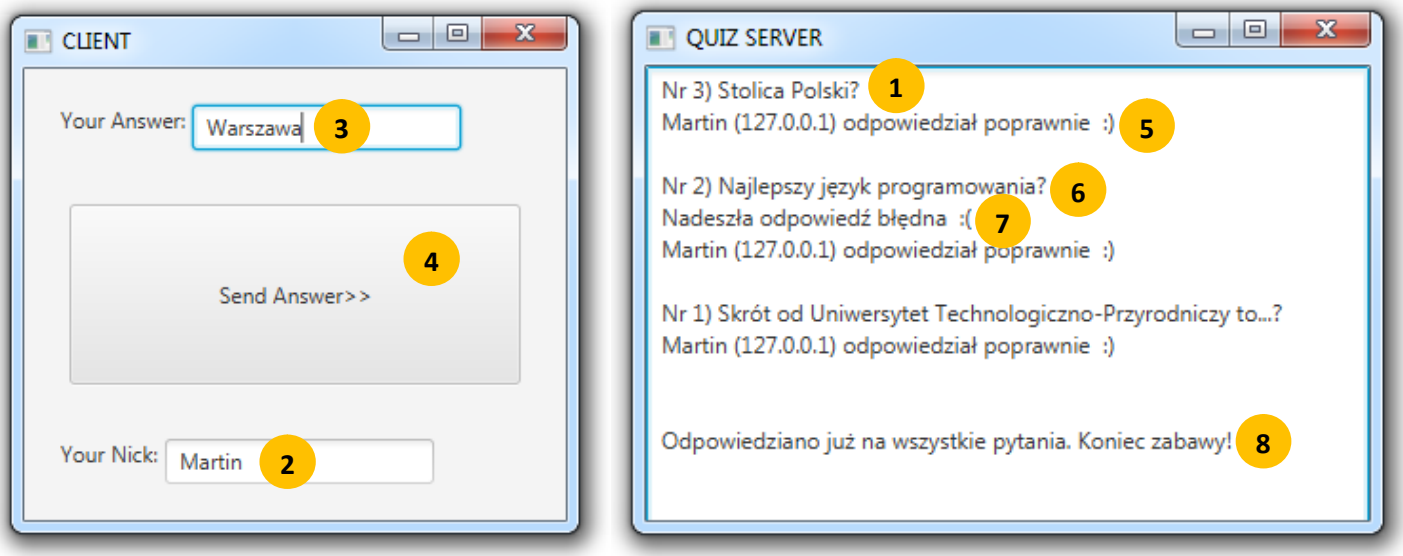
ĆWICZENIE: APLIKACJA TYPU CLIENT SERWER Z WYKORZYSTANIEM GNIAZD TCP/IP ORAZ WIELOWĄTKOWOŚCI W JAVIE (4-6 godzin zajęciowych)

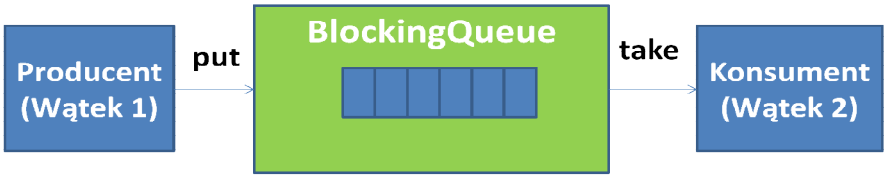
Napisać aplikacje okienkowe typu serwer oraz klient (z wykorzystaniem bilbioteki Swing lub JavaFX) **do prowadzania prostych quizów wśród lokalnej grupy uczestników**. Do komunikacji należy wykorzystać protokół TCP/IP oraz gniazda, a do przechowywania nadesłanych odpowiedzi tzw. kolejkę blokującą (dodatek A). Okienko serwera powinno być wyświetlane w sali zajęciowej (np. przez projektor multimedialny), a wszyscy uczestnicy zabawy odpowiadać w czasie rzeczywistym za pomocą aplikacji klienta na wyświetlane pytania. Liczy się zawsze pierwsza poprawna nadesłana odpowiedź. W aplikacjach należy spełnić następujące wymagania:



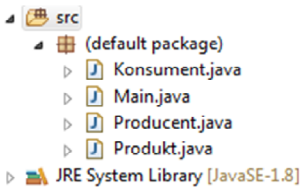
- ✓ Serwer i klient powinny stanowić osobne aplikacje, a komunikacja odbywać się tylko od klienta do serwera.
- ✓ Po uruchomieniu serwera, pytania z pliku testowego powinny zostać wczytane do odpowiedniej kolekcji np. mapy, która pozwoli na łatwe sprawdzanie nadesłanych odpowiedzi.
- ✓ Format krótkich pytań/odpowiedzi dla każdej linii w pliku tekstowym może być następujący: *Pytanie?*|*Odpowiedź*. Liczba linii w tym pliku będzie odpowiadać wtedy liczbie pytań wyświetlanych podczas zabawy, a separator „|" będzie oddzielać pytania od odpowiedzi.
- ✓ Serwer po uruchomieniu automatycznie wyświetla pierwsze pytanie (1) i zaczyna nasłuchiwać na odpowiednim porcie nadchodzących odpowiedzi.
- ✓ W aplikacji klienta należy podać dane gracza (2), a odpowiedź na wyświetlane pytanie zostanie wysłana po wpisaniu tej odpowiedzi przez gracza (3) i kliknięciu przycisku *Send Answer>>* (4).
- ✓ Przetwarzanie nadesłanych odpowiedzi powinna odbywać się z wykorzystaniem modelu Producent-Konsument, którzy działają na osobnych wątkach, z wykorzystaniem kolejki blokującej. Przykład takiego rozwiązania został zamieszczony **w dodatku A** poniżej.
- ✓ Producent uruchomiony na osobnym wątku po stronie serwera powinien zajmować się **tylko** nasłuchiwaniem odpowiedzi na odpowiednim porcie oraz umieszczać nadesłane odpowiedzi w kolejce blokującej.
- ✓ Konsument uruchomiony na osobnym wątku po stronie serwera powinien analizować pobierane z kolejki blokującej odpowiedzi i sterować całą aplikacją. W przypadku nadesłania poprawnej odpowiedzi wyświetlana jest informacja o graczu, który był pierwszy (5), oczyszczana jest kolejka blokująca z wszystkich innych nadesłanych odpowiedzi (które stają się już nieaktualne) i pojawia się następne pytanie (6). W przypadku nadsyłania niepoprawnych odpowiedzi wyświetlany jest stosowny komunikat (7). Zabawa kończy się (8) po nadesłaniu poprawnych odpowiedzi na wszystkie pytania zdefiniowane w pliku.

DODATEK A

Współbieżność/BlockingQueue



Współbieżność/BlockingQueue 1 z 4



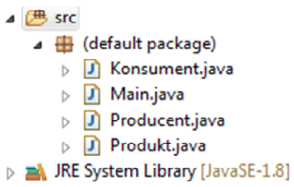
Produkt.java

```
public class Produkt  {
    private String nazwa;

    public Produkt(String nazwa){
      this.nazwa=nazwa;
    }
    public String getProdukt () {
      return this.nazwa;
    }
}
```

Producent.java

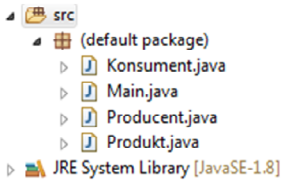
Współbieżność/BlockingQueue 2 z 4




```
import java.util.concurrent.BlockingQueue;
public class Producent implements Runnable {
  private BlockingQueue<Produkt> kolejka;
  private Produkt produkt;
  Producent(BlockingQueue<Produkt> kolejka){
   this.kolejka=kolejka;
  }
  public void run() {
     for(int i = 0; i <= 10; i++) {//produkcja zadań
        produkt = new Produkt("Produkt "+i);
        try {
              Thread.sleep(500);
              kolejka.put(produkt);
              System.out.println("Producent wykonał „
                                  + produkt.getProdukt()+"!");
        } catch (InterruptedException e) {}
      }//for
      produkt = new Produkt("brak");
      try {
            kolejka.put(produkt);
      } catch (InterruptedException e) {}
  }
}
```

Współbieżność/BlockingQueue 3 z 4

Konsument.java

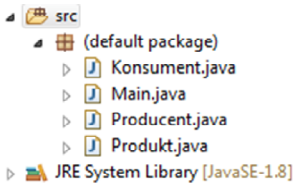



```
import java.util.concurrent.BlockingQueue;
public class Konsument implements Runnable {
  private BlockingQueue<Produkt> kolejka;
  private Produkt produkt;

  Konsument(BlockingQueue<Produkt> kolejka){
   this.kolejka=kolejka;
  }

   public void run() {
     try{
        while(!(produkt = kolejka.take()).getProdukt().equals("brak")){
        Thread.sleep(1500);
        System.out.println("*Konsument odebrał "
                                          +produkt.getProdukt()+"*");
        }
      }catch(InterruptedException e) {}
          System.out.println("\nProdukcja zakończona, bye!");
   }
}
```

Współbieżność/BlockingQueue 4 z 4



Main.java

```
import java.util.concurrent.*;

public class Main {
  public static void main(String[] args) {

          BlockingQueue<Produkt> kolejka = new ArrayBlockingQueue<>(2);
          Producent producent = new Producent(kolejka);
          Konsument konsument = new Konsument(kolejka);

          new Thread(producent).start();
          new Thread(konsument).start();
          System.out.println("Producent i Konsument uruchomieni!");
  }//main
}//Main
```