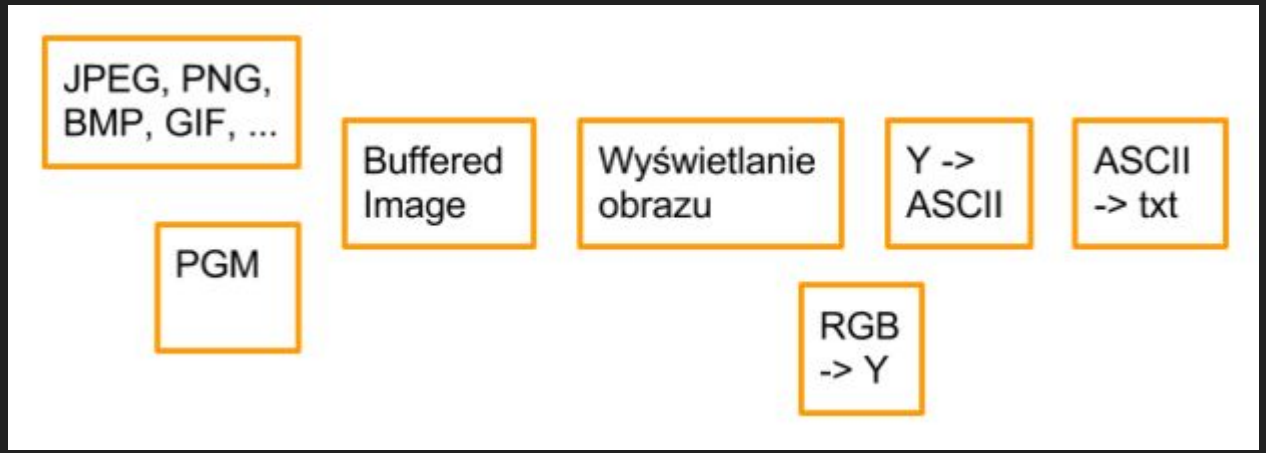
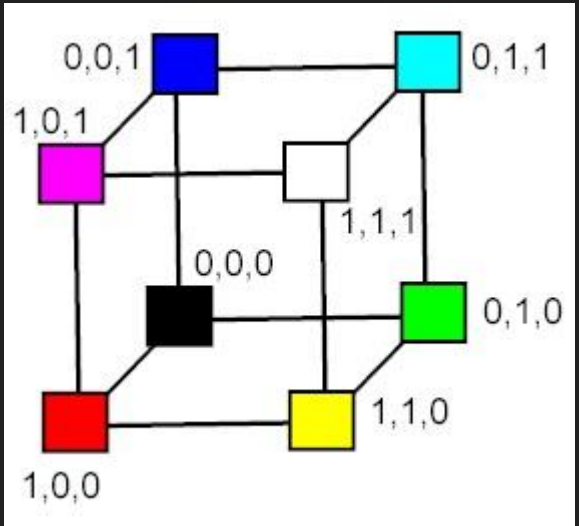
Platformy Programistyczne Java

Laboratorium 4

Projekt ASCII Art



RGB



- Trzy (lub cztery) kanały:
  - Red
  - Green
  - Blue
  - Alpha (Przezroczystość)
- Każdy kanał ma wartości z przedziału:
  - 0 .. 255 (int) lub 0.0 .. 1.0 (float / double)
- Wiele możliwych typów reprezentacji w BufferedImage .

Odcienie szarości

- Jeden kanał
- Wartości z przedziału:
  - 0 .. 255 (int) lub 0.0 .. 1.0 (float / double)
- Typ BufferedImage.TYPE_BYTE_GRAY
- Czasem trzy kanały, każdy o tej samej wartości

Konwersja RGB -> Odcienie szarości

Oparta o względną luminację kolorów

$$Y = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

Pobranie koloru piksela

```
import java.awt.Color; import java.awt.image.BufferedImage;
BufferedImage image = …;
Color color = new Color(image.getRGB(x, y));
color.getRed(); color.getGreen(); color.getBlue();
```

// uniezależniamy się od reprezentacji obrazu (ARGB, BGR, RGB, etc - niech Java się tym zajmie), operujemy na obiektach typu Color

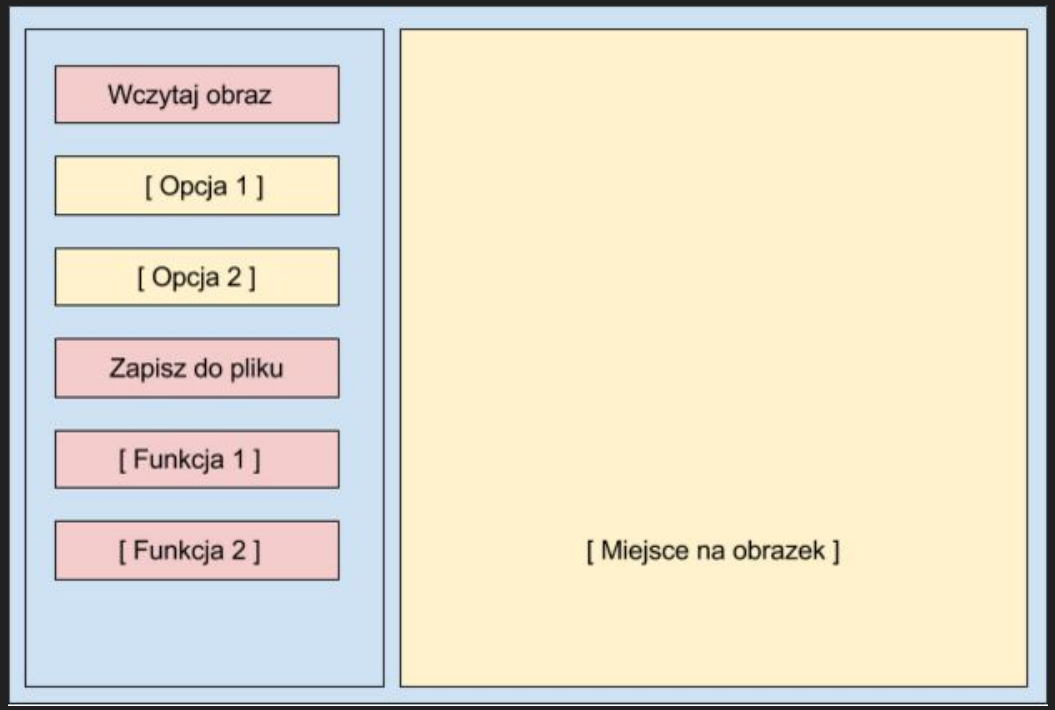
// ignorujemy alphę (przezroczystość), ale jest nam niepotrzebna

// dodatkowo ImageIO.read(File file) wczytuje pliki JPEG, PNG, GIF, etc.

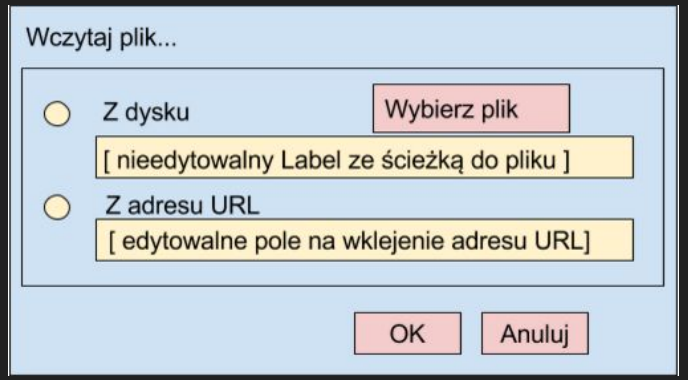
Zadanie 1 (na 3.0)

1. Przycisk “Zapisz do pliku” staje się aktywny po wczytaniu (dowolnego) obrazka, i jest nieaktywny jeżeli żaden obraz nie został wczytany.
2. Jeżeli wczytany plik jest obrazem kolorowym (typ obrazu inny niż BufferedImage.TYPE_BYTE_GRAY), to po naciśnięciu “Zapisz do pliku” następuje konwersja obrazu do odcieni szarości, następnie do znaków ASCII i ostatecznie zapis do pliku tekstowego. Jeżeli plik jest już w odcieniach szarości to nie należy go ponownie konwertować.

Projekt ASCII Editor’a



Okno główne aplikacji



Okno dialogowe

Opis oznaczeń

Zadanie 2 (na 4.0)

1. Jakość - [Opcja 1] (wg projektu graficznego), pozwala na wybór ilości poziomów szarości. Dostępne opcje:
   a. Niska
   b. Wysoka
2. “Niska” jakość oznacza konwersję do 10 znaków ASCII (co już mają Państwo zaimplementowane). “Wysoka” jakość oznacza konwersję do 70 znaków ASCII wg poniższego schematu (od czarnego do białego, ostatni znak to spacja, otwierający i zamykający cudzysłów nie należą do znaków):

```
"$@B%8&WM#*oahkbdpqwmZO0QLCJUYXzcvunxrjft/\|()1{}[]?-_+~<>i!lI;:,"^`'. "
```

// “Escapowanie” znaków robi się za pomocą \ np. \”

Opcja 2 - Skalowanie obrazu

```
private static BufferedImage resizeImage(BufferedImage
   originalImage, int type, int w, int h) {
  BufferedImage resizedImage = new BufferedImage(w,h,type);
  Graphics2D g = resizedImage.createGraphics();
  g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
                     RenderingHints.VALUE_INTERPOLATION_BILINEAR);
  g.drawImage(originalImage, 0, 0, w, h, null);
  g.dispose();
  return resizedImage;
}
```

Opcja 2 - Rozdzielczość ekranu

```
Dimension screenSize =
                Toolkit.getDefaultToolkit().getScreenSize();
double width = screenSize.getWidth();
double height = screenSize.getHeight();
```

Zadanie 3 (na 5.0)

1. Szerokość - [Opcja 2] (wg projektu graficznego), ma pozwalać na wybranie szerokości wynikowego ASCII Art. Dostępne są (co najmniej) 4 opcje:
   a. 80 znaków
   b. 160 znaków
   c. Szerokość ekranu
   d. Oryginalna (bez skalowania)

   Po wczytaniu obrazu, wybraniu szerokości w polu [Opcja 2] a następnie naciśnięciu “Zapisz do pliku”, obraz powinien zostać przeskalowany, z zachowaniem proporcji szerokość:wysokość, do żądanej szerokości, następnie przekonwertowany do znaków ASCII i zapisany do pliku tekstowego.

Ostateczna ocena

Zależy od:

- Ilości wykonanych zadań
- Jakości kodu