Platformy Technologiczne
Laboratorium 3
Java: Gniazdka

Należy zaimplementować dwie aplikacje (nadawca i odbiorca) pozwalające na wysyłanie wybranego pliku poprzez sieć, za pomocą gniazdek połączeniowych.

Aplikacja nadawcy musi być zrealizowana jako aplikacja JavaFX. Wybieranie pliku do wysłania powinno być zrealizowane za pomocą komponentu ***DirectoryChooser***. Samo wysyłanie pliku musi odbywać się w osobnym wątku, którego implementacja będzie dziedziczyć po klasie ***Task***. Postęp wysyłania pliku musi być prezentowany przez kontrolkę ***ProgressBar***, której własność ***progressProperty*** należy powiązać z odpowiednią własnością zaimplementowanego wątku (***progressProperty***). Informacja o tym co aktualnie realizuje zaimplementowany wątek powinna być wyświetlona za pomocą komponentu Label, którego własność **textProperty** powinna być powiązana z odpowiednią własnością wątku (**messageProperty**).

Przykład:

```
@FXML
private Label label;

@FXML
private ProgressBar progressBar;
```

```
Task<Void> task = new Task<>();
label.textProperty().bind(task.messageProperty());
progressBar.progressProperty().bind(task.progressProperty());
```

Logika odpowiedzialna za wysyłanie pliku powinna zostać z w metodzie ***call*** zdefiniowanej przez klasę ***Task***.

```
public class MagesSendWorker extends Task<Void> {
	@Override
	protected Void call() throws Exception {
		return null;
	}
}
```

Aktualizacje postępu w wysyłaniu pliku oraz stanu co robi zaimplementowany wątek powinny się dobywać za pomocą metod ***updateMessage*** i ***updateProgress*** zdefiniowanych w klasie ***Task***.

Nawiązywanie połączenia z aplikacją odbiorcy powinna być realizowana za pomocą instancji klasy ***Socket***.

```
Socket socket = new Socket("localhost", 9876);
```

Aplikacja odbiorcy może być zrealizowana jako zwykła aplikacja Java SE bez graficznego interfejsu użytkownika. Aplikacja musi pozwalać na odbieranie wielu plików równolegle w osobnych wątkach. Nasłuchiwanie na nowe połączenia musi być realizowane poprzez obiekt klasy ***ServerSocket*** oraz metodę ***accept*** zwracającą nowy obiekt typu ***Socket*** będącym zestawionym połączeniem z aplikacją

nadawcy. Uzyskany obiekt typu ***Socket*** musi zostać przesłany do nowego wątku który zajmie się jego obsługą. Główny wątek aplikacji odbiorcy może zajmować się jedynie przyjmowaniem przychodzących połączeń. Wątki obsługi nadawców powinny implementować interfejs **Runnable** i być uruchomione za pomocą obiektu klasy **Thread**.

Przykład:

```
ServerSocket server = new ServerRocket(9876);
while (true) {
	Socket socket = server.accept();
}
```

```
Runnable runnable = new Runnable();
Thread thrad = new Thread(runnable);
thread.start();
```

Zapis/odczyt do/z pliku powinien być realizowany przez strumienie bajtowe (programy muszą pozwalać na przesyłanie dowolnych plików, również binarnych). Należy do tego wykorzystać odpowiednio klasy ***FileOutputStream*** i ***FileInputStream***. W przypadku korzystania ze strumieni bajtowych należy utworzyć odpowiedni bufor będący w praktyce tablicą typu ***byte***. Do odczytu danych ze strumieni binarnych wykorzystuje się metodę ***read(byte[] buffor)*** zwracającą liczbę przeczytanych bajtów. Zwrócona liczba może być mniejsza niż rozmiar buforu w przypadku gdy danych na strumieniu było mniej niż zdefiniowany rozmiar. Zapis do strumieni bajtowych realizuje się poprzez metodę ***write(byte[] buffor, int offset, int size)*** pozwalającą na zapisanie całego bufora lub jego części poprzez odpowiednie ustawienie parametrów ***offset*** i ***size***.

Przykład:

```
byte[] buffor = new byte[100];
FileInputStream fis = new FileInputStream("file");
int dataSize;
while ((dataSize = fis.read(buffor)) > -1) {
	//...
}
```

```
FileOutputStream fos = new FileOutputStream("file");
fos.write(buffor, 0, buffor.length;
```

Czytanie/pisanie z/do gniazdek realizuje się analogicznie poprzez strumienie bajtowe ***InputStream*** i ***OutputStream***, które mogą być pobrane z obiektu klasy ***Socket***.

Przykład:

```
InputStream is = socket.getInputStream();
OutputStream os = socket.getOutputStream();
```

Wszystkie strumienie muszą być poprawnie obsłużone i zamknięte. Można przy tym korzystać ze starej składni wykorzystującej interfejs Closeable lub nowej wykorzystującej interfejs AutoCloseable.

Przykład:

```
Closeable c;
try {
	Closeable c = ...;
	...
} catch (Exception ex) {
	...
} finally {
	if (c != null) {
		try {
			c.close();
		} catch (Exception ex) {
			...
		}
	}
}
```

```
try (AutoCloseable ac = ...) {
	...
} catch (Exception ex) {
	...
}
```

Punktacja:

- wysyłanie i odbieranie pojedynczego pliku: 2 pkt,
- odbieranie dowolnej liczby plików: 2 pkt,
- poprawna aktualizacja kontrolek ***ProgressBar*** i ***Label***: 1 pkt.