Dokumentacja projektu

Rysowanie

Zdzisław Kozłecki 187010
Konrad Kania 186757

Projekt przygotowany w ramach przedmiotu "Programowanie systemów rozproszonych"

Uniwersytet Ekonomiczny w Krakowie, 2017

1. Opis projektu

Projekt realizuje wspólne rysowanie/pisanie po pewnym obszarze roboczym. Działaniem zatem przypomina środowisko do tworzenia grafiki rastrowej w bardzo podstawowej formie. Po obszarze roboczym może rysować dwóch lub większa liczba użytkowników. Połączenie jest realizowane przez sieć komputerową poprzez architekturę klient-serwer.

2. Zasady działania

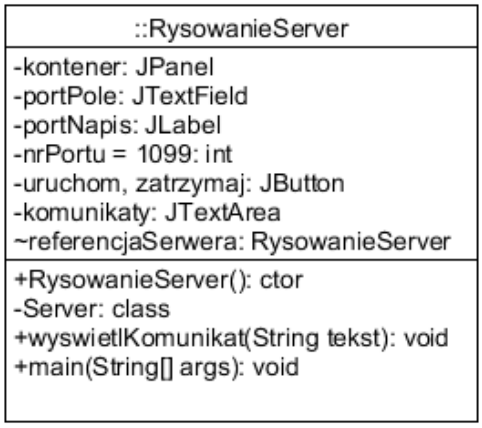
Przed uruchomieniem właściwego programu do rysowania, znajdującego się po stronie klienta, należy zgłosić żądanie o podłączenia do środowiska. Aby to uczynić, trzeba włączyć moduł serwera i nacisnąć na przycisk uruchom. Następnie włączamy moduły klientów. Po wykonaniu tych czynności mamy program gotowy do działania. Instrukcja obsługi właściwego rysowania opisana została poniżej.

3. Wykorzystane zagadnienia

W programie wykorzystane zostały między innymi zagadnienia poznane w ramach przedmiotu. Kilkuosobowe rysowanie jest realizowane za pośrednictwem zdalnego wywołania metod(ang. *RMI – Remote Method Invocation*) . Środowisko graficzne utworzono za pomocą bibliotek Swing oraz AWT. Skorzystano także z wątków(klasa odpowiedzialna za uruchomienie serwera) i biblioteki umożliwiającej obsługę operacji wejścia/wyjścia(zapis i odczyt plików graficznych).

4. Serwer Rysowanie

4.1. Klasa RysowanieServer



Pola klasy:

- JPanel kontener – jest to kontener umieszczony w górnej części okna serwera, zawierający JLabel dla portu, JTextField dla portu oraz trzy przyciski (JButton uruchom, zatrzymaj, klient)
- JLabel portNapis – napis informujący jaką wartość umieścić w JTextField
- JTextField portPole – pole tekstowe do wpisywania numeru portu
- int nrPortu – domyślnie ustawiona w JTextField wartość portu
- JButton uruchom – przycisk do uruchamiania serwera
- JButton zatrzymaj – przycisk do zatrzymywania serwera
- JButton klient – przycisk do uruchamiania klienta
- JTextArea komunikaty – pole tekstowe do wyświetlania komunikatów
- RysowanieServer referencjaSerwera – referencja do serwera używana w konstruktorze

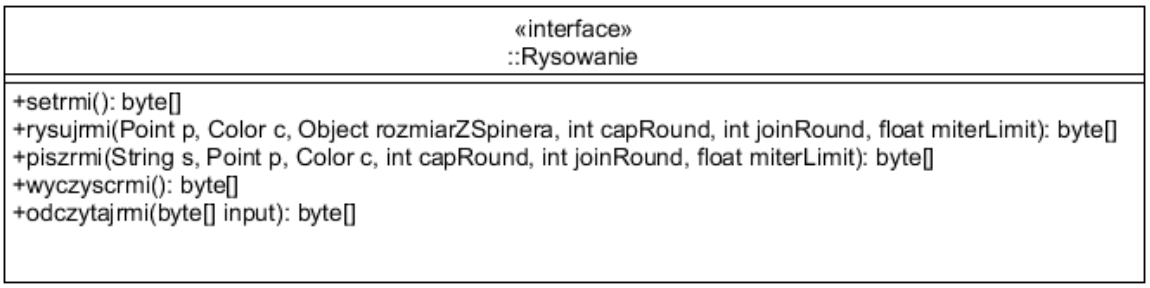
Metody klasy:

- RysowanieServer() – konstruktor
- void wyswietlKomunikat(String tekst) – wyświetla komunikaty na serwerze

Klasa wewnętrzna:

- Server – klasa rejestrująca serwer na porcie

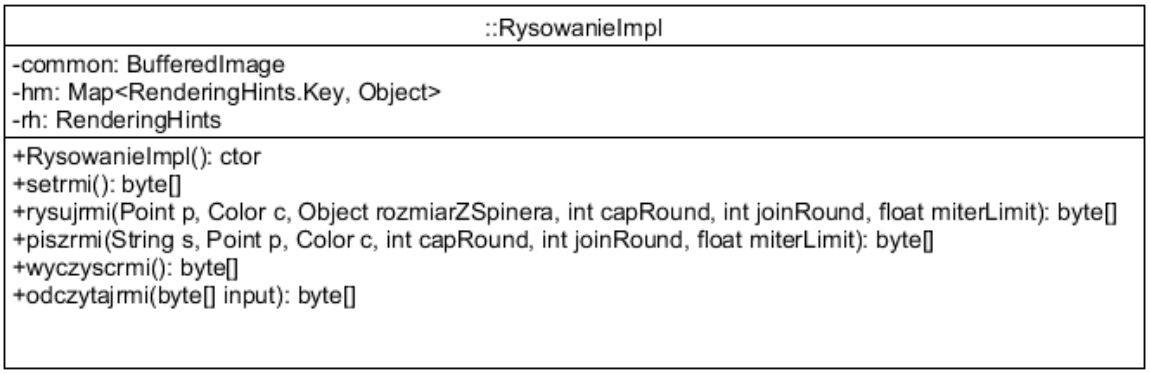
4.2. Interfejs Rysowanie



Metody interfejsu (nagłówki metod):

- byte[] setrmi() – nagłówek metody ustawiającej początkowy stan obszaru roboczego poprzez RMI
- byte[] rysujrmi(Point p, Color c, Object rozmiarZSpinera, int capRound, int joinRound, float miterLimit) – nagłówek metody rysującej poprzez RMI
- byte[] piszami(String s, Point p, Color c, int capRound, int joinRound, float miterLimit) – nagłówek metody pozwalającej na dodawanie tekstu poprzez RMI
- byte[] wyczyscrmi() – nagłówek metody pozwalającej wyczyścić obszar roboczy poprzez RMI
- byte[] odczytajrmi() – nagłówek metody pozwalającej odczytać poprzez RMI uprzednio zapisany obraz w pliku graficznym

4.3. Klasa RysowanieImpl



Pola klasy:

- BufferedImage common – referencja do klasy po stronie serwera, będącej obszarem roboczym po którym się rysuje
- Map<RenderingHints.Key, Object> hm – klasa generyczna przekazywana do klasy RenderingHints
- RenderingHints rh – klasa wykorzystywana do prawidłowego działania rysowania z użyciem BufferedImage

Metody klasy:

- RysowanieImpl – konstruktor
- byte[] setrmi() – metoda ustawiająca początkowy stan obszaru roboczego poprzez RMI
- byte[] rysujrmi(Point p, Color c, Object rozmiarZSpinera, int capRound, int joinRound, float miterLimit) – metoda pozwalająca na rysowanie poprzez RMI
- byte[] piszami(String s, Point p, Color c, int capRound, int joinRound, float miterLimit) – metoda pozwalająca na dodawanie tekstu poprzez RMI
- byte[] wyczyscrmi() – metoda pozwalająca wyczyścić obszar roboczy poprzez RMI
- byte[] odczytajrmi() – metoda pozwalająca odczytać poprzez RMI uprzednio zapisany obraz w pliku graficznym

5. Klient Rysowanie

5.1. Klasa RysowanieClient

::RysowanieClient
-obszar: JPanel -kontenerPrzyciskGora, kontenerPrzyciskiDol: JPanel -kontenerPrzybornik: JToolBar -zapis, wczytywanie, wyczysc, kolorBT: JButton -przybornikLabel, obszarLabel, rozmiarLabel: JLabel -rysuj, pisz: JRadioButton -rozmiarRysowania: JSpinner -rozmiarRysowaniaTryb: SpinnerNumberModel -zapisOknoDialog: JFileChooser -odczytOknoDialog: JFileChooser +RYSOWANIE_NARZEDZIE = 0: int +TEXT_NARZEDZIE = 1: int -aktywneNarzedzie: int -BIwyjsciowyObszarRob: BufferedImage -BIzmienianyObszarRob: BufferedImage -probkaKoloru = new BufferedImage(16, 16, BufferedImage.TYPE_INT_RGB): BufferedImage -rh: RenderingHints -kolor: Color -nrPortu: int +obrys = new BasicStroke(7, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND, 1.7f): Stroke ~cap_round = 1, join_round = 1: int ~miterlimit = 1.7f: float -registry: Registry -rys: Rysowanie -post: byte[]
+RysowanieClient(int port): ctor +ustawObszar(BufferedImage bi): void -KolorML: class -ObszarMML class -ObszarMA class -RozmiarRysCL class -JRadioAL class +kolor(Color kolor): void +rysowanie(Point w): void +pisanie(Point wspolrzedna): void +wyczysc(BufferedImage bufferedImage): void +wyczyscObszar(BufferedImage bufferedImage): void +main(String[] args): void

Pola klasy:

- JPanel obszar – kontener w którym umieszczany jest obszar roboczy
- JPanel kontenerPrzyciskGora – kontener zlokalizowany w górnej części interfejsu użytkownika , w którym umieszczony jest przycisk do czyszczenia obszaru roboczego
- JPanelkontenerPrzyciskiDol – kontener zlokalizowany w dolnej części interfejsu użytkownika, w którym umieszczone są przyciski zapisu i odczytywania pliku graficznego
- JToolBar kontenerPrzybornik – pasek narzędzi w którym umieszczany jest przybornik
- JButton zapis – przycisk do zapisywania dotychczasowej pracy w pliku graficznym

- JButton wczytywanie – przycisk do wczytywania pliku graficznego i ustawiania go w obszarze roboczym
- JButton wyczysc – przycisk do czyszczenia obszaru roboczego
- JButton kolorBT - przycisk do zmieniania koloru rysowania
- JLabel przybornikLabel – napis przybornika
- JLabel obszarLabel – w tym komponencie ustawiany jest obszar roboczy, który następnie jest wstawiany do kontenera JPanel obszar
- JLabel rozmiarLabel – napis w przyborniku dotyczący rozmiaru
- JRadioButton rysuj – okrągłe pole zaznaczania wskazujące, że aktualnym narzędziem przybornika jest pędzel
- JRadioButton pisz - okrągłe pole zaznaczania wskazujące, że aktualnym narzędziem przybornika jest tekst
- SpinnerNumberModel rozmiarRysowaniaTryb – komponent wykorzystywany w konstruktorze JSpinner
- JSpinner rozmiarRysowania – komponent do określania rozmiaru rysowania
- JFileChooser zapisOknoDialog– okno dialogowe do wskazywania katalogu w którym zapisac plik graficzny
- JFileChooser odczytOknoDialog – okno dialogowe do wskazywania pliku graficznego, który ma zostać odczytany i wstawiony do obszaru roboczego
- int aktywneNarzedzia – zmienna pomocnicza, która wskazuje aktualnie zaznaczone narzędzie
- BufferedImage BIwyjsciowyObszarRob – referencja do klasy BufferedImage, która przechowuje początkowy stan obszaru roboczego
- BufferedImage BIzmienianyObszarRob – referencja do klasy BufferedImage, która przechowuje aktualny stan obszaru roboczego
- Color kolor – referencja do klasy Color, przechowująca aktualnie wybrany kolor
- RenderingHints rh – referencja do klasy RenderingHints, wykorzystywana do prawidłowego działania rysowania z użyciem BufferedImage
- int nrPortu – zmienna w której przechowywany jest numer portu
- Stroke obrys – referencja do klasy Stroke, wykorzystywana przy zmianie rozmiaru rysowania

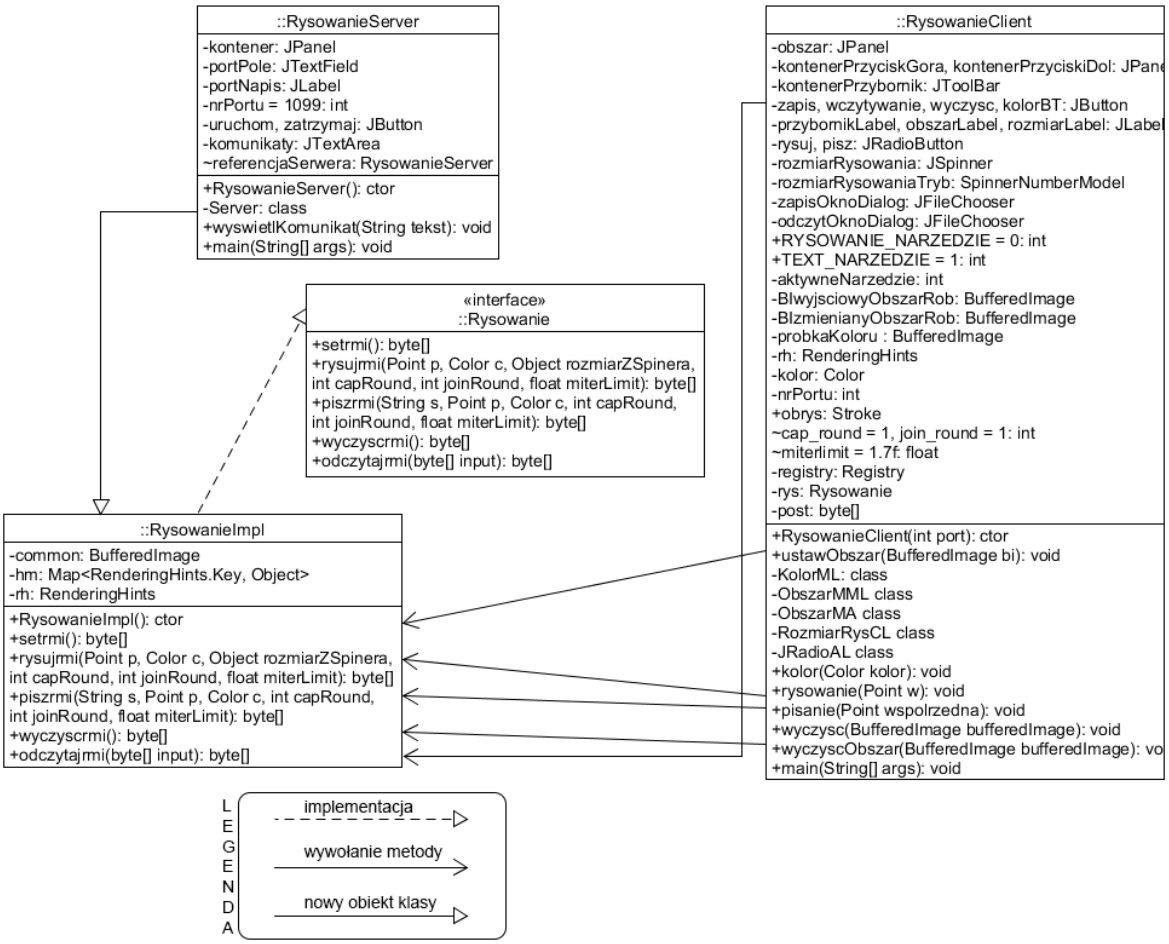
Metody klasy:

- RysowanieClient(int port) – konstruktor
- void ustawObszar(BufferedImage bi) – metoda wykorzystywana do ustawienia początkowego stanu obszaru roboczego
- void kolor(Color kolor) – metoda do ustawiania aktualnie wybranego koloru
- void rysowanie(Point w) – metoda służąca do rysowania po obszarze roboczym
- void pisanie(Point wspolrzedna) – metoda służąca do pisania po obszarze roboczym
- void wyczysc(BufferedImage bufferedImage) – metoda wykorzystywana przy ustawianiu nowo wybranego koloru
- void wyczyscObszar(BufferedImage bufferedImage) – metoda służąca do czyszczenia obszaru roboczego

Klasy wewnętrzne:

- oprócz tego w klasie znajdują się klasy wewnętrzne do obsługi zdarzeń

6. Diagram UML

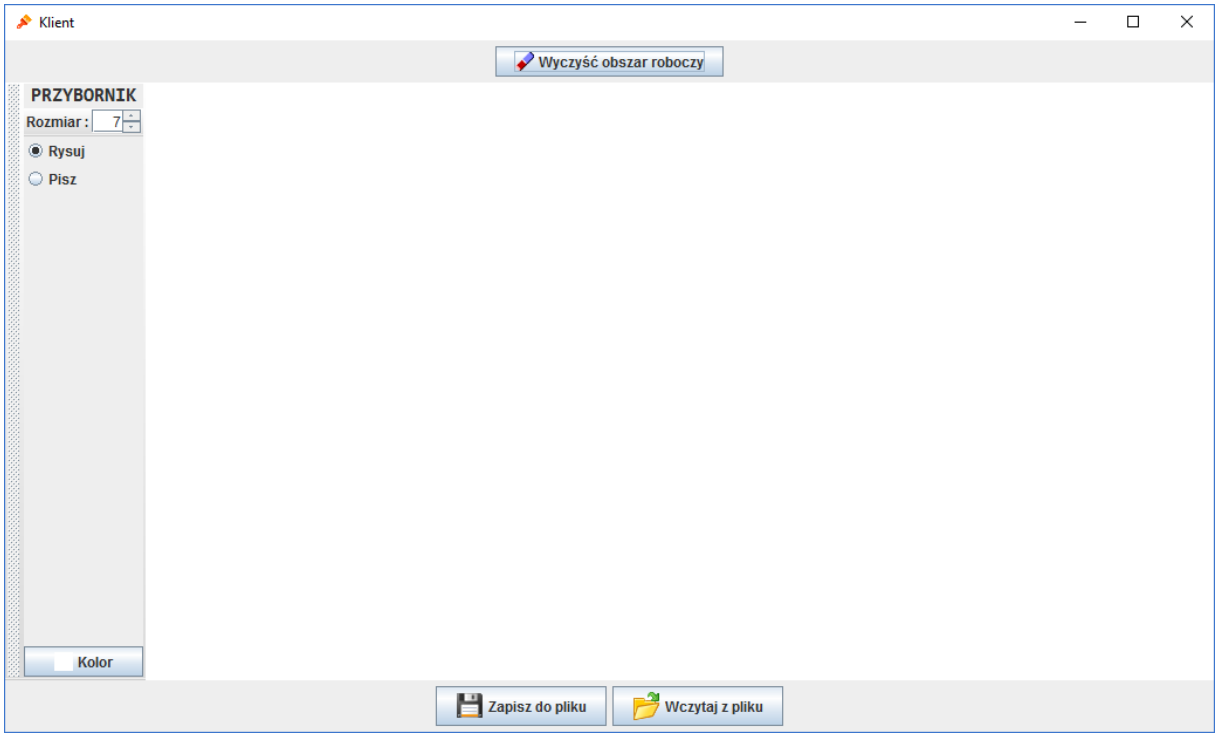


7. Instrukcja obsługi

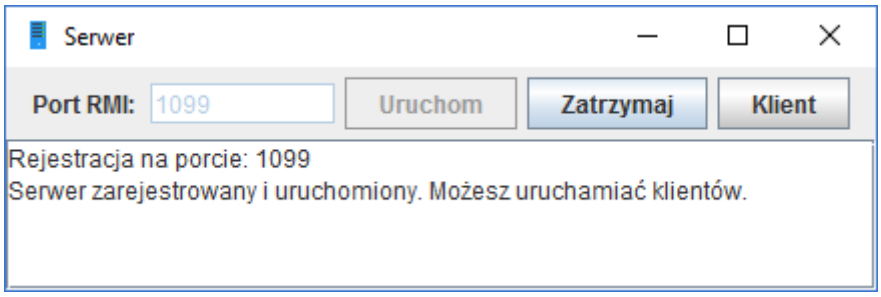
Rysowanie odbywa się po białym obszarze roboczym, umieszczonym w centralnej części okna. Na lewo od tego obszaru znajduje się przybornik. W górnej jego części zlokalizowany jest element odpowiedzialny za określenie rozmiaru pędzla w skali 1-70. Nieco niżej umiejscowione zostały pola wyboru narzędzia. Przy ich użyciu wskazujemy czy chcemy dokonać wstawienia napisu czy zacząć rysować. Równoczesne ich zaznaczenie nie jest możliwe. Domyślnie zaznaczone jest to do rysowania. Na samym dole przybornika usytuowany jest przycisk włączający paletę kolorów. Pożądany kolor wskazujemy przy użyciu gotowych próbek lub korzystając z umieszczonych modeli barw, położonych w kolejnych kartach palety. Dostępne są najbardziej popularne modele, takie jak HSV, RGB, CMYK i HSL. Nad obszarem roboczym osadzony został przycisk, który czyści element po którym rysujemy.

W ten sposób możemy zacząć rysować od początku bez konieczności ponownego uruchamiania programu. Na samym dole przypięte są jeszcze dwa przyciski. Jeden pozwala na zapis obrazka, drugi z kolei na jego odczyt. Do dyspozycji użytkownika są najpopularniejsze formaty, jak *.jpg czy *.png.

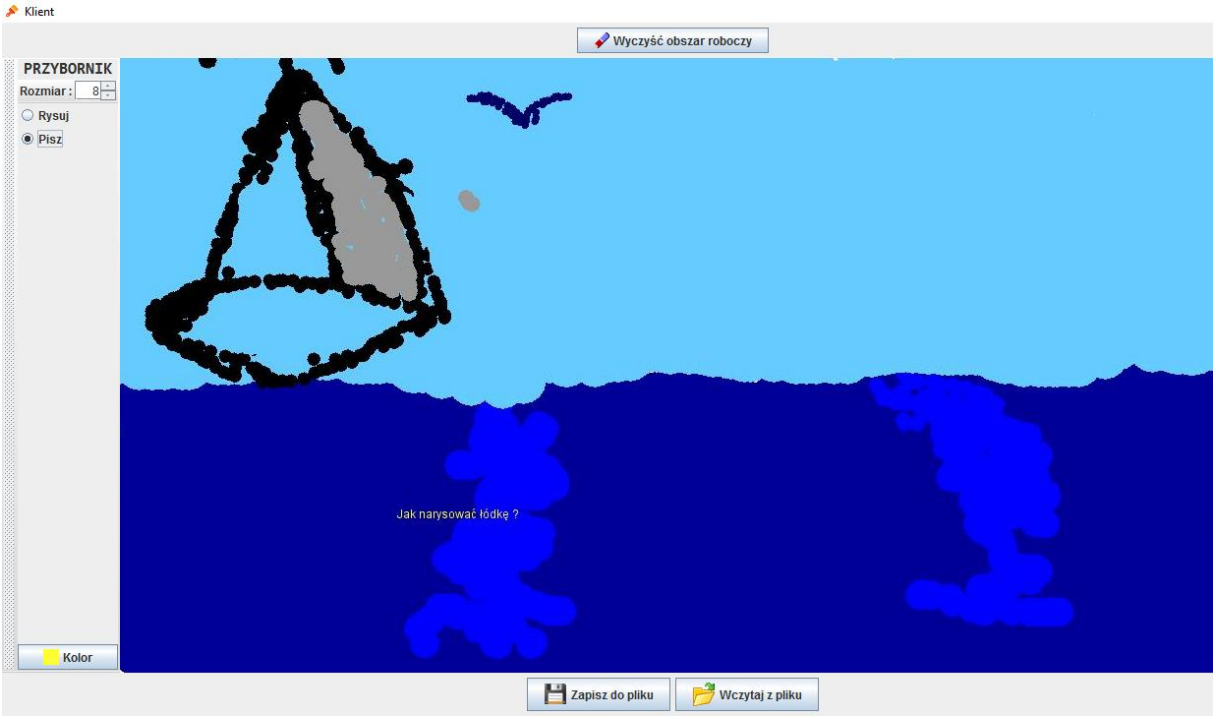
Panel do rysowania, widoczny po uruchomieniu klienta, z widocznymi wszystkimi opisywanymi elementami:



8. Zrzuty ekranu



Pierwszy klient zaczyna rysować:



Drugi klient zaczyna rysować: