Zaprojektuj rozproszony system służący do definiowania, rejestrowania i wykonywania zadań obliczeniowych, w którym do komunikacja zostaną wykorzystane gniazda TCP/IP (a dokładniej: klasy `Socket` oraz `SocketServer`).

W systemie powinny być zaimplementowane następujące klasy:

- `Rejestrator` – odpowiedzialna za przyjmowanie zadań od `Interesantów` i wydawanie zadań `Egzekutorom`. W klasie tej powinna istnieć jakaś struktura danych do przechowywania zadań. Każde zadanie zgłoszone przez `Interesanta` powinno trafiać do tej struktury i uzyskiwać kolejny numer (identyfikator). W strukturze tej powinno być również przewidziane miejsce na przechowywanie wyników zadań oraz ich statusów. Wyniki będą dostarczane przez `Egzekutorów`, statusy zaś mają odpowiadać stanom: zadanie zgłoszone, zadanie pobrane do wykonania, zadanie wykonane, brak zadania, błąd (sposób implementacji statusów – dowolny). Klasa `Rejestrator` ma posiadać `SocketServer` (cały czas nasłuchuje).
- `Egzekutor` – odpowiedzialna za pobieranie zadań z `Rejestratora`, wykonywanie zadań i zgłaszanie ich wyników `Rejestratorowi`. Klasa `Egzekutor` ma posiadać `Socket` (do komunikacji dwustronnej z `Rejestratorem`).
- `Interesant` – odpowiedzialna za tworzenie zadań i zgłaszanie ich do `Rejestratora`. Klasa `Interesant` służyć może do odpytywania `Rejestratora` o wynik oraz usuwania zgłoszonego wcześniej zadania. Klasa`Interesant` ma posiadać `Socket` (do komunikacji dwustronnej z `Rejestratorem`).

Zakłada się, że zbiór obsługiwanych zadań będzie ograniczony (przynajmniej 2 zadania), np.:

Nazwa zadania	Parametry	Wynik
suma	2.4,3.4,5.2 (tj. ciąg liczb)	9.0 (tj. liczba)
rowKwadr	1.0,0.0,-1.0 (tj. współczynniki a,b,c równ. kw.)	1.0,-1.0 (tj. pierwiastki równ. kw.)

Komunikacja pomiędzy `Rejestratorem` a `Interesantem` i `Egzekutorem` powinna odbywać się z wykorzystaniem następującego protokołu (zobacz rysunek na końcu):

- operacja zgłoszenia zadania:
  - `Interesant` wysyła do `Rejestratora` ciąg znaków:
    `r:nazwaZadania, par1, par2, ...`
  - `Rejestrator` odpowiada ciągiem znaków:
    `n:numerZadania`
- operacja pobrania wyniku (zmienia status zadania):
  - `Interesant` wysyła do `Rejestratora` ciąg znaków:
    `p:numerZadania`
  - `Rejestrator` odpowiada ciągiem znaków:
    `w:wynikZadania` (jeśli `Egzekutor` dostarczył wynik) albo `s:Status` (jeśli `Egzekutor` nie dostarczył jeszcze wyniku działania)
- operacja usunięcie zadania (zmienia status zadania):
  - `Interesant` wysyła do `Rejestratora` ciąg znaków:
    `u:numerZadania`
  - `Rejestrator` odpowiada ciągiem znaków:
    `s:Status`

- operacja pobrania zadania (pobierane jest pierwsze nieobsłużone jeszcze zadanie, pobranie zadania zmienia jego status):
    - `Egzekutor` wysyła do `Rejestratora` ciąg znaków:
      `p:`
    - `Rejestrator` odpowiada ciągiem znaków:
      `z:numerZadania,nazwaZadania,par1,par2,...`
- operacja zgłoszenia wyniku:
    - `Egzekutor` wysyła do `Rejestratora` ciąg znaków (podobnie do parametrów w szczególnym przypadku może być cały ciąg wyników):
      `w:numerZadania,wynik`
    - `Rejestrator` odpowiada ciągiem znaków:
      `s:status`

Uwaga:

System powinien być zabezpieczony przed problemami wielowątkowości (np. jeśli `Rejestrator` równocześnie otrzyma żądanie usunięcia zadania i wykonania zadania).

Uruchomienie systemu powinno polegać na wystartowaniu kilku aplikacji (osobna aplikacja korzystająca z `Rejestratora`, osobne aplikacje korzystające z `Interesantów` i `Egzekutorów`). Wszystkie aplikacje powinny oferować GUI, na którym można będzie wprowadzić parametry połączeń (host, port), wprowadzać inne parametry (jak: wybór operacji do wykonania, parametry operacji) oraz wizualizować wyniki.

Można w widoku Rejestratora pokazywać zadania razem z ich statusami.

Wykonanie zadania w `Egzekutorze` można wyzwalać poprzez naciśnięcie przycisku (dzięki czemu można będzie przetestować zabezpieczenie przed próbą usunięcia aktualnie wykonywanego zadania).

Wszystkie nieopisane reguły (np. co zrobić z usuniętym zadaniem – zmienić tylko jego status czy też całkowicie usunąć z `Rejestratora`) należy samemu dodefiniować.