POLITECHNIKA GDAŃSKA

Platformy Technologiczne

Laboratorium 3

Gniazdka Sieciowe

Studenci w ramach zajęć laboratoryjnych zapoznają się z podstawowymi mechanizmami obsługi wejścia-wyjścia oraz komunikacji sieciowej za pomocą gniazdek.

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.

Należy zaimplementować aplikację w trybie tekstowym pozwalającą na komunikację sieciową dwóch niezależnie uruchamianych komponentów (serwera i klienta). W ramach aplikacji powinny się znaleźć dwie klasy zawierające metodę `main`, odpowiednio dla klienta i serwera.

Komunikacja między klientem a serwerem powinna być oparta na zdefiniowanym protokole binarnym. Protokół jest definiowany przez prowadzącego na zajęciach (przykład na końcu instrukcji). Serwer powinien poprawie obsługiwać połączenia od wielu klientów za pomocą wątków. Jeśli wymagane, dane od użytkownika po stronie klienta można pobrać np.: za pomocą klasy `Scanner`. Serwer nie powinien pobierać żadnych informacji od użytkownika w czasie komunikacji. Kolejne etapy komunikacji powinny być wyświetlane za pomocą mechanizmu logowania.

Należy zrealizować następujące zadania:

1. Komponent serwera. Poprawna implementacja nasłuchiwania na połączenia oraz delegowania ich obsługi do nowych wątków (1 pkt)
2. Komponent klienta. Implementacja nawiązywania połączenia po stronie klienta. (1 pkt)
3. Poprawna implementacja zadanego protokołu. (3 pkt)

Uwaga: W przypadku wykorzystywania strumienie `ObjectOutputStream` i `ObjectInputStream` na strumieniach pobranych z gniazdka sieciowego należy zwrócić uwagę aby w pierwszej kolejności stworzyć `ObjectOutputStream`.

Przykład:

Klient:	Serwer:
	wysyła obiekt `String` o stałej wartości *ready*
wysyła obiekt `Integer` o wartości `n` podanej przez użytkownika	
	wysyła obiekt `String` o stałej wartości *ready for messages*
wysyła `n` obiektów klasy `Message` jako `n` kolejnych obiektów (nie mylić z tablicą lub kolekcją)	
	wysyła obiekt `String` o stałej wartości *finished*

`Message.java:`

```
public class Messsage implements Serialization {

     private int number;

     private String content;

     //setters and getters

}
```