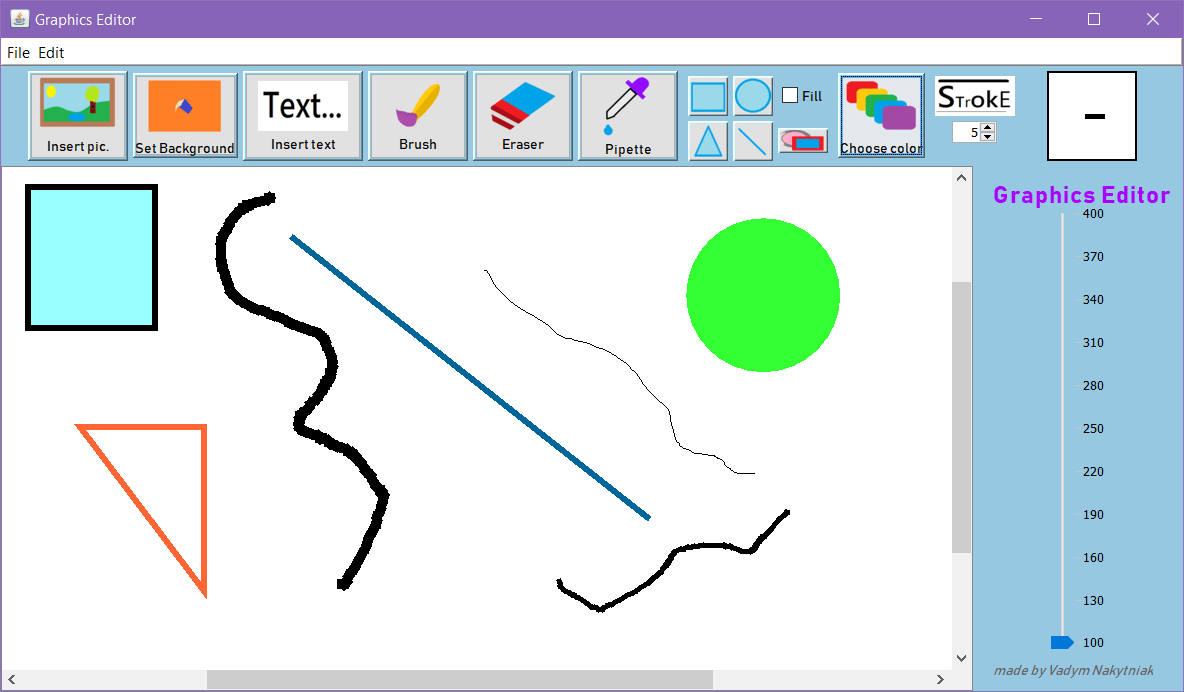


Рисунок 27. Фігури, пензель, колір та штрих

А ось і той самий PreviewPanel:

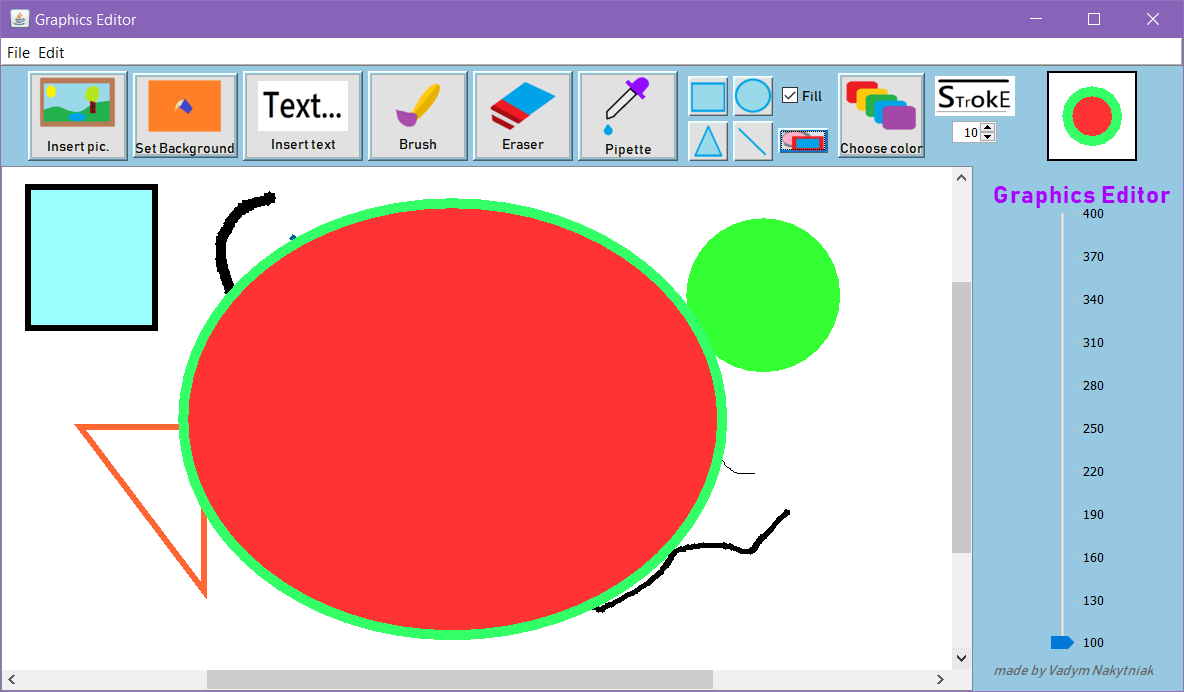


Рисунок 28. PreviewPanel

Зміна масштабу:

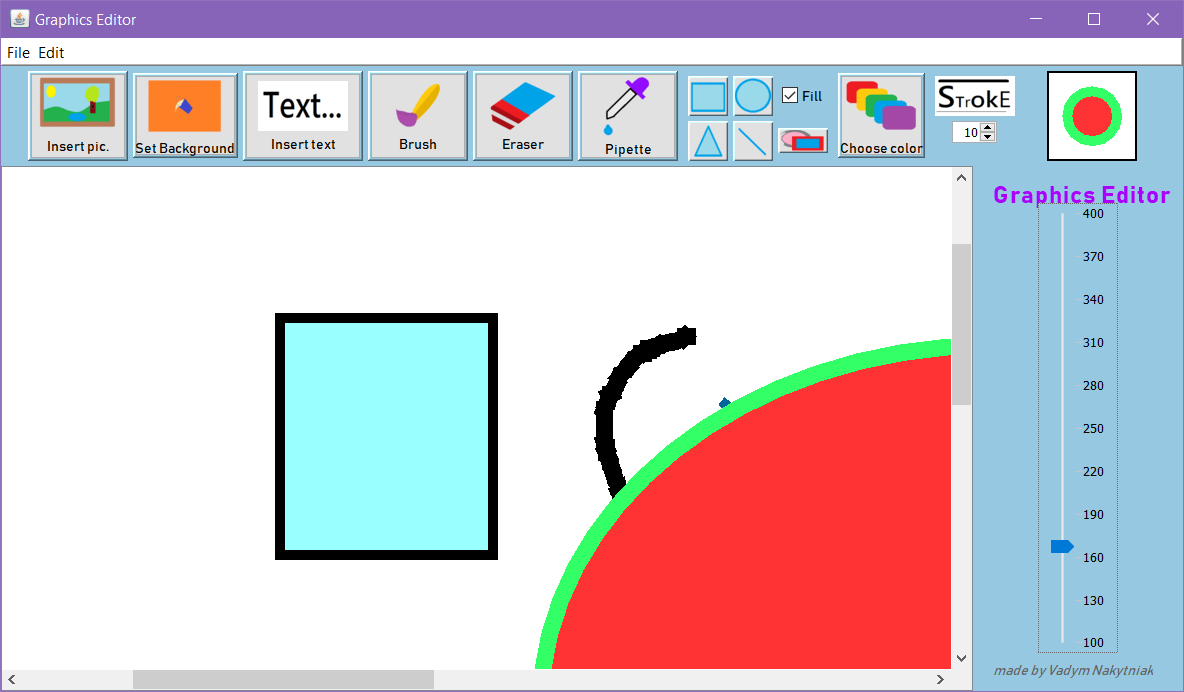


Рисунок 29. Зміна масштабу 1

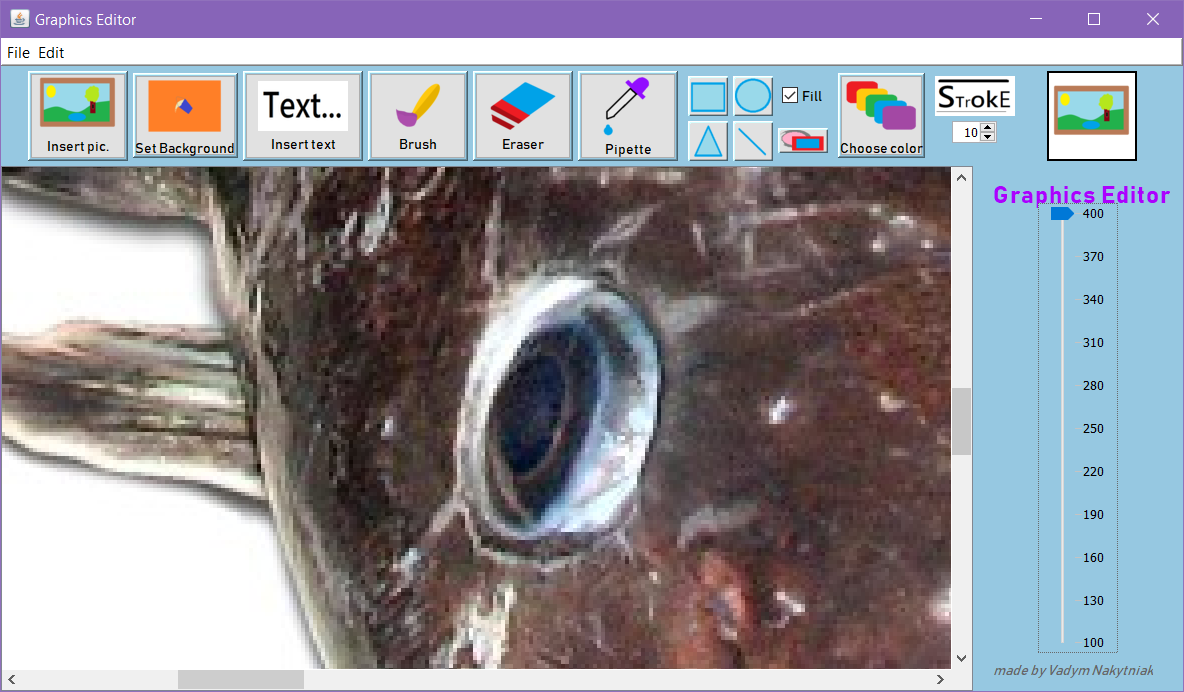


Рисунок 30. Зміна масштабу 2

А тепер перейдем до меню бару. Спочатку Edit. Тут маємо 4 опції:

кнопки – **Back** and **Forward**(а точніше Ctrl+Z і Ctrl + Y);

кнопки – **Cut Out** and **Insert**(Ctrl+X і Ctrl+V) – які надають змогу вирізати та вставляти з буферу обміну зображення

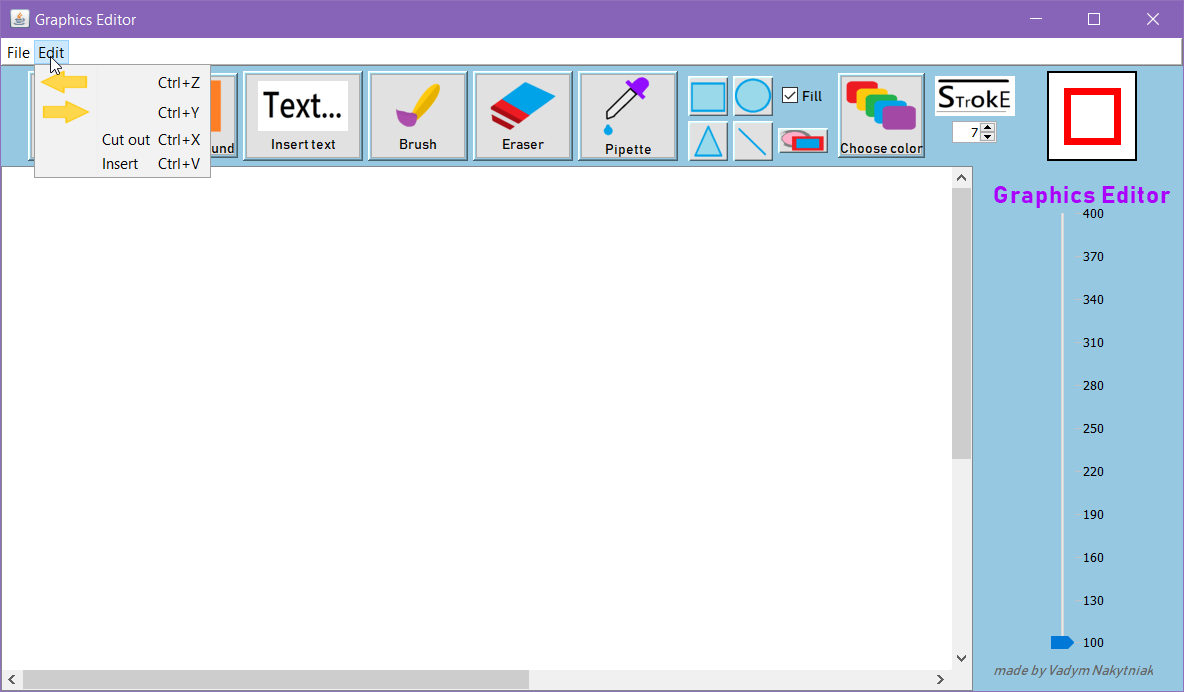


Рисунок 31. MenuBar Edit

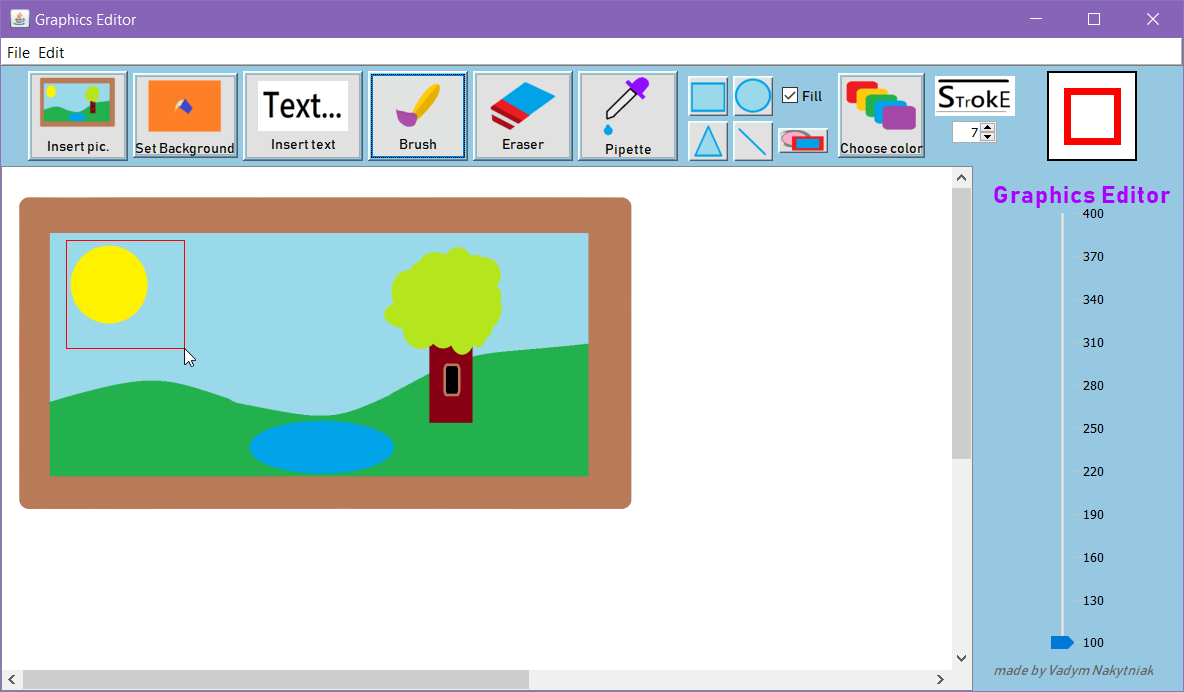


Рисунок 32. Виділення зони

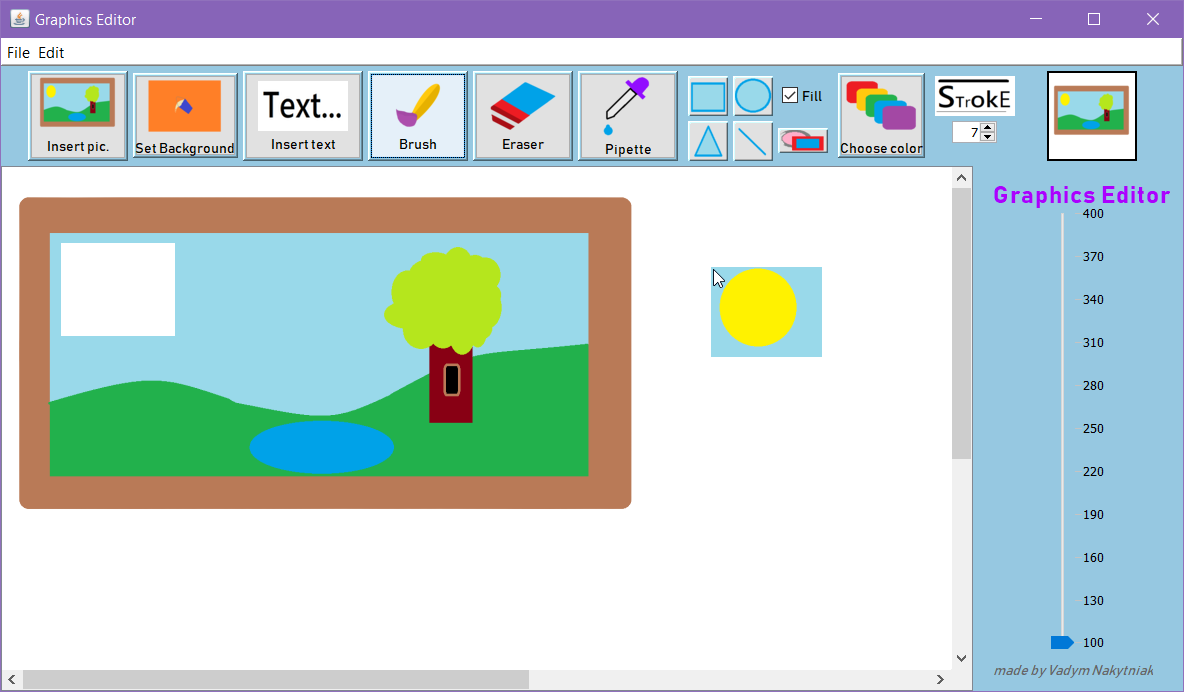


Рисунок 33. Вирізання та вставка

І File:

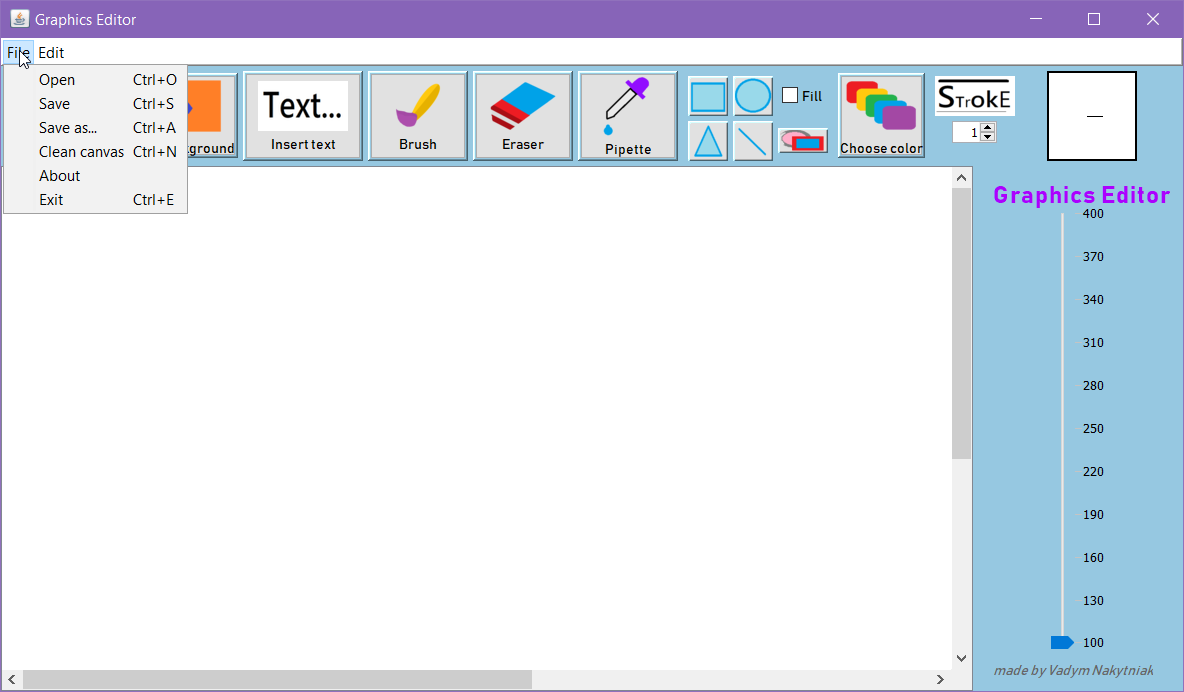


Рисунок 34. MenuBar File

Відповідно пункти:

**Open** – відкриває нове зображення полотна;

**Save** – зберігає зображення полотна в папку savedImages;

**Save as** – зберігає зображення полотна куди завгодно;

**Clean canvas** – чистить полотно;

**About** – коротка інформація про програму;

**Exit** – вихід.

Ось і все.

Висновок

**Враження та отриманий досвід:** Робота над другим завданням літнього практикуму дала мені неабиякий досвід. Перш за все, я зрозумів, що перд початком роботи необхідно мати план. Ти маєш точно знати що саме ти будеш програмувати, як ти пов’яжеш всі класи, як буде виглядати програма, зручність використання. Тому декілька годин в мене пішло лише на те, щоб зрозуміти, як саме я маю реалізувати Paint like програму.

**Труднощі:** Напевно найскладні було на початку, коли я придумував як реалізувати малювання. Було прийнято рішення, зробити власний тип класів на основі вже існуючих, що реалізуєть інтерфейс **Drawable**. Завдяки цьому я зміг додати всі об’єкти в колекцію, а потім звідти, за допомогою поліморфного посилання, кожен раз їх вимальовувати.

Також важко було продумати інтерфейс. Всі картинки, використані в програмі, намальовані мною.

**GitHub:** Як і в першому завданні, я використав систему керування версіями **Git**, а саме **GitHub**. Для користування **Git** я обрав графічний інтерфейс **Sourcetree**. В кінці звіту я залишив посилання на свій репозиторій.

Посилання

Репозиторій проекту на GitHub - <https://github.com/brutovsky/Graphics-Editor>

Лістинг коду

**GEFrame**

package graphicsEditor;

import static graphicsEditor.CanvasPanel.getScreenComponent;

import graphicsEditor.drawnShapes.Drawable;

import graphicsEditor.drawnShapes.DrawnImage;

import graphicsEditor.drawnShapes.DrawnRectangle;

import graphicsEditor.instruments.Tool;

import java.awt.BasicStroke;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Image;

import java.awt.Toolkit;

import java.awt.datatransfer.Clipboard;

import java.awt.datatransfer.DataFlavor;

import java.awt.datatransfer.UnsupportedFlavorException;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.imageio.ImageIO;

import javax.swing.JFileChooser;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import javax.swing.filechooser.FileFilter;

/\*\*

\* The main frame of the application

\*

\* @author VADYM NAKYTNIAK

\*/

public class GEFrame extends javax.swing.JFrame {

final private CanvasPanel canvas; // Canvas for drawing on

final private ToolbarPanel toolbar; // Panel with all tools

final private ScrollSidePanel scrollside; // Side panel with a scrollbar and some labels

private JScrollPane scrollCanvas; // ScrollPane for CanvasPanel to be scrollable

// Current location of a mouse on the frame

private int current\_x\_coordinate;

private int current\_y\_coordinate;

/\*\*

\* Creates new form GEFrame

\*/

public GEFrame() {

canvas = new CanvasPanel(this);

toolbar = new ToolbarPanel(this);

scrollside = new ScrollSidePanel(this);

scrollCanvas = new JScrollPane(canvas);

scrollCanvas.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

scrollCanvas.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

add(scrollCanvas, BorderLayout.CENTER);

add(toolbar, BorderLayout.PAGE\_START);

add(scrollside, BorderLayout.EAST);

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

menuBar = new javax.swing.JMenuBar();

fileMenu = new javax.swing.JMenu();

openItem = new javax.swing.JMenuItem();

saveItem = new javax.swing.JMenuItem();

saveAsItem = new javax.swing.JMenuItem();

newPictureItem = new javax.swing.JMenuItem();

aboutItem = new javax.swing.JMenuItem();

exitItem = new javax.swing.JMenuItem();

editMenu = new javax.swing.JMenu();

backItem = new javax.swing.JMenuItem();

forwardItem = new javax.swing.JMenuItem();

cutMenu = new javax.swing.JMenuItem();

insertItem = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Graphics Editor");

setBackground(new java.awt.Color(204, 243, 166));

setMinimumSize(new java.awt.Dimension(1200, 700));

addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

formMouseClicked(evt);

}

});

menuBar.setForeground(Color.red);

menuBar.setBackground(new java.awt.Color(255, 255, 255));

menuBar.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

fileMenu.setText("File");

openItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_O, java.awt.event.InputEvent.CTRL\_MASK));

openItem.setText("Open");

openItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

openItemActionPerformed(evt);

}

});

fileMenu.add(openItem);

saveItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_S, java.awt.event.InputEvent.CTRL\_MASK));

saveItem.setText("Save");

saveItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

saveItemActionPerformed(evt);

}

});

fileMenu.add(saveItem);

saveAsItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_A, java.awt.event.InputEvent.CTRL\_MASK));

saveAsItem.setText("Save as...");

saveAsItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

saveAsItemActionPerformed(evt);

}

});

fileMenu.add(saveAsItem);

newPictureItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_N, java.awt.event.InputEvent.CTRL\_MASK));

newPictureItem.setText("Clean canvas");

newPictureItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

newPictureItemActionPerformed(evt);

}

});

fileMenu.add(newPictureItem);

aboutItem.setText("About");

aboutItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

aboutItemActionPerformed(evt);

}

});

fileMenu.add(aboutItem);

exitItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_E, java.awt.event.InputEvent.CTRL\_MASK));

exitItem.setText("Exit");

exitItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

exitItemActionPerformed(evt);

}

});

fileMenu.add(exitItem);

menuBar.add(fileMenu);

editMenu.setText("Edit");

backItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_Z, java.awt.event.InputEvent.CTRL\_MASK));

backItem.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/back.png"))); // NOI18N

backItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

backItemActionPerformed(evt);

}

});

editMenu.add(backItem);

forwardItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_Y, java.awt.event.InputEvent.CTRL\_MASK));

forwardItem.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/forward.png"))); // NOI18N

forwardItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

forwardItemActionPerformed(evt);

}

});

editMenu.add(forwardItem);

cutMenu.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_X, java.awt.event.InputEvent.CTRL\_MASK));

cutMenu.setText("Cut out");

cutMenu.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

cutMenuActionPerformed(evt);

}

});

editMenu.add(cutMenu);

insertItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK\_V, java.awt.event.InputEvent.CTRL\_MASK));

insertItem.setText("Insert");

insertItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

insertItemActionPerformed(evt);

}

});

editMenu.add(insertItem);

menuBar.add(editMenu);

setJMenuBar(menuBar);

pack();

}// </editor-fold>

private void formMouseClicked(java.awt.event.MouseEvent evt) {

if (canvas.getTool() == Tool.PIPETTE) {

Color color = new Color(getScreenComponent(this).getRGB(evt.getX(), evt.getY()));

switch (canvas.getPipetteChoice()) {

case CanvasPanel.BRUSH\_COLOR: {

canvas.setBrushColor(color);

break;

}

case CanvasPanel.FILL\_COLOR: {

canvas.setFillColor(color);

break;

}

case CanvasPanel.BACKGROUND\_COLOR: {

DrawnRectangle background = new DrawnRectangle(canvas.getStroke(), color, color, 0, 0, CanvasPanel.WIDTH, CanvasPanel.HEIGHT);

canvas.setBackgroundRectangle(background);

break;

}

}

}

}

private void forwardItemActionPerformed(java.awt.event.ActionEvent evt) {

if (canvas.getBin().isEmpty()) {

return;

}

canvas.getShapes().add(canvas.getBin().pop());

canvas.repaint();

}

private void backItemActionPerformed(java.awt.event.ActionEvent evt) {

if (canvas.getShapes().size() == 1) {

return;

}

canvas.getBin().add(canvas.getShapes().get(canvas.getShapes().size() - 1));

canvas.getShapes().remove(canvas.getShapes().size() - 1);

canvas.repaint();

}

private void openItemActionPerformed(java.awt.event.ActionEvent evt) {

File file = new File(System.getProperty("user.dir"));

JFileChooser fc = new JFileChooser(file);

fc.setFileFilter(new FileFilter() {

@Override

public boolean accept(File f) {

String fileName = f.getName();

return fileName.endsWith(".png");

}

@Override

public String getDescription() {

return ".png";

}

});

int i = fc.showOpenDialog(null);

switch (i) {

case JFileChooser.APPROVE\_OPTION: {

try {

File fileSave = fc.getSelectedFile();

if (!fileSave.getName().endsWith(".png")) {

throw new IOException();

}

BufferedImage image = ImageIO.read(fileSave);

canvas.getShapes().add(new DrawnImage(image, 0, 0));

canvas.repaint();

} catch (IOException ioe) {

JOptionPane.showMessageDialog(null, "Invalid file", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

private void saveItemActionPerformed(java.awt.event.ActionEvent evt) {

try {

File folder = new File(System.getProperty("user.dir") + "//savedPictures");

if (!folder.exists()) {

folder.mkdir();

}

String name = JOptionPane.showInputDialog(null, "Enter the name of the file", "Picture");

if (name == null) {

return;

}

File fileSave = new File(System.getProperty("user.dir") + "//savedPictures//" + name + ".png");

if (fileSave.exists()) {

int res = JOptionPane.showConfirmDialog(null, "Are you sure you want to overwrite " + fileSave.getName() + "?", "Overwrite ?", JOptionPane.YES\_NO\_CANCEL\_OPTION);

if (res == JOptionPane.YES\_OPTION) {

fileSave.createNewFile();

} else {

return;

}

} else {

fileSave.createNewFile();

}

ImageIO.write(CanvasPanel.getScreenComponent(canvas), "png", fileSave);

} catch (IOException ioe) {

JOptionPane.showMessageDialog(null, "Invalid file", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

private void saveAsItemActionPerformed(java.awt.event.ActionEvent evt) {

File file = new File(System.getProperty("user.dir"));

JFileChooser fc = new JFileChooser(file);

int i = fc.showSaveDialog(null);

switch (i) {

case JFileChooser.APPROVE\_OPTION: {

try {

File fileSave = new File(fc.getSelectedFile() + ".png");

if (fileSave.exists()) {

int res = JOptionPane.showConfirmDialog(null, "Are you sure you want to rewrite the file " + file.getName() + "?", "Rewrite ?", JOptionPane.YES\_NO\_CANCEL\_OPTION);

if (res == JOptionPane.YES\_OPTION) {

} else {

return;

}

}

fileSave.createNewFile();

ImageIO.write(CanvasPanel.getScreenComponent(canvas), "png", fileSave);

} catch (IOException ioe) {

JOptionPane.showMessageDialog(null, "Invalid file", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

private void newPictureItemActionPerformed(java.awt.event.ActionEvent evt) {

int res = JOptionPane.showConfirmDialog(null, "Are you sure you want to clean the canvas ?", "Clean?", JOptionPane.YES\_NO\_CANCEL\_OPTION);

if (res == JOptionPane.YES\_OPTION) {

ArrayList<Drawable> list = new ArrayList<>();

canvas.setShapes(list);

list.add(canvas.getBackgroundRectangle());

canvas.repaint();

}

}

private void aboutItemActionPerformed(java.awt.event.ActionEvent evt) {

JOptionPane.showMessageDialog(null, "Graphics Editor\n\nmade by Vadym Nakytniak in 2019", "About", JOptionPane.DEFAULT\_OPTION);

}

private void exitItemActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

private void insertItemActionPerformed(java.awt.event.ActionEvent evt) {

canvas.reset();

Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();

if (clipboard.isDataFlavorAvailable(DataFlavor.imageFlavor)) {

try {

BufferedImage image = (BufferedImage) clipboard.getData(DataFlavor.imageFlavor);

canvas.setImage(image, 0, 0);

canvas.repaint();

canvas.setTool(Tool.INSERT\_PICTURE);

canvas.setIsMouseOnImage(true);

toolbar.updatePreview();

} catch (UnsupportedFlavorException | IOException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage(), "ERROR", JOptionPane.ERROR\_MESSAGE);

}

} else {

JOptionPane.showMessageDialog(null, "NO IMAGES", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

private void cutMenuActionPerformed(java.awt.event.ActionEvent evt) {

canvas.reset();

canvas.setTool(Tool.CUT\_OUT);

toolbar.updatePreview();

/\*BufferedImage image;

Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();

BufferedImage img = new BufferedImage(100, 100, BufferedImage.TYPE\_INT\_ARGB);

CanvasPanel.getScreenComponent(canvas).get;

ImageIO.write(, "png", fileSave);

\*/

}

public void setCurrentCoordinates() {

current\_x\_coordinate = (int) this.getMousePosition().getX();

current\_y\_coordinate = (int) this.getMousePosition().getX();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

try {

// Set System L&F

UIManager.setLookAndFeel(

UIManager.getSystemLookAndFeelClassName());

} catch (UnsupportedLookAndFeelException | ClassNotFoundException | InstantiationException | IllegalAccessException e) {

}

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(() -> {

new GEFrame().setVisible(true);

});

}

public CanvasPanel getCanvas() {

return canvas;

}

public ToolbarPanel getToolbar() {

return toolbar;

}

public ScrollSidePanel getScrollside() {

return scrollside;

}

public JMenu getFileMenu() {

return fileMenu;

}

public JMenuBar getjMenuBar() {

return menuBar;

}

public int getCurrent\_x\_coordinate() {

return current\_x\_coordinate;

}

public int getCurrent\_y\_coordinate() {

return current\_y\_coordinate;

}

public JScrollPane getScrollCanvas() {

return scrollCanvas;

}

public void setCurrent\_x\_coordinate(int current\_x\_coordinate) {

this.current\_x\_coordinate = current\_x\_coordinate;

}

public void setCurrent\_y\_coordinate(int current\_y\_coordinate) {

this.current\_y\_coordinate = current\_y\_coordinate;

}

public void setScrollCanvas(JScrollPane scrollCanvas) {

this.scrollCanvas = scrollCanvas;

}

// Variables declaration - do not modify

private javax.swing.JMenuItem aboutItem;

private javax.swing.JMenuItem backItem;

private javax.swing.JMenuItem cutMenu;

private javax.swing.JMenu editMenu;

private javax.swing.JMenuItem exitItem;

private javax.swing.JMenu fileMenu;

private javax.swing.JMenuItem forwardItem;

private javax.swing.JMenuItem insertItem;

private javax.swing.JMenuBar menuBar;

private javax.swing.JMenuItem newPictureItem;

private javax.swing.JMenuItem openItem;

private javax.swing.JMenuItem saveAsItem;

private javax.swing.JMenuItem saveItem;

// End of variables declaration

}

**CanvasPanel**

package graphicsEditor;

import graphicsEditor.instruments.ImageTransfarable;

import graphicsEditor.instruments.Tool;

import graphicsEditor.drawnShapes.DrawnRectangle;

import graphicsEditor.drawnShapes.DrawnTriangle;

import graphicsEditor.drawnShapes.Drawable;

import graphicsEditor.drawnShapes.DrawnCircle;

import graphicsEditor.drawnShapes.DrawnImage;

import graphicsEditor.drawnShapes.DrawnLine;

import graphicsEditor.drawnShapes.DrawnText;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Component;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.RenderingHints;

import java.awt.Shape;

import java.awt.Stroke;

import java.awt.Toolkit;

import java.awt.datatransfer.Clipboard;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

import java.util.Stack;

import javax.swing.JPanel;

/\*\*

\* Panel for drwing on

\*

\* @author VADYM NAKYTNIAK

\*/

public class CanvasPanel extends JPanel {

//Rectangle zone

DrawnRectangle zone;

//GEFrame

GEFrame frame;

//Preferred size initially

public static final int PREFERRED\_WIDTH = 1700;

public static final int PREFERRED\_HEIGHT = 850;

//Constants

public static final int WIDTH = 1725;

public static final int HEIGHT = 875;

//Color aim for PIPETTE

public static final int BRUSH\_COLOR = 1;

public static final int FILL\_COLOR = 2;

public static final int BACKGROUND\_COLOR = 3;

//PIPETTE choice

private int pipetteChoice;

//Booleans

private boolean isMouseClicked;

private boolean isMouseOnImage;

private boolean isMouseOnText;

private boolean isMouseOnZone;

//The list of all shapes to be drawn

private ArrayList<Drawable> shapes;

//The stack of the shapes in the bin

private Stack<Drawable> bin;

//Coordinates

private float x1;

private float y1;

private float x2;

private float y2;

//Drawn shape

private Drawable s;

private DrawnImage image;

private DrawnText text;

//Background rectangle

private DrawnRectangle background;

//Colors

private Color brushColor;

private Color backgroundColor;

private Color fillColor;

//Stroke

private Stroke stroke;

//Tool

private Tool tool;

//Scale

private float scale;

//Varibles initialization

{

tool = Tool.BRUSH;

stroke = new BasicStroke(1f);

scale = 1f;

brushColor = Color.BLACK;

backgroundColor = Color.WHITE;

fillColor = null;

background = new DrawnRectangle(stroke, backgroundColor, backgroundColor, 0, 0, WIDTH, HEIGHT);

shapes = new ArrayList();

shapes.add(background);

bin = new Stack<>();

}

/\*\*

\* Creates new form CanvasPanel

\*

\* @param frame

\*/

public CanvasPanel(GEFrame frame) {

initComponents();

this.frame = frame;

setSize(PREFERRED\_WIDTH, PREFERRED\_HEIGHT);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

setBackground(new java.awt.Color(255, 255, 255));

setMaximumSize(new java.awt.Dimension(1690, 850));

setMinimumSize(new java.awt.Dimension(1690, 850));

setOpaque(false);

setPreferredSize(new java.awt.Dimension(1690, 850));

addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {

public void mouseDragged(java.awt.event.MouseEvent evt) {

formMouseDragged(evt);

}

});

addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

formMousePressed(evt);

}

public void mouseReleased(java.awt.event.MouseEvent evt) {

formMouseReleased(evt);

}

});

}// </editor-fold>

private void formMouseDragged(java.awt.event.MouseEvent evt) {

if (isMouseClicked) {

x2 = evt.getX() / scale;

y2 = evt.getY() / scale;

switch (tool) {

case INSERT\_PICTURE: {

image = new DrawnImage(image.getImage(), (int) x1, (int) y1);

break;

}

case INSERT\_TEXT: {

setText(new DrawnText(text.getFont(), text.getText(), text.getColor(), (int) x1, (int) y1));

break;

}

case BRUSH: {

shapes.add(new DrawnLine(stroke, brushColor, x1, y1, x2, y2));

break;

}

case ERASER: {

shapes.add(new DrawnLine(stroke, backgroundColor, x1, y1, x2, y2));

break;

}

case PIPETTE: {

break;

}

case SHAPE\_RECTANGLE: {

s = new DrawnRectangle(stroke, brushColor, fillColor, (int) x1, (int) y1, (int) x2 - (int) x1, (int) y2 - (int) y1);

repaint();

return;

}

case SHAPE\_CIRCLE: {

s = new DrawnCircle(stroke, brushColor, fillColor, (int) x1, (int) y1, (int) x2 - (int) x1, (int) y2 - (int) y1);

repaint();

return;

}

case SHAPE\_TRIANGLE: {

s = new DrawnTriangle(stroke, brushColor, fillColor, (int) x1, (int) y1, (int) x2, (int) y2, (int) x2, (int) y1);

repaint();

return;

}

case SHAPE\_LINE: {

s = new DrawnLine(stroke, brushColor, (double) x1, (double) y1, (double) x2, (double) y2);

repaint();

return;

}

case NO\_TOOL: {

break;

}

case CUT\_OUT: {

zone = new DrawnRectangle(zone.getStroke(), zone.getColor(), zone.getFillColor(), (int) x1, (int) y1, (int) x2 - (int) x1, (int) y2 - (int) y1);

repaint();

return;

}

}

repaint();

x1 = x2;

y1 = y2;

}

}

private void formMousePressed(java.awt.event.MouseEvent evt) {

isMouseClicked = true;

x1 = evt.getX() / scale;

y1 = evt.getY() / scale;

switch (tool) {

case INSERT\_PICTURE: {

isMouseOnImage = true;

image = new DrawnImage(image.getImage(), (int) x1, (int) y1);

break;

}

case INSERT\_TEXT: {

isMouseOnText = true;

setText(new DrawnText(text.getFont(), text.getText(), text.getColor(), (int) x1, (int) y1));

break;

}

case BRUSH: {

shapes.add(new DrawnLine(stroke, brushColor, x1, y1, x1, y1));

break;

}

case ERASER: {

shapes.add(new DrawnLine(stroke, backgroundColor, x1, y1, x1, y1));

break;

}

case PIPETTE: {

if (getTool() == Tool.PIPETTE) {

Color color = new Color(getScreenComponent(this).getRGB(evt.getX(), evt.getY()));

switch (getPipetteChoice()) {

case CanvasPanel.BRUSH\_COLOR: {

setBrushColor(color);

break;

}

case CanvasPanel.FILL\_COLOR: {

setFillColor(color);

break;

}

case CanvasPanel.BACKGROUND\_COLOR: {

background = new DrawnRectangle(frame.getCanvas().getStroke(), color, color, 0, 0, CanvasPanel.WIDTH, CanvasPanel.HEIGHT);

setBackgroundRectangle(background);

break;

}

}

}

break;

}

case SHAPE\_RECTANGLE: {

s = new DrawnRectangle(stroke, brushColor, fillColor, (int) x1, (int) y1, 5, 5);

break;

}

case SHAPE\_CIRCLE: {

s = new DrawnCircle(stroke, brushColor, fillColor, (int) x1, (int) y1, 5, 5);

break;

}

case SHAPE\_TRIANGLE: {

s = new DrawnTriangle(stroke, brushColor, fillColor, (int) x1, (int) y1, (int) x1 + 5, (int) y1 - 5, (int) x1 + 10, (int) y1);

break;

}

case SHAPE\_LINE: {

s = new DrawnLine(stroke, brushColor, (double) x1, (double) y1, (double) x1, (double) y1);

break;

}

case NO\_TOOL: {

break;

}

case CUT\_OUT: {

isMouseOnZone = true;

zone = new DrawnRectangle(new BasicStroke(1f), Color.RED, null, (int) x1, (int) y1, 50, 50);

break;

}

}

repaint();

}

private void formMouseReleased(java.awt.event.MouseEvent evt) {

isMouseClicked = false;

reset();

repaint();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);//To change body of generated methods, choose Tools | Templates.

Graphics2D g2d = (Graphics2D) g;

g2d.scale(scale, scale);

g2d.setColor(brushColor);

g2d.setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING, RenderingHints.VALUE\_TEXT\_ANTIALIAS\_ON);

//DRAWING

drawAllShapes(g2d);

//Draw Shape s

drawShape(g2d);

//Draw Image image

drawImage(g2d);

//Draw String text

drawText(g2d);

//Draw zone

drawZone(g2d);

}

/\*\*

\* Method for drawing all objects from the arraylist shapes

\*

\* @param g2d

\*/

public void drawAllShapes(Graphics2D g2d) {

for (Drawable s : shapes) {

if (s instanceof DrawnImage) {

DrawnImage im = (DrawnImage) s;

g2d.drawImage((BufferedImage) im.getImage(), im.getX(), im.getY(), null);

continue;

}

if (s instanceof DrawnText) {

DrawnText t = (DrawnText) s;

Font current\_font = g2d.getFont();

g2d.setFont(t.getFont());

g2d.setColor(t.getColor());

g2d.drawString(t.getText(), t.getX(), t.getY());

g2d.setColor(s.getColor());

g2d.setFont(current\_font);

continue;

}

if (s.getFillColor() != null) {

g2d.setColor(s.getFillColor());

g2d.fill((Shape) s);

}

g2d.setStroke(s.getStroke());

g2d.setColor(s.getColor());

g2d.draw((Shape) s);

}

}

public void drawZone(Graphics2D g2d) {

if (zone != null) {

g2d.setStroke(zone.getStroke());

g2d.setColor(zone.getColor());

g2d.draw((Shape) zone);

}

}

/\*\*

\* Method for drawing a temporary shape

\*

\* @param g2d

\*/

public void drawShape(Graphics2D g2d) {

if (s != null) {

if (s.getFillColor() != null) {

g2d.setColor(s.getFillColor());

g2d.fill((Shape) s);

}

g2d.setStroke(s.getStroke());

g2d.setColor(s.getColor());

g2d.draw((Shape) s);

}

}

/\*\*

\* Method for drawing a temporary image

\*

\* @param g2d

\*/

public void drawImage(Graphics g2d) {

if (image != null) {

g2d.drawImage(image.getImage(), image.getX(), image.getY(), null);

}

}

/\*\*

\* Method for drawing temporary text

\*

\* @param g2d

\*/

public void drawText(Graphics g2d) {

if (text != null) {

g2d.setFont(text.getFont());

g2d.setColor(text.getColor());

g2d.drawString(text.getText(), text.getX(), text.getY());

}

}

/\*\*

\* Method to reset the temporary variables

\*/

public void reset() {

if (isMouseOnZone) {

isMouseOnZone = false;

Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();

BufferedImage canvas\_image = getScreenComponent(this).getSubimage((int) ((zone.getX()+1)), (int) ((zone.getY()+1)), (int) ((zone.getWidth()-1)), (int) (((zone.getHeight()-1))));

ImageTransfarable cutout\_image = new ImageTransfarable(canvas\_image);

clipboard.setContents(cutout\_image, null);

shapes.add(new DrawnRectangle(new BasicStroke(1f), backgroundColor, backgroundColor,(int) zone.getX(), (int) zone.getY(), (int) zone.getWidth()+1, (int) zone.getHeight()+1));

zone = null;

}

if (isMouseOnImage) {

isMouseOnImage = false;

setTool(Tool.NO\_TOOL);

shapes.add(image);

image = null;

}

if (isMouseOnText) {

isMouseOnText = false;

setTool(Tool.NO\_TOOL);

shapes.add(text);

text = null;

}

if (s != null) {

shapes.add(s);

s = null;

}

}

/\*\*

\* Method for taking the picture of the panel

\*

\* @param component

\* @return

\*/

public static BufferedImage getScreenComponent(Component component) {

BufferedImage image = new BufferedImage(

component.getWidth(),

component.getHeight(),

BufferedImage.TYPE\_INT\_RGB

);

component.paint(image.getGraphics());

return image;

}

public boolean isIsMouseClicked() {

return isMouseClicked;

}

public ArrayList<Drawable> getShapes() {

return shapes;

}

public float getX1() {

return x1;

}

public float getY1() {

return y1;

}

public float getX2() {

return x2;

}

public float getY2() {

return y2;

}

public Drawable getS() {

return s;

}

public DrawnRectangle getBackgroundRectangle() {

return background;

}

public Color getBrushColor() {

return brushColor;

}

public Color getBackgroundColor() {

return backgroundColor;

}

public Color getFillColor() {

return fillColor;

}

public Stroke getStroke() {

return stroke;

}

public Tool getTool() {

return tool;

}

public float getScale() {

return scale;

}

public int getPipetteChoice() {

return pipetteChoice;

}

public DrawnImage getImage() {

return image;

}

public DrawnText getText() {

return text;

}

public Stack<Drawable> getBin() {

return bin;

}

public boolean isIsMouseOnImage() {

return isMouseOnImage;

}

public boolean isIsMouseOnText() {

return isMouseOnText;

}

public void setIsMouseClicked(boolean isMouseClicked) {

this.isMouseClicked = isMouseClicked;

}

public void setShapes(ArrayList<Drawable> shapes) {

this.shapes = shapes;

}

public void setX1(int x1) {

this.x1 = x1;

}

public void setY1(int y1) {

this.y1 = y1;

}

public void setX2(int x2) {

this.x2 = x2;

}

public void setY2(int y2) {

this.y2 = y2;

}

public void setS(Drawable s) {

this.s = s;

}

public void setBackgroundRectangle(DrawnRectangle background) {

setBackgroundColor(background.getColor());

getShapes().set(0, background);

this.background = background;

repaint();

}

public void setBrushColor(Color brushColor) {

this.brushColor = brushColor;

}

public void setBackgroundColor(Color backgroundColor) {

this.backgroundColor = backgroundColor;

}

public void setFillColor(Color fillColor) {

this.fillColor = fillColor;

}

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

public void setTool(Tool tool) {

this.tool = tool;

}

public void setScale(float scale) {

this.scale = scale;

}

public void setPipetteChoice(int pipetteChoice) {

this.pipetteChoice = pipetteChoice;

}

public void setImage(BufferedImage image, int x, int y) {

this.image = new DrawnImage(image, x, y);

}

public void setText(DrawnText text) {

this.text = text;

}

public void setIsMouseOnImage(boolean isMouseOnImage) {

this.isMouseOnImage = isMouseOnImage;

}

public void setIsMouseOnText(boolean isMouseOnText) {

this.isMouseOnText = isMouseOnText;

}

public void setBin(Stack<Drawable> bin) {

this.bin = bin;

}

public DrawnRectangle getZone() {

return zone;

}

public void setZone(DrawnRectangle zone) {

this.zone = zone;

}

public boolean isIsMouseOnZone() {

return isMouseOnZone;

}

public void setIsMouseOnZone(boolean isMouseOnZone) {

this.isMouseOnZone = isMouseOnZone;

}

// Variables declaration - do not modify

// End of variables declaration

}

**ScrollSidePanel**

package graphicsEditor;

import static graphicsEditor.CanvasPanel.getScreenComponent;

import graphicsEditor.drawnShapes.DrawnRectangle;

import graphicsEditor.instruments.Tool;

import java.awt.Color;

import java.awt.Dimension;

/\*\* Side panel for scroll

\*

\* @author VADIM

\*/

public class ScrollSidePanel extends javax.swing.JPanel {

final private GEFrame frame;

/\*\*

\* Creates new form ScrollSidePanel

\* @param frame

\*/

public ScrollSidePanel(GEFrame frame) {

initComponents();

setBackground(new Color(150, 200, 225));

this.frame = frame;

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

slider = new javax.swing.JSlider();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

formMouseClicked(evt);

}

});

slider.setBackground(new java.awt.Color(150, 200, 225));

slider.setMajorTickSpacing(30);

slider.setMaximum(400);

slider.setMinimum(100);

slider.setMinorTickSpacing(30);

slider.setOrientation(javax.swing.JSlider.VERTICAL);

slider.setPaintLabels(true);

slider.setPaintTicks(true);

slider.setPreferredSize(new java.awt.Dimension(80, 450));

slider.addChangeListener(new javax.swing.event.ChangeListener() {

public void stateChanged(javax.swing.event.ChangeEvent evt) {

sliderStateChanged(evt);

}

});

slider.addMouseWheelListener(new java.awt.event.MouseWheelListener() {

public void mouseWheelMoved(java.awt.event.MouseWheelEvent evt) {

sliderMouseWheelMoved(evt);

}

});

jLabel1.setFont(new java.awt.Font("Bahnschrift", 2, 14)); // NOI18N

jLabel1.setForeground(new java.awt.Color(102, 102, 102));

jLabel1.setText("made by Vadym Nakytniak");

jLabel2.setFont(new java.awt.Font("Bahnschrift", 1, 24)); // NOI18N

jLabel2.setForeground(new java.awt.Color(172, 0, 255));

jLabel2.setText("Graphics Editor");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

this.setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(20, 20, 20)

.addComponent(jLabel1))

.addGroup(layout.createSequentialGroup()

.addGap(20, 20, 20)

.addComponent(jLabel2)))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addComponent(slider, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 635, Short.MAX\_VALUE)

.addComponent(jLabel1)

.addContainerGap())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addComponent(slider, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

);

}// </editor-fold>

private void formMouseClicked(java.awt.event.MouseEvent evt) {

if (frame.getCanvas().getTool() == Tool.PIPETTE) {

Color color = new Color(getScreenComponent(this).getRGB(evt.getX(), evt.getY()));

switch (frame.getCanvas().getPipetteChoice()) {

case CanvasPanel.BRUSH\_COLOR: {

frame.getCanvas().setBrushColor(color);

break;

}

case CanvasPanel.FILL\_COLOR: {

frame.getCanvas().setFillColor(color);

break;

}

case CanvasPanel.BACKGROUND\_COLOR: {

DrawnRectangle background = new DrawnRectangle(frame.getCanvas().getStroke(), color, color, 0, 0, CanvasPanel.WIDTH, CanvasPanel.HEIGHT);

frame.getCanvas().setBackgroundRectangle(background);

break;

}

}

}

}

private void sliderStateChanged(javax.swing.event.ChangeEvent evt) {

frame.getCanvas().setScale(slider.getValue() / 100f);

Dimension dim = new Dimension((int) (CanvasPanel.PREFERRED\_WIDTH \* frame.getCanvas().getScale()), (int) (CanvasPanel.PREFERRED\_HEIGHT \* frame.getCanvas().getScale()));

frame.getCanvas().setPreferredSize(dim);

frame.getCanvas().repaint();

frame.getCanvas().revalidate();

}

private void sliderMouseWheelMoved(java.awt.event.MouseWheelEvent evt) {

slider.setValue(slider.getValue() + (int) evt.getPreciseWheelRotation());

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JSlider slider;

// End of variables declaration

}

**ToolbarPanel**

package graphicsEditor;

import static graphicsEditor.CanvasPanel.getScreenComponent;

import graphicsEditor.drawnShapes.DrawnRectangle;

import graphicsEditor.drawnShapes.DrawnText;

import graphicsEditor.instruments.Tool;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.ButtonGroup;

import javax.swing.JColorChooser;

import javax.swing.JComboBox;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JRadioButton;

import javax.swing.JSpinner;

import javax.swing.JTextField;

import javax.swing.filechooser.FileFilter;

/\*\*Panel for the toolbar

\*

\* @author VADYM NAKYTNIAK

\*/

public class ToolbarPanel extends javax.swing.JPanel {

final private GEFrame frame;

/\*\*

\* Creates new form ToolbarPanel

\* @param frame

\*/

public ToolbarPanel(GEFrame frame) {

setBackground(new Color(150, 200, 225));

this.frame = frame;

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

java.awt.GridBagConstraints gridBagConstraints;

insertPictureButton = new javax.swing.JButton();

setBackgroundButton = new javax.swing.JButton();

insertTextButton = new javax.swing.JButton();

brushButton = new javax.swing.JButton();

eraserButton = new javax.swing.JButton();

pipetteButton = new javax.swing.JButton();

shapesPanel = new javax.swing.JPanel();

rectangleButton = new javax.swing.JButton();

circleButton = new javax.swing.JButton();

fillCheckBox = new javax.swing.JCheckBox();

triangleButton = new javax.swing.JButton();

lineButton = new javax.swing.JButton();

fillColorButton = new javax.swing.JButton();

chooseColorButton = new javax.swing.JButton();

strokePanel = new javax.swing.JPanel();

strokeLabel = new javax.swing.JLabel();

strokeSpinner = new javax.swing.JSpinner();

previewPanel = new javax.swing.JPanel();

setForeground(new java.awt.Color(0, 150, 200));

setMaximumSize(new java.awt.Dimension(1200, 100));

setMinimumSize(new java.awt.Dimension(1200, 100));

setPreferredSize(new java.awt.Dimension(1200, 100));

addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

formMouseClicked(evt);

}

});

setLayout(new java.awt.GridBagLayout());

insertPictureButton.setBackground(new java.awt.Color(150, 200, 225));

insertPictureButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

insertPictureButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/insertPicture.png"))); // NOI18N

insertPictureButton.setText("Insert pic.");

insertPictureButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

insertPictureButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

insertPictureButton.setPreferredSize(new java.awt.Dimension(100, 90));

insertPictureButton.setVerticalAlignment(javax.swing.SwingConstants.TOP);

insertPictureButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

insertPictureButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

insertPictureButtonMouseReleased(evt);

}

});

gridBagConstraints = new java.awt.GridBagConstraints();

gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;

gridBagConstraints.weightx = 5.0;

add(insertPictureButton, gridBagConstraints);

setBackgroundButton.setBackground(new java.awt.Color(150, 200, 225));

setBackgroundButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

setBackgroundButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/background.png"))); // NOI18N

setBackgroundButton.setText("Set Background");

setBackgroundButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

setBackgroundButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

setBackgroundButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

setBackgroundButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

setBackgroundButtonMouseClicked(evt);

}

});

gridBagConstraints = new java.awt.GridBagConstraints();

gridBagConstraints.insets = new java.awt.Insets(0, 5, 0, 5);

add(setBackgroundButton, gridBagConstraints);

insertTextButton.setBackground(new java.awt.Color(150, 200, 225));

insertTextButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

insertTextButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/insertText.png"))); // NOI18N

insertTextButton.setText("Insert text");

insertTextButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

insertTextButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

insertTextButton.setMaximumSize(new java.awt.Dimension(120, 90));

insertTextButton.setMinimumSize(new java.awt.Dimension(120, 90));

insertTextButton.setPreferredSize(new java.awt.Dimension(120, 90));

insertTextButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

insertTextButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

insertTextButtonMouseReleased(evt);

}

});

add(insertTextButton, new java.awt.GridBagConstraints());

brushButton.setBackground(Color.ORANGE);

brushButton.setBackground(new java.awt.Color(150, 200, 225));

brushButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

brushButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/brush.png"))); // NOI18N

brushButton.setText("Brush");

brushButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

brushButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

brushButton.setMaximumSize(new java.awt.Dimension(100, 90));

brushButton.setMinimumSize(new java.awt.Dimension(100, 90));

brushButton.setPreferredSize(new java.awt.Dimension(100, 90));

brushButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

brushButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

brushButtonMouseReleased(evt);

}

});

gridBagConstraints = new java.awt.GridBagConstraints();

gridBagConstraints.insets = new java.awt.Insets(0, 5, 0, 5);

add(brushButton, gridBagConstraints);

eraserButton.setBackground(new java.awt.Color(150, 200, 225));

eraserButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

eraserButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/eraser.png"))); // NOI18N

eraserButton.setText("Eraser");

eraserButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

eraserButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

eraserButton.setPreferredSize(new java.awt.Dimension(100, 90));

eraserButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

eraserButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

eraserButtonMouseReleased(evt);

}

});

add(eraserButton, new java.awt.GridBagConstraints());

pipetteButton.setBackground(new java.awt.Color(150, 200, 225));

pipetteButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

pipetteButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/pipette.png"))); // NOI18N

pipetteButton.setText("Pipette");

pipetteButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

pipetteButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

pipetteButton.setMaximumSize(new java.awt.Dimension(100, 90));

pipetteButton.setMinimumSize(new java.awt.Dimension(100, 90));

pipetteButton.setPreferredSize(new java.awt.Dimension(100, 90));

pipetteButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

pipetteButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

pipetteButtonMouseReleased(evt);

}

});

gridBagConstraints = new java.awt.GridBagConstraints();

gridBagConstraints.insets = new java.awt.Insets(0, 5, 0, 5);

add(pipetteButton, gridBagConstraints);

shapesPanel.setBackground(new Color(150, 200, 225));

shapesPanel.setMaximumSize(new java.awt.Dimension(150, 90));

shapesPanel.setMinimumSize(new java.awt.Dimension(150, 90));

shapesPanel.setPreferredSize(new java.awt.Dimension(150, 90));

rectangleButton.setBackground(new java.awt.Color(150, 200, 225));

rectangleButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/rectangle.png"))); // NOI18N

rectangleButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

rectangleButton.setMaximumSize(new java.awt.Dimension(40, 40));

rectangleButton.setMinimumSize(new java.awt.Dimension(40, 40));

rectangleButton.setPreferredSize(new java.awt.Dimension(40, 40));

rectangleButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

rectangleButtonMouseReleased(evt);

}

});

shapesPanel.add(rectangleButton);

circleButton.setBackground(new java.awt.Color(150, 200, 225));

circleButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/circle.png"))); // NOI18N

circleButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

circleButton.setMaximumSize(new java.awt.Dimension(40, 40));

circleButton.setMinimumSize(new java.awt.Dimension(40, 40));

circleButton.setPreferredSize(new java.awt.Dimension(40, 40));

circleButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

circleButtonMouseReleased(evt);

}

});

shapesPanel.add(circleButton);

fillCheckBox.setBackground(new Color(150, 200, 225));

fillCheckBox.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

fillCheckBox.setText("Fill");

fillCheckBox.setMaximumSize(new java.awt.Dimension(50, 25));

fillCheckBox.setMinimumSize(new java.awt.Dimension(50, 25));

fillCheckBox.setPreferredSize(new java.awt.Dimension(50, 25));

fillCheckBox.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

fillCheckBoxMouseReleased(evt);

}

});

shapesPanel.add(fillCheckBox);

triangleButton.setBackground(new java.awt.Color(150, 200, 225));

triangleButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/triangle.png"))); // NOI18N

triangleButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

triangleButton.setMaximumSize(new java.awt.Dimension(40, 40));

triangleButton.setMinimumSize(new java.awt.Dimension(40, 40));

triangleButton.setPreferredSize(new java.awt.Dimension(40, 40));

triangleButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

triangleButtonMouseReleased(evt);

}

});

shapesPanel.add(triangleButton);

lineButton.setBackground(new java.awt.Color(150, 200, 225));

lineButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/line.png"))); // NOI18N

lineButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

lineButton.setMaximumSize(new java.awt.Dimension(40, 40));

lineButton.setMinimumSize(new java.awt.Dimension(40, 40));

lineButton.setPreferredSize(new java.awt.Dimension(40, 40));

lineButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

lineButtonMouseReleased(evt);

}

});

shapesPanel.add(lineButton);

fillColorButton.setBackground(new java.awt.Color(150, 200, 225));

fillColorButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/fillColor.png"))); // NOI18N

fillColorButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

fillColorButton.setMaximumSize(new java.awt.Dimension(50, 25));

fillColorButton.setMinimumSize(new java.awt.Dimension(50, 25));

fillColorButton.setPreferredSize(new java.awt.Dimension(50, 25));

fillColorButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

fillColorButtonMouseReleased(evt);

}

});

shapesPanel.add(fillColorButton);

add(shapesPanel, new java.awt.GridBagConstraints());

chooseColorButton.setBackground(new java.awt.Color(150, 200, 225));

chooseColorButton.setFont(new java.awt.Font("Bahnschrift", 0, 14)); // NOI18N

chooseColorButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/chooseColor.png"))); // NOI18N

chooseColorButton.setText("Choose color");

chooseColorButton.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

chooseColorButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

chooseColorButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

chooseColorButton.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

chooseColorButtonMouseReleased(evt);

}

});

gridBagConstraints = new java.awt.GridBagConstraints();

gridBagConstraints.insets = new java.awt.Insets(0, 5, 0, 5);

add(chooseColorButton, gridBagConstraints);

strokePanel.setBackground(new Color(150, 200, 225));

strokePanel.setMaximumSize(new java.awt.Dimension(90, 90));

strokePanel.setMinimumSize(new java.awt.Dimension(90, 90));

strokePanel.setPreferredSize(new java.awt.Dimension(90, 90));

strokeLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/icons/stroke.png"))); // NOI18N

strokePanel.add(strokeLabel);

strokeSpinner.setModel(new javax.swing.SpinnerNumberModel(1, 1, 10, 1));

strokeSpinner.addChangeListener(new javax.swing.event.ChangeListener() {

public void stateChanged(javax.swing.event.ChangeEvent evt) {

strokeSpinnerStateChanged(evt);

}

});

strokePanel.add(strokeSpinner);

add(strokePanel, new java.awt.GridBagConstraints());

previewPanel = new PreviewPanel(frame.getCanvas().getTool(),frame.getCanvas().getStroke(),frame.getCanvas().getBrushColor(),frame.getCanvas().getFillColor());

previewPanel.setBackground(Color.white);

previewPanel.setBackground(new java.awt.Color(255, 255, 255));

previewPanel.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0), 2));

previewPanel.setMaximumSize(new java.awt.Dimension(90, 90));

previewPanel.setMinimumSize(new java.awt.Dimension(90, 90));

previewPanel.setPreferredSize(new java.awt.Dimension(90, 90));

gridBagConstraints = new java.awt.GridBagConstraints();

gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;

gridBagConstraints.weightx = 5.0;

gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 45);

add(previewPanel, gridBagConstraints);

}// </editor-fold>

private void brushButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setTool(Tool.BRUSH);

updatePreview();

}

private void eraserButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setTool(Tool.ERASER);

updatePreview();

}

private void chooseColorButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

Color color = JColorChooser.showDialog(this, "Choose brush color", frame.getCanvas().getBrushColor());

if (color != null) {

frame.getCanvas().setBrushColor(color);

}

updatePreview();

}

private void pipetteButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

JPanel panel = new JPanel();

JRadioButton button1 = new JRadioButton("Brush color");

JRadioButton button2 = new JRadioButton("Fill color");

JRadioButton button3 = new JRadioButton("Background color");

ButtonGroup group = new ButtonGroup();

group.add(button1);

group.add(button2);

group.add(button3);

panel.add(button1);

panel.add(button2);

panel.add(button3);

button1.setSelected(true);

JOptionPane.showMessageDialog(null, panel, "Choose color aim", JOptionPane.DEFAULT\_OPTION, null);

if (group.isSelected(button1.getModel())) {

frame.getCanvas().setPipetteChoice(CanvasPanel.BRUSH\_COLOR);

} else if (group.isSelected(button2.getModel())) {

frame.getCanvas().setPipetteChoice(CanvasPanel.FILL\_COLOR);

} else if (group.isSelected(button3.getModel())) {

frame.getCanvas().setPipetteChoice(CanvasPanel.BACKGROUND\_COLOR);

} else {

frame.getCanvas().setPipetteChoice(0);

return;

}

frame.getCanvas().setTool(Tool.PIPETTE);

updatePreview();

}

private void formMouseClicked(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

if (frame.getCanvas().getTool() == Tool.PIPETTE) {

Color color = new Color(getScreenComponent(this).getRGB(evt.getX(), evt.getY()));

switch (frame.getCanvas().getPipetteChoice()) {

case CanvasPanel.BRUSH\_COLOR: {

frame.getCanvas().setBrushColor(color);

break;

}

case CanvasPanel.FILL\_COLOR: {

frame.getCanvas().setFillColor(color);

break;

}

case CanvasPanel.BACKGROUND\_COLOR: {

DrawnRectangle background = new DrawnRectangle(frame.getCanvas().getStroke(), color, color, 0, 0, CanvasPanel.WIDTH, CanvasPanel.HEIGHT);

frame.getCanvas().setBackgroundRectangle(background);

break;

}

}

}

}

private void setBackgroundButtonMouseClicked(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

Color color = JColorChooser.showDialog(this, "Choose background color", frame.getCanvas().getBackgroundColor());

if (color != null) {

DrawnRectangle background = new DrawnRectangle(frame.getCanvas().getStroke(), color, color, 0, 0, CanvasPanel.WIDTH, CanvasPanel.HEIGHT);

frame.getCanvas().setBackgroundRectangle(background);

}

}

private void strokeSpinnerStateChanged(javax.swing.event.ChangeEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setStroke(new BasicStroke((Integer) strokeSpinner.getValue()));

updatePreview();

}

private void fillColorButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

Color color = JColorChooser.showDialog(this, "Choose fill color", frame.getCanvas().getFillColor());

if (frame.getCanvas().getFillColor() != null) {

frame.getCanvas().setFillColor(color);

}

updatePreview();

}

private void rectangleButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setTool(Tool.SHAPE\_RECTANGLE);

updatePreview();

}

private void insertPictureButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

JFileChooser picChooser = new JFileChooser(System.getProperty("user.home"));

picChooser.setFileFilter(new FileFilter() {

@Override

public boolean accept(File f) {

String fileName = f.getName();

return fileName.endsWith(".png");

}

@Override

public String getDescription() {

return ".png";

}

});

int choice = picChooser.showOpenDialog(null);

switch (choice) {

case JFileChooser.APPROVE\_OPTION: {

try {

BufferedImage image = ImageIO.read(picChooser.getSelectedFile());

if (image == null) {

throw new IOException();

}

frame.getCanvas().setImage(image, 0, 0);

frame.getCanvas().repaint();

frame.getCanvas().setTool(Tool.INSERT\_PICTURE);

frame.getCanvas().setIsMouseOnImage(true);

updatePreview();

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Choose picture(file ending with .png, .jpg...)", "Invalid file chosen", JOptionPane.ERROR\_MESSAGE);

}

break;

}

}

}

private void insertTextButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

JPanel panel = new JPanel();

JComboBox comboBox = new JComboBox();

Font font;

DrawnText text;

comboBox.addItem("Arial");

comboBox.addItem("Comic Sans MS");

comboBox.addItem("Courier New");

comboBox.addItem("Georgia1");

comboBox.addItem("Impact");

comboBox.addItem("Lucida");

comboBox.addItem("Palatino Linotype");

comboBox.addItem("Tahoma");

comboBox.addItem("Times New Roman");

comboBox.addItem("Trebuchet MS1");

comboBox.addItem("Verdana");

comboBox.addItem("MS Sans Serif4");

comboBox.addItem("MS Serif4");

JRadioButton button1 = new JRadioButton("Italic");

JRadioButton button2 = new JRadioButton("Bold");

ButtonGroup group = new ButtonGroup();

group.add(button1);

group.add(button2);

button1.setSelected(true);

JSpinner spinner = new JSpinner();

spinner.setModel(new javax.swing.SpinnerNumberModel(12, 12, 72, 1));

JTextField textField = new JTextField("Enter your text");

textField.setBounds(0, 0, 80, 40);

panel.add(comboBox);

panel.add(button1);

panel.add(button2);

panel.add(spinner);

panel.add(textField);

panel.setLayout(new FlowLayout());

JOptionPane.showMessageDialog(null, panel, "Choose your font", JOptionPane.DEFAULT\_OPTION, null);

if (group.isSelected(button1.getModel())) {

font = new Font((String) comboBox.getSelectedItem(), Font.ITALIC, (Integer) spinner.getValue());

text = new DrawnText(font, textField.getText(), frame.getCanvas().getBrushColor(), (Integer) spinner.getValue(), (Integer) spinner.getValue());

frame.getCanvas().setText(text);

frame.getCanvas().setTool(Tool.INSERT\_TEXT);

frame.getCanvas().repaint();

frame.getCanvas().setIsMouseOnText(true);

updatePreview();

} else if (group.isSelected(button2.getModel())) {

font = new Font((String) comboBox.getSelectedItem(), Font.BOLD, (Integer) spinner.getValue());

text = new DrawnText(font, textField.getText(), frame.getCanvas().getBrushColor(), (Integer) spinner.getValue(), (Integer) spinner.getValue());

frame.getCanvas().setText(text);

frame.getCanvas().setTool(Tool.INSERT\_TEXT);

frame.getCanvas().repaint();

frame.getCanvas().setIsMouseOnText(true);

updatePreview();

}

}

private void circleButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setTool(Tool.SHAPE\_CIRCLE);

updatePreview();

}

private void triangleButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setTool(Tool.SHAPE\_TRIANGLE);

updatePreview();

}

private void lineButtonMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

frame.getCanvas().setTool(Tool.SHAPE\_LINE);

updatePreview();

}

private void fillCheckBoxMouseReleased(java.awt.event.MouseEvent evt) {

frame.getCanvas().reset();

if (fillCheckBox.isSelected()) {

frame.getCanvas().setFillColor(frame.getCanvas().getBrushColor());

} else {

frame.getCanvas().setFillColor(null);

}

updatePreview();

}

/\*\*

\* Method to update the preview panel

\*/

public void updatePreview() {

((PreviewPanel)previewPanel).update(frame.getCanvas().getTool(),frame.getCanvas().getStroke(),frame.getCanvas().getBrushColor(),frame.getCanvas().getFillColor());

}

// Variables declaration - do not modify

private javax.swing.JButton brushButton;

private javax.swing.JButton chooseColorButton;

private javax.swing.JButton circleButton;

private javax.swing.JButton eraserButton;

private javax.swing.JCheckBox fillCheckBox;

private javax.swing.JButton fillColorButton;

private javax.swing.JButton insertPictureButton;

private javax.swing.JButton insertTextButton;

private javax.swing.JButton lineButton;

private javax.swing.JButton pipetteButton;

private javax.swing.JPanel previewPanel;

private javax.swing.JButton rectangleButton;

private javax.swing.JButton setBackgroundButton;

private javax.swing.JPanel shapesPanel;

private javax.swing.JLabel strokeLabel;

private javax.swing.JPanel strokePanel;

private javax.swing.JSpinner strokeSpinner;

private javax.swing.JButton triangleButton;

// End of variables declaration

}

**PreviewPanel**

package graphicsEditor;

import graphicsEditor.drawnShapes.DrawnCircle;

import graphicsEditor.drawnShapes.DrawnLine;

import graphicsEditor.drawnShapes.DrawnRectangle;

import graphicsEditor.drawnShapes.DrawnTriangle;

import graphicsEditor.instruments.Tool;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Stroke;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.JOptionPane;

/\*\*Panel for previewing all tools

\*

\* @author VADYM NAKYTNIAK

\*/

public class PreviewPanel extends javax.swing.JPanel {

Tool tool;

Stroke stroke;

Color color;

Color fillColor;

/\*\*

\* Creates new form PreviewPanel

\*

\* @param tool

\* @param stroke

\* @param color

\* @param fillColor

\*/

public PreviewPanel(Tool tool, Stroke stroke, Color color, Color fillColor) {

initComponents();

this.tool = tool;

this.stroke = stroke;

this.color = color;

this.fillColor = fillColor;

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g); //To change body of generated methods, choose Tools | Templates.

Graphics2D g2d = (Graphics2D) g;

g2d.setStroke(stroke);

g2d.setColor(color);

switch (tool) {

case INSERT\_PICTURE: {

BufferedImage image;

try {

image = (ImageIO.read(new File(System.getProperty("user.dir") + "\\src\\icons\\insertPicture.png")));

g2d.drawImage(image, 5, 10, null);

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Image insertPicture.png is not found", "Image is not foun", JOptionPane.ERROR\_MESSAGE);

}

break;

}

case INSERT\_TEXT: {

BufferedImage image;

try {

image = (ImageIO.read(new File(System.getProperty("user.dir") + "\\src\\icons\\insertText.png")));

g2d.drawImage(image, 0, 20, null);

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Image insertText.png is not found", "Image is not foun", JOptionPane.ERROR\_MESSAGE);

}

break;

}

case BRUSH: {

g2d.draw(new DrawnLine(stroke, color, 40, 45, 55, 45));

break;

}

case ERASER: {

BufferedImage image;

try {

image = (ImageIO.read(new File(System.getProperty("user.dir") + "\\src\\icons\\eraser.png")));

g2d.drawImage(image, 10, 20, null);

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Image eraser.png is not found", "Image is not foun", JOptionPane.ERROR\_MESSAGE);

}

break;

}

case PIPETTE: {

BufferedImage image;

try {

image = (ImageIO.read(new File(System.getProperty("user.dir") + "\\src\\icons\\pipette.png")));

g2d.drawImage(image, 20, 10, null);

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Image pipette.png is not found", "Image is not foun", JOptionPane.ERROR\_MESSAGE);

}

break;

}

case SHAPE\_RECTANGLE: {

DrawnRectangle rect = new DrawnRectangle(stroke, color, fillColor, 20, 20, 50, 50);

if (fillColor != null) {

g2d.setColor(fillColor);

g2d.fill(rect);

}

g2d.setColor(color);

g2d.draw(rect);

break;

}

case SHAPE\_CIRCLE: {

DrawnCircle rect = new DrawnCircle(stroke, color, fillColor, 20, 20, 50, 50);

if (fillColor != null) {

g2d.setColor(fillColor);

g2d.fill(rect);

}

g2d.setColor(color);

g2d.draw(rect);

break;

}

case SHAPE\_TRIANGLE: {

DrawnTriangle rect = new DrawnTriangle(stroke, color, fillColor, 20, 70, 45, 20, 70, 70);

if (fillColor != null) {

g2d.setColor(fillColor);

g2d.fill(rect);

}

g2d.setColor(color);

g2d.draw(rect);

break;

}

case SHAPE\_LINE: {

DrawnLine rect = new DrawnLine(stroke, color, 20, 20, 70, 70);

g2d.draw(rect);

break;

}

case NO\_TOOL: {

break;

}

case CUT\_OUT:{

DrawnRectangle rect = new DrawnRectangle(new BasicStroke(1f), Color.red, null, 20, 20, 50, 50);

g2d.setColor(rect.getColor());

g2d.draw(rect);

break;

}

}

}

public Tool getTool() {

return tool;

}

public Stroke getStroke() {

return stroke;

}

public Color getColor() {

return color;

}

public Color getFillColor() {

return fillColor;

}

public void setTool(Tool tool) {

this.tool = tool;

}

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

public void setColor(Color color) {

this.color = color;

}

public void setFillColor(Color fillColor) {

this.fillColor = fillColor;

}

public void update(Tool tool, Stroke stroke, Color color, Color fillColor) {

this.tool = tool;

this.stroke = stroke;

this.color = color;

this.fillColor = fillColor;

repaint();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

setMaximumSize(new java.awt.Dimension(90, 90));

setMinimumSize(new java.awt.Dimension(90, 90));

setPreferredSize(new java.awt.Dimension(90, 90));

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

this.setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 90, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 90, Short.MAX\_VALUE)

);

}// </editor-fold>

// Variables declaration - do not modify

// End of variables declaration

}

**Drawable**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Stroke;

/\*\*

\* Interface for all objects to be drawn

\* @author VADYM NAKYTNIAK

\*/

public interface Drawable {

Stroke getStroke();

Color getColor();

Color getFillColor();

void setStroke(Stroke stroke);

void setColor(Color color);

void setFillColor(Color color);

}

**DrawnCircle**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Stroke;

import java.awt.geom.Ellipse2D;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class DrawnCircle extends Ellipse2D.Double implements Drawable {

private Stroke stroke;

private Color color;

private Color fillColor;

public DrawnCircle(Stroke stroke, Color color, Color fillColor, double x, double y, double w, double h) {

super(x, y, w, h);

this.stroke = stroke;

this.color = color;

this.fillColor = fillColor;

}

@Override

public Stroke getStroke() {

return stroke;

}

@Override

public Color getColor() {

return color;

}

@Override

public Color getFillColor() {

return fillColor;

}

@Override

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

@Override

public void setColor(Color color) {

this.color = color;

}

@Override

public void setFillColor(Color fillColor) {

this.fillColor = fillColor;

}

}

**DrawnCircle**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Stroke;

import java.awt.image.BufferedImage;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class DrawnImage extends BufferedImage implements Drawable {

private BufferedImage image;

private int x;

private int y;

public DrawnImage(BufferedImage image, int x, int y) {

super(image.getWidth(), image.getHeight(), image.getType());

this.image = image;

this.x = x;

this.y = y;

}

public DrawnImage(int width, int height, int imageType) {

super(width, height, imageType);

}

/\*\*

\*

\* @return null

\*/

@Override

public Stroke getStroke() {

return null;

}

/\*\*

\*

\* @return null

\*/

@Override

public Color getColor() {

return null;

}

/\*\*

\*

\* @return null

\*/

@Override

public Color getFillColor() {

return null;

}

public BufferedImage getImage() {

return image;

}

public int getX() {

return x;

}

public int getY() {

return y;

}

/\*\*

\* EMPTY method

\*

\* @param stroke

\*/

@Override

public void setStroke(Stroke stroke) {

}

/\*\*

\* EMPTY method

\*

\* @param color

\*/

@Override

public void setColor(Color color) {

}

/\*\*

\* EMPTY method

\*

\* @param color

\*/

@Override

public void setFillColor(Color color) {

}

public void setImage(BufferedImage image) {

this.image = image;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

}

**DrawnLine**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Stroke;

import java.awt.geom.Line2D;

import java.awt.geom.Point2D;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class DrawnLine extends Line2D.Double implements Drawable {

private Stroke stroke;

private Color color;

private Color fillColor;

public DrawnLine(Stroke stroke, Color color, double x1, double y1, double x2, double y2) {

super(x1, y1, x2, y2);

this.stroke = stroke;

this.color = color;

fillColor = color;

}

public DrawnLine(Stroke stroke, Color color, Point2D p1, Point2D p2) {

super(p1, p2);

this.stroke = stroke;

this.color = color;

}

@Override

public Stroke getStroke() {

return stroke;

}

@Override

public Color getColor() {

return color;

}

@Override

public Color getFillColor() {

return fillColor;

}

@Override

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

@Override

public void setColor(Color color) {

this.color = color;

}

@Override

public void setFillColor(Color fillColor) {

this.fillColor = fillColor;

}

}

**DrawnRectangle**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Rectangle;

import java.awt.Stroke;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class DrawnRectangle extends Rectangle implements Drawable {

private Stroke stroke;

private Color color;

private Color fillColor;

public DrawnRectangle(Stroke stroke, Color color, Color fillColor, int x, int y, int width, int height) {

super(x, y, width, height);

this.stroke = stroke;

this.color = color;

this.fillColor = fillColor;

}

@Override

public Stroke getStroke() {

return stroke;

}

@Override

public Color getColor() {

return color;

}

@Override

public Color getFillColor() {

return fillColor;

}

@Override

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

@Override

public void setColor(Color color) {

this.color = color;

}

@Override

public void setFillColor(Color fillColor) {

this.fillColor = fillColor;

}

}

**DrawnText**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Font;

import java.awt.Stroke;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class DrawnText implements Drawable {

private Font font;

private String text;

private Color color;

private int x;

private int y;

public DrawnText(Font font, String text, Color color, int x, int y) {

this.font = font;

this.text = text;

this.color = color;

this.x = x;

this.y = y;

}

/\*\*

\*

\* @return null

\*/

@Override

public Stroke getStroke() {

return null;

}

/\*\*

\*

\* @return color

\*/

@Override

public Color getColor() {

return color;

}

/\*\*

\*

\* @return color

\*/

@Override

public Color getFillColor() {

return color;

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public Font getFont() {

return font;

}

public String getText() {

return text;

}

/\*\*

\* EMPTY method

\*

\* @param stroke

\*/

@Override

public void setStroke(Stroke stroke) {

}

/\*\*

\* EMPTY method

\*

\* @param color

\*/

@Override

public void setColor(Color color) {

this.color = color;

}

/\*\*

\*

\*

\* @param color

\*/

@Override

public void setFillColor(Color color) {

this.color = color;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public void setFont(Font font) {

this.font = font;

}

public void setText(String text) {

this.text = text;

}

}

**DrawnTriangle**

package graphicsEditor.drawnShapes;

import java.awt.Color;

import java.awt.Stroke;

import java.awt.geom.Path2D;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class DrawnTriangle extends Path2D.Double implements Drawable {

private Stroke stroke;

private Color color;

private Color fillColor;

public DrawnTriangle(Stroke stroke, Color color, Color fillColor, double x1, double y1, double x2, double y2, double x3, double y3) {

this.stroke = stroke;

this.color = color;

this.fillColor = fillColor;

moveTo(x1, y1);

lineTo(x2, y2);

lineTo(x3, y3);

closePath();

}

@Override

public Stroke getStroke() {

return stroke;

}

@Override

public Color getColor() {

return color;

}

@Override

public Color getFillColor() {

return fillColor;

}

@Override

public void setStroke(Stroke stroke) {

this.stroke = stroke;

}

@Override

public void setColor(Color color) {

this.color = color;

}

@Override

public void setFillColor(Color fillColor) {

this.fillColor = fillColor;

}

}

**Tool**

package graphicsEditor.instruments;

/\*\*Enum with all tools used for drawing

\*

\* @author VADYM NAKYTNIAK

\*/

public enum Tool {

INSERT\_PICTURE,INSERT\_TEXT,BRUSH,ERASER,PIPETTE,SHAPE\_RECTANGLE,SHAPE\_CIRCLE,SHAPE\_TRIANGLE,SHAPE\_LINE,NO\_TOOL,CUT\_OUT;

}

**ImageTransfarable**

package graphicsEditor.instruments;

import java.awt.Image;

import java.awt.datatransfer.DataFlavor;

import java.awt.datatransfer.Transferable;

import java.awt.datatransfer.UnsupportedFlavorException;

import java.io.IOException;

/\*\*

\*

\* @author VADYM NAKYTNIAK

\*/

public class ImageTransfarable implements Transferable {

final private Image theImage;

public ImageTransfarable(Image image) {

this.theImage = image;

}

@Override

public DataFlavor[] getTransferDataFlavors() {

return new DataFlavor[]{DataFlavor.imageFlavor};

}

@Override

public boolean isDataFlavorSupported(DataFlavor flavor) {

return flavor.equals(DataFlavor.imageFlavor);

}

@Override

public Object getTransferData(DataFlavor flavor) throws UnsupportedFlavorException, IOException {

if (flavor.equals(DataFlavor.imageFlavor)) {

return theImage;

} else {

throw new UnsupportedFlavorException(flavor);

}

}

}