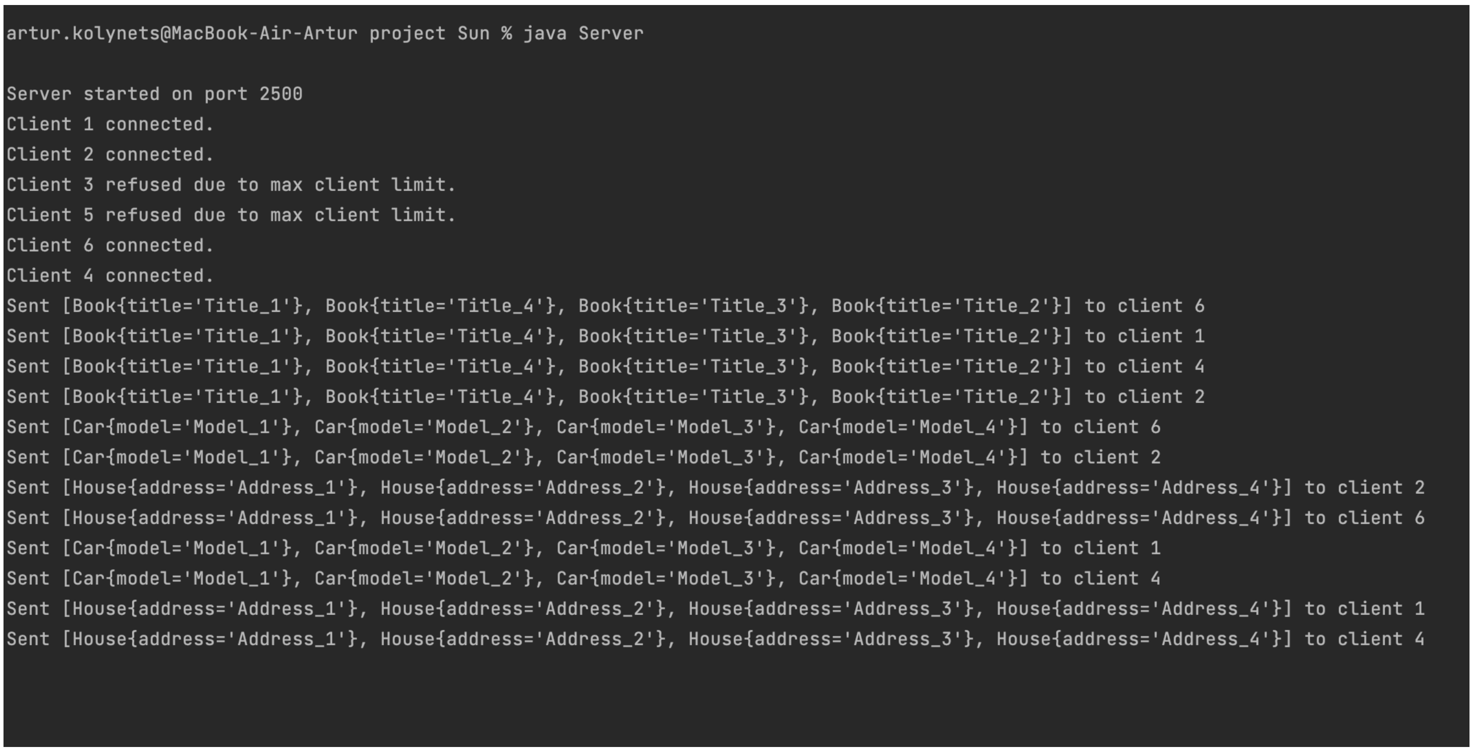
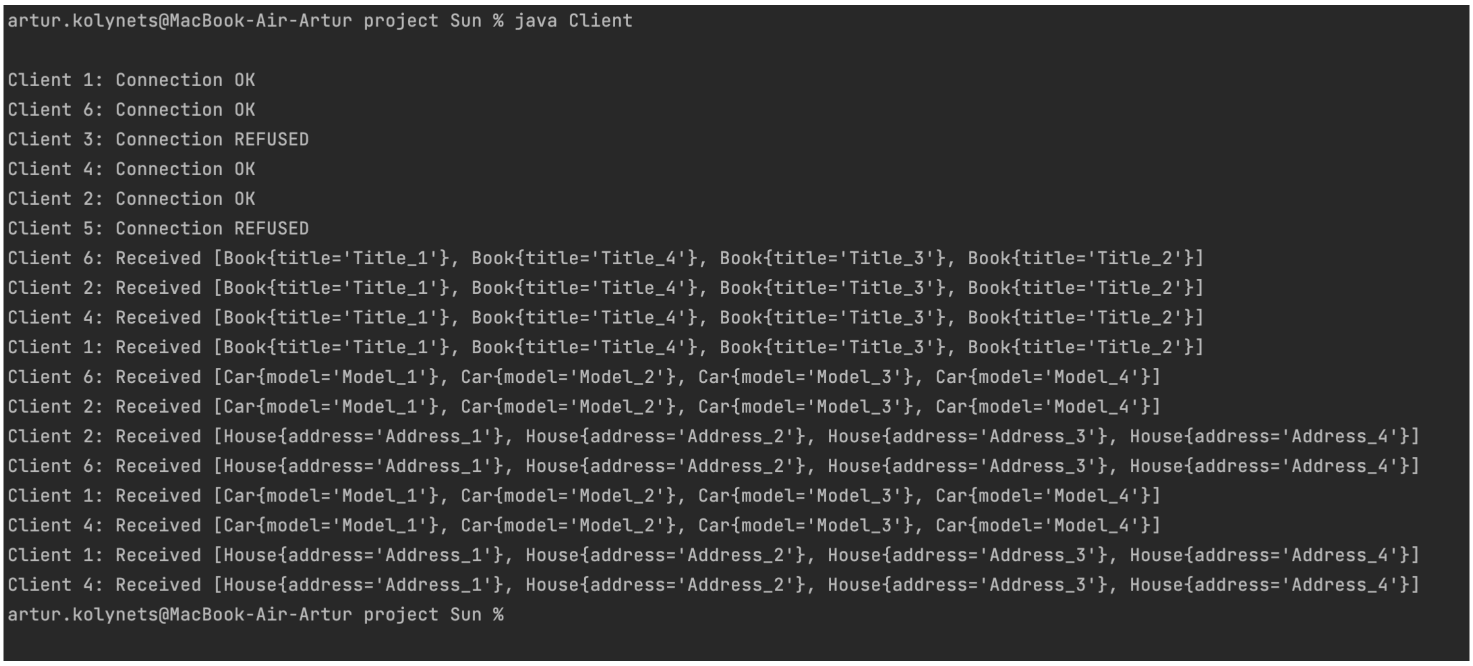
Project Java SUN

Artur Kolynets nr. 161579
Vladyslav Haitko nr. 159857



```
artur.kolynets@MacBook-Air-Artur project Sun % java Server

Server started on port 2500
Client 1 connected.
Client 2 connected.
Client 3 refused due to max client limit.
Client 5 refused due to max client limit.
Client 6 connected.
Client 4 connected.
Sent [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}] to client 6
Sent [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}] to client 1
Sent [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}] to client 4
Sent [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}] to client 2
Sent [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}] to client 6
Sent [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}] to client 2
Sent [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}] to client 2
Sent [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}] to client 6
Sent [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}] to client 1
Sent [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}] to client 4
Sent [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}] to client 1
Sent [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}] to client 4
```



```
artur.kolynets@MacBook-Air-Artur project Sun % java Client

Client 1: Connection OK
Client 6: Connection OK
Client 3: Connection REFUSED
Client 4: Connection OK
Client 2: Connection OK
Client 5: Connection REFUSED
Client 6: Received [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}]
Client 2: Received [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}]
Client 4: Received [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}]
Client 1: Received [Book{title='Title_1'}, Book{title='Title_4'}, Book{title='Title_3'}, Book{title='Title_2'}]
Client 6: Received [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}]
Client 2: Received [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}]
Client 2: Received [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}]
Client 6: Received [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}]
Client 1: Received [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}]
Client 4: Received [Car{model='Model_1'}, Car{model='Model_2'}, Car{model='Model_3'}, Car{model='Model_4'}]
Client 1: Received [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}]
Client 4: Received [House{address='Address_1'}, House{address='Address_2'}, House{address='Address_3'}, House{address='Address_4'}]
artur.kolynets@MacBook-Air-Artur project Sun %
```

Opis programu

Aplikacja została zaprojektowana jako model klient-serwer, który ilustruje podstawowe operacje

związane z obsługą wielu klientów jednocześnie. Program spełnia następujące założenia:
Założenia funkcjonalne:

1. **Definicja prostych klas**:
   - Aplikacja definiuje trzy proste klasy: Book, Car i House, z których każda zawiera jedno pole (title, model i address) oraz konstruktor inicjalizujący to pole.
2. **Tworzenie obiektów na serwerze**:
   - Serwer przy starcie tworzy po cztery obiekty każdej klasy, inicjalizując je różnymi danymi. Obiekty te są przechowywane w mapie, gdzie kluczem jest nazwa klasy oraz numer porządkowy, np. dla klasy Book i dwóch instancji mamy: book_1, book_2.
3. **Obsługa wielu klientów**:
   - Serwer może obsługiwać wielu klientów jednocześnie, ograniczając ich maksymalną liczbę do 4. Klienci przekraczający ten limit otrzymują informację o odmowie obsługi.
4. **Komunikacja klient-serwer**:
   - Klient łączy się z serwerem, wysyła swoje id i otrzymuje status połączenia: OK lub REFUSED.
   - W przypadku odpowiedzi REFUSED, klient kończy działanie.
   - W przypadku odpowiedzi OK, klient prosi serwer o przesłanie kolekcji obiektów konkretnej klasy (np. get_books). Serwer wysyła odpowiednie obiekty do klienta w postaci zserializowanej.
   - Klient odbiera kolekcję, wypisuje jej zawartość na konsoli i powtarza operację dla różnych klas obiektów trzy razy, po czym kończy działanie.

Przykład działania:

1. **Uruchomienie serwera**:
   - Serwer zostaje uruchomiony na porcie 2500 i tworzy po cztery obiekty dla każdej z klas (Book, Car, House).
2. **Klienci próbują się połączyć**:
   - Sześciu klientów próbuje nawiązać połączenie z serwerem. Pierwszych czterech klientów jest akceptowanych, a pozostali otrzymują informację REFUSED.
3. **Wymiana danych**:
   - Akceptowani klienci wysyłają żądania o obiekty konkretnej klasy (np. book_1, car_2, house_3). Serwer odpowiada, wysyłając listy obiektów zgodnych z żądaniem.
   - Klienci wypisują otrzymane obiekty na konsoli i powtarzają tę operację trzy razy dla różnych klas obiektów, a następnie kończą działanie.

Podsumowanie

Program demonstruje podstawowe zasady komunikacji klient-serwer, obsługi wielu klientów równocześnie oraz zarządzania obiektami i ich serializacją. Dzięki wprowadzeniu losowych opóźnień przy obsłudze klientów, aplikacja pokazuje, jak serwer radzi sobie z równoczesnymi żądaniami, co jest kluczowe dla zrozumienia rzeczywistych systemów rozproszonych.