# Laboratorium 1: Wprowadzenie. Manipulowanie obrazem w Java.

Dzisiejsze ćwiczenia skupiają się na używaniu obiektów z klasy `BufferedImage`. Będą one wykorzystywane na kolejnych kilku laboratoriach i posłużą do obsługi zapisywania i odczytywania pikseli obrazów (które będziemy tworzyć za pomocą własnych algorytmów).

## Klasa `BufferedImage` dostarcza dwóch istotnych metod, którymi możemy modyfikować wartości pikseli:

```
public int getRGB(int x,int y)
```

Metoda zwraca kolor zapisany w systemie ARGB (TYPE_INT_ARGB) zakodowany na 32 bitach zmiennej int. Każdy z kanałów, ma zatem 8 bitów precyzji, co pozwala na zapisanie wartości od 0 do 255.

- Parametry:
  - x - współrzędna X piksela
  - y - współrzędna Y piksela

```
public void setRGB(int x, int y, int rgb)
```

Ustawia wskazany piksel obrazu na przekazaną jako parametr wartość. Piksel powienien być w domyślnym modelu kolorów, tj. TYPE_INT_ARGB.

- Parametry:
  - x - współrzędna X piksela
  - y - współrzędna Y piksela
  - rgb - wartość koloru piksela

Przyjrzyjmy się, jak wartości poszczególnych kanałów (przezroczystości, czerwonego, niebieskiego i zielonego) są przechowywane na jednej zmiennej typu int.

Mamy 4 kanały, po 8 bitów każdy, co daje dokładnie rozmiar zmiennej typu int. Wartości te można zmieścić na jednej zmiennej typu int używając operacji bitowych:

```
    (8-bit alpha)    (8-bit red)   (8-bit green)    (8-bit blue)
```


8 bitowe wartości są przesunięte w następujący sposób:

- kanał alpha w lewo o 24, zajmuje bity 31-24
- kanał czerwony w lewo o 16, zajmuje bity 23-16
- kanał zielony w lewo o 8, zajmuje bity 15-8
- kanał niebieski jest na pozycjach 7-0, nie ma potrzeby przesuwania.

Praktyczna realizacja ćwiczenia będzie opierała się o kod klasy App.java. Fragment odpowiedzialny za operacje na obrazie wygląda następująco:

```
// pobieramy szerokość i wysokość obrazów
int width = img.getWidth();
int height = img.getHeight();

/* pobieramy środkowy piksel */
int p = img.getRGB(width/ 2, height / 2);

// Odczytujemy wartosci kanalow przesuwajac o odpowiednia liczbe bitow w prawo, tak ab

int a = (p>>24) & 0xff;
int r = (p>>16) & 0xff;
int g = (p>>8) & 0xff;
int b = p & 0xff;

// Ustawiamy wartosci poszczegolnych kanalow na przykładowe liczby

a = 255;
r = 100;
g = 150;
b = 200;

// TODO: ustaw ponownie wartości kanałów dla zmiennej p
img.setRGB(width/ 2, height/ 2, p);
```

**Ćwiczenie 1.: Uruchom projekt korzystając z interfejsu gradle'a w linii poleceń, lub importując go do ulubionego IDE.**

**Ćwiczenie 2.: Uzupełnij linię kodu odpowiedzialną za zapis zmodyfikowanych kanałów do zmiennej p. Uruchom program i sprawdź rezultat.**

**Ćwiczenie 3.: Zaimplementuj metodę `allWhite()` , która przyjmuje obiekt klasy BufferedImage jako parametr i ustawia wszystkie jego piksele na kolor biały (255,255,255,255).**

**Ćwiczenie 4.: Zaimplementuj metodę `imgNegative()`, która przyjmuje obiekt klasy BufferedImage jako parametr i zmienia wartości pikseli tak, aby uzyskać negatyw obrazu. Negatyw będzie można uzyskać odejmując wartości poszczególnych kanałów od wartości maksymalnej (255).**