W niniejszym zadaniu musimy zrefaktoryzować kod istniejącej aplikacji. W zadaniu po otwarciu solucji zobaczymy dwa projekty:

- LegacyApp - to jest aplikacja, którą będziemy chcieli zrefaktoryzować.
- LegacyAppConsumer - to jest przykład aplikacji, która wykorzystuje LegacyApp.

Pamiętajmy, że refaktoryzacja polega na tym, że nie zmieniamy działania istniejącej aplikacji. Zakładamy, że aplikacja działa poprawnie. Mamy zrefaktoryzować klasę UserService wraz z metodą AddUser.
Uwaga: w LegacyApp znajdziecie klasy, które symulują odpytywanie zewnętrzny źródeł danych poprze użycie Thread.Wait.

- Pamiętaj, że aplikacja ma działać tak samo jak teraz po procesie refaktoryzacji.
- W trakcie refaktoryzacji możesz modyfikować dowolne pliki w LegacyApp oprócz klasy UserDataAccess. Ta klasa reprezentuje przykład spadkowej biblioteki, której z różnych powodów nie możemy edytować.
- Pamiętać, że kod w aplikacji LegacyAppConsumer musi się cały czas kompilować i działać - również po procesie refaktoryzacji. Nie chcemy przecież, żeby po naszej refaktoryzacji kod innych aplikacji nagle przestał działać. Nie możemy również kodu z tego projektu w żaden sposób modyfikować.
- Kieruj się zasadami SOLID, testowalnością kodu i jego czytelnością.
- Staraj się zapanować nad strukturę programu pamiętając o metrykach cohesion i coupling.
- Postaraj się wykorzystać w rozwiązaniu testy jednostkowe.