Laboratorium 6

Zrównoleglanie operacji

Studenci w ramach zajęć laboratoryjnych zapoznają się z podstawowymi mechanizmami zrównoleglania operacji na potokach realizowanych z wykorzystaniem Stream API.

Zadanie powinno być realizowane jako projekt oparty na narzędziu Apache Maven skonfigurowanym tak aby wykorzystywał platformę Java w wersji 11 lub wyższej. Należy zwrócić uwagę na poprawną identyfikację projektu oraz pakiet wykorzystane w aplikacji.

Do wykonania zadania należy przygotować zestaw przynajmniej 20 dużych obrazków, np. w formacie JPG.

Należy zaimplementować aplikację przetwarzającą zbiór obrazków za pomocą potoku operacji. Aplikacja jako argumenty startowe powinna przyjmować lokalizację katalogu zawierającego obrazki oraz lokalizację gdzie zostaną zapisane obrazki po przetworzeniu.

Aby pobrać listę plików z wybranego katalogu należy skorzystać z New File API:

```
List<Path> files;
Path source = Path.of("/tmp/input");
try (Stream<Path> stream = Files.list(source))
    files = stream.collect(Collectors.toList());
} catch (IOException) {
}
```

Uwaga: Listę plików należy obraz do kolekcji i dopiero z niej utworzyć strumień na potrzeby zrównoleglonego potoku. Nie należy zrównoleglać strumienia pobranego za pomocą metody `Files.list`.

Zaimplementowany potok powinien składać się z następujących operacji:

- pobranie równoległego strumienia z kolekcji,
- odwzorowanie strumienia ścieżek (`Path`) na strumień par zawierających nazwę pliku oraz wczytany obrazek,
- odwzorowanie strumienia part na strumień par zawierających nazwę pliku oraz obrazek po przekształceniach,
- wykonanie operacji na każdej parze zapisującej obrazek po przekształceniach pod zadaną nazwą do katalogu wyjściowego.

Jako implementacji pary można wykorzystać klasę `Pair` z biblioteki Apache Commons Lang:

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.12.0</version>
</dependency>
```

```
Path path = …;
BufferedImage image = …;
String name = path.getFileName();
Pair<String, BufferedImage> pair = Pair.of(name, image);
```

Wczytanie obrazku można zrealizować za pomocą klasy ImageIO:

```
Path path = Path.of("/tmp/image.png");
BufferedImage image = ImageIO.read(path.toFile);
```

Stworzenie nowego obrazka w pamięci na podstawie oryginalnego:

```
BufferedImage original = ...
BufferedImage image = new BufferedImage(original.getWidth(),
                                        original.getHeight(),
                                        original.getType());
```

Operacje na pojedynczych pikselach:

```
BufferedImage original = ...
BufferedImage image = …
for (int i = 0; i < original.getWidth(); i++) {
    for (int j = 0; j < original.getHeight(); j++) {
        int rgb = original.getRGB(i, j);
        image.setRGB(i, j, rgb);
    }
}
```

W ramach zadania aby uzyskać obrazek wyjściowy należy dokonać przekształcenia kolorów. Rodzaj przekształcenia określa prowadzący. Przykład zamiany składowej zielonej z niebieską:

```
BufferedImage original = …
int i = …;
int j = …;
int rgb = original.getRGB(i, j);
Color color = new Color(rgb);
int red = color.getRed();
int blue = color.getBlue();
int green = color.getGreen();
Color outColor = new Color(red, blue, green);
int outRgb = outColor.getRGB();
```

Aby sprawdzić czas wykonania programu można skorzystać z:

```
long time = System.currentTimeMillis();
…
System.out.println(System.currentTimeMillis() - time);
```

Należy zrealizować następujące zadania:

1. Implementacja potoku przetwarzającego strumień obrazków z zadanego katalogu. (3 pkt)
2. Zrównoleglenie potoku i uruchomienie go w ramach własnej puli wątków. Przetestowanie jak zmienia się czas wykonania aplikacja w zależności od rozmiaru puli. (2 pkt)