

**Dokumentacja projektu**

Przedmiot: **Programowanie obiektowe**

**Tytuł projektu:**

**Gra Typu Quiz**

**Prowadzący:** **Wykonawca:**

mgr inż. Ewa Żesławska Piotr Marzec

w64153

3IIZ/2020-GP02

Rzeszów 2022

Spis treści

[1. Opis Założeń 3](#_Toc95521162)

[2. Specyfikacja wymagań 3](#_Toc95521163)

[2.1 Wymagania funkcjonalne 3](#_Toc95521164)

[2.2 Wymagania niefunkcjonalne 3](#_Toc95521165)

[3. Diagram Przypadków Użycia 4](#_Toc95521166)

[4. Harmonogram Realizacji Projektu 4](#_Toc95521167)

[5. Opis techniczny projektu 5](#_Toc95521168)

[6. Prezentacja warstwy użytkowej 12](#_Toc95521169)

[7. Raporty z testów jednostkowych 13](#_Toc95521170)

[8. System kontroli wersji 14](#_Toc95521171)

[9. Dokumentacja projektu 15](#_Toc95521172)

[10.  Literatura 15](#_Toc95521173)

# Opis Założeń

Nasza wiedza sprawdzana jest na każdym etapie naszego życia, czy to początkowo w szkole, czy później w pracy zawodowej. Nawet oglądając telewizje, możemy natknąć się na różnego rodzaju teleturnieje. Prowadzący zadaje w nich pytania uczestnikom, a my czasami mamy wrażenie, że wiemy więcej od nich. Właśnie dlatego powstała ta prosta gra typu QUIZ. Odpowiadamy w niej na 10 krótkich pytań o różnym poziomie trudności, udzielając jednej z czterech możliwych odpowiedzi. Możemy sprawdzić się nie tylko sami, ale też wraz ze znajomymi – a to dzięki prostemu rankingowi, który podlicza nasze punkty. Gra zawiera bardzo prosty interfejs konsolowy dzięki któremu odnajdą się w nim nawet osoby mniej zaznajomione z obsługą komputera.

# Specyfikacja wymagań

## Wymagania funkcjonalne

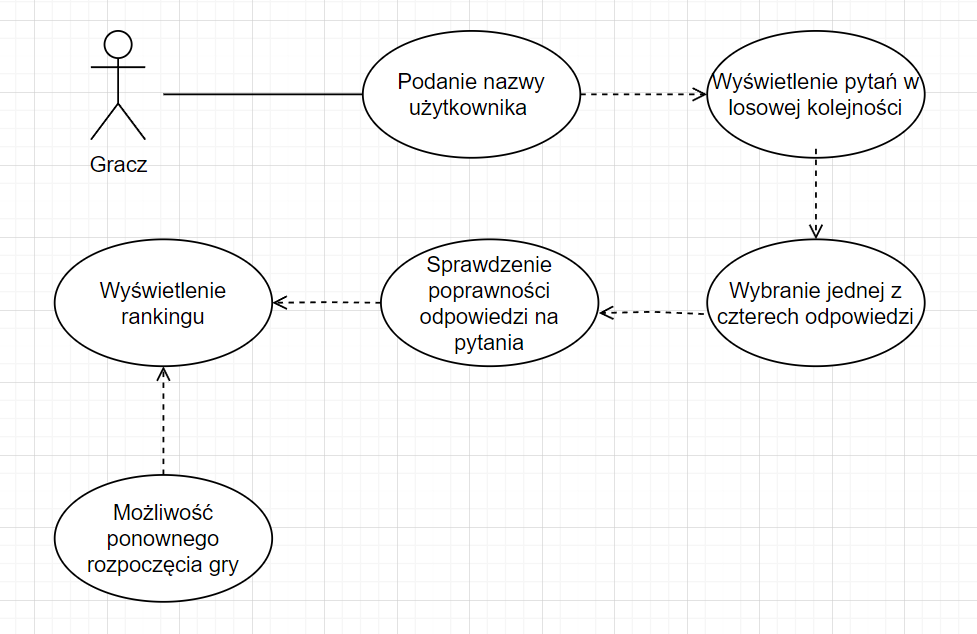
* Użytkownik podaje swoje imię/nick
* Pytania wyświetlają się w losowej kolejności
* Użytkownik ma do wyboru cztery odpowiedzi na każde pytanie
* Sprawdzenie poprawności odpowiedzi
* Użytkownik na końcu wyświetla ranking zawierający najlepsze wyniki
* Użytkownik ma możliwość rozpoczęcia ponownej gry

## Wymagania niefunkcjonalne

* Aplikacja na system Windows
* Aplikacja jest prosta i możliwie przyjazna dla użytkownika
* Aplikacja tworzona w języku Java w środowisku IntelliJ
* Aplikacja nie potrzebuje dostępu do Internetu

# Diagram Przypadków Użycia

Poniżej przedstawiam diagram przypadków użycia



Rys. Diagram przypadków użycia

# Harmonogram Realizacji Projektu

Poniżej przedstawiam harmonogram realizacji projektu w formie wykresu Gantta.

Rys. Harmonogram realizacji projektu

# Opis techniczny projektu

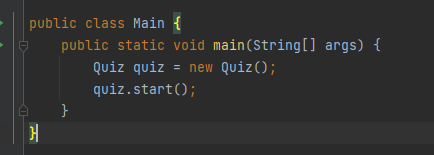
Gra typu QUIZ została napisana w języku Java 17, a jej kod jest podzielony na kilka klas:

* Main – klasa wywołująca program
* Question – klasa odnosząca się do pytań zawartych w grze
* Quiz – jest to „serce” programu, to ona w głównej mierze odpowiada za rozgrywkę
* QuizFileReader – klasa dotycząca odczytywania pytań
* Score – klasa dotycząca punktacji
* QuizFileWriter – klasa dotycząca zapisu wyników

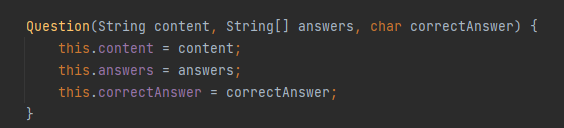
Oprócz tego program posiada również trzy klasy testowe:

* QuestionTests – odpowiada za testy pytań
* QuizFileReader – odpowiada za test ilości ładowanych pytań
* ScoreTests – odpowiada za test punktacji

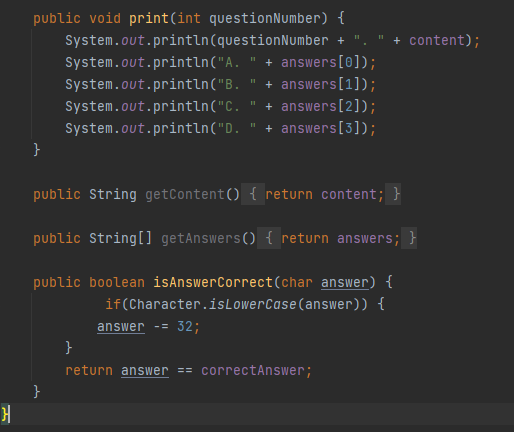
Projekt został wykonany w środowisku programistycznym IntelliJ Idea 2021.3.2 (Community Edition) na komputerze z systemem Windows 10. Teraz pokrótce chciałbym przedstawić dalsze zagadnienia techniczne dotyczące klas w tym projekcie.

Klasa Main odpowiada za utworzenie nowego Quizu i wywołania metody rozpoczęcia gry

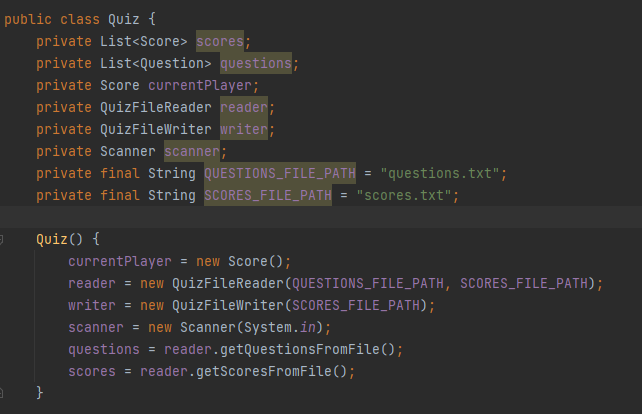
Rys. Kod klasy Main

Teraz przejdźmy to klasy Question. Konstruktor Question odpowiada za zainicjalizowanie pól klasy danymi które są podane w argumentach metody.

Rys. Konstruktor Question

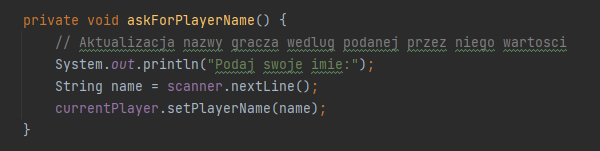
Następnie w kodzie widnieje metoda „print” która odpowiada za wyświetlanie pytań, oraz getery, dzięki którym mamy dostęp do pobrania wartości prywatnego pola klasy z zewnątrz. Dzięki metodzie „isAnswerCorrect” mamy możliwość sprawdzenia odpowiedzi i w razie potrzeby przekształcenia małej litery (jeśli właśnie taką poda użytkownik) na wielką.

Rys. Dalszy ciąg klasy Question

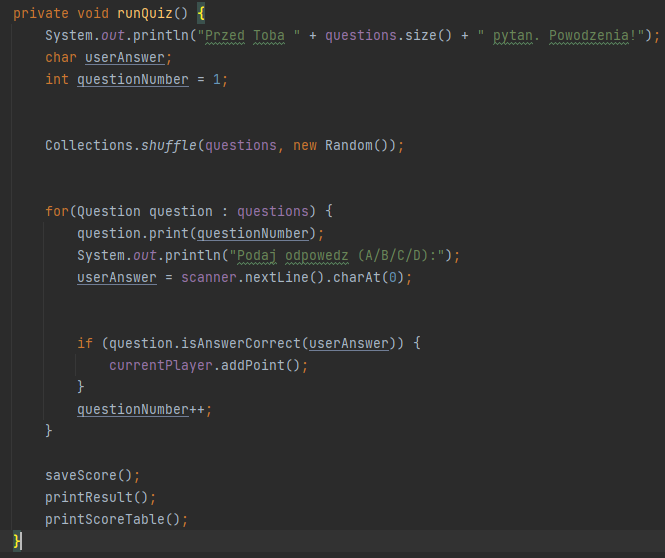
Klasa „Quiz” jest można rzec klasą główną, odpowiedzialną za rozgrywkę. Posiada pola klasy w której dwa ostatnie są zainicjalizowane i wskazują na dwa pliki tekstowe „questions.txt” oraz „scores.txt”. W plikach tych znajdują się odpowiednio pytania i wyniki. Konstruktor „Quiz” m.in. tworzy nową instancję klasy Score i przypisuje ją do pola currentPlayer.

Rys. Konstruktor i pola klasy Quiz

Metoda „askForPlayerName” przypisuje podane imie/nick (które użytkownik podaje podczas rozgrywki) do pola name za pomocą metody SetPlayerName



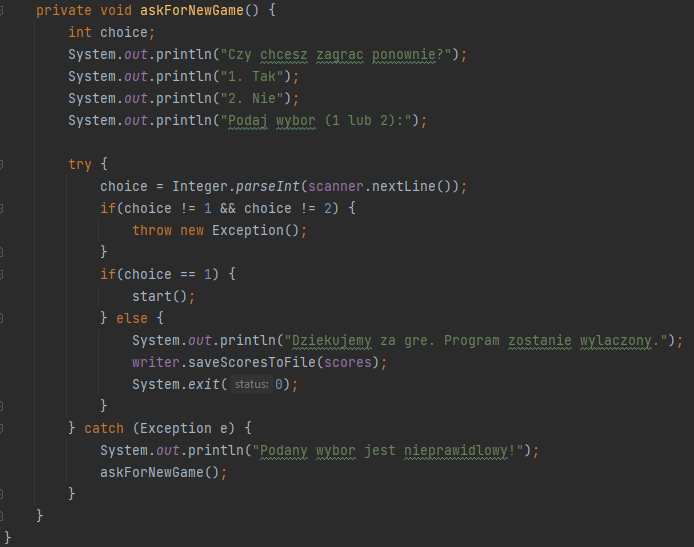
Rys. 7 Metoda askForPlayerName

Metoda „runQuiz” odpowiada w głównej mierze za przebieg rozgrywki. Na początku wyświetlany jest komunikat mówiący o ilości pytań (liczba pytań pobierana jest z pola questions). Tyle ile jest pytań to taka liczba się wyświetli. Następnie pobierany jest numer pytania (który po wykonaniu pętli jest inkrementowany), w każdym obrocie pętli są wyświetlane pytanie i odpowiedzi. Kolejną rzeczą jest pobranie odpowiedzi z klawiatury od użytkownika. Na końcu odpowiedź jest sprawdzana i zostaje przyznany punkt w razie poprawnej odpowiedzi. Za tasowanie (losową kolejność) pytań odpowiada „Collections.shuffle”. Po quizie wynik zostaje zapisany w liście wyników a następnie wyświetla się ranking.

Rys. Metoda runQuiz

Poniżej (Rys. 9) widzimy metody „SaveScore” oraz „printResult” – pierwsza zapisuje wynik aktualnego gracza, a druga wyświetla wynik w danej rozgrywce. Wyświetlaniem rankingu zajmuje się metoda „printScoreTable” w której lista wyników jest sortowana i odwracana, aby uzyskać kolejność od najwyższych do najniższych wyników. Takie sortowanie jest możliwe, ponieważ klasa „Score” implementuje interfejs Comparable.

Rys. Metody SaveScore oraz printResult

W metodzie „askForNewGame” mamy wyświetlane zapytanie o ponowną grę i pobranie odpowiedzi. Jeśli jest przecząca to wyniki zostają zapisane do pliku scores.txt, a program się kończy. W przypadku błędnej wartości (innej niż 1 lub 2) funkcja zostaje wywołana

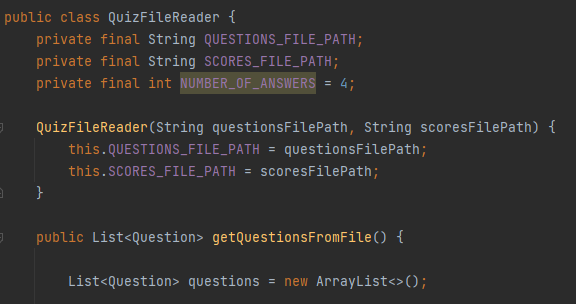
Rys. Metoda askForNewGame

Klasa Score dotyczy punktów i implementuje interfejs Comparable. Konstruktor Score domyślnie przypisuje nazwę gracza i punkty, a kolejne z jednym argumentem playerName oraz points przypisują graczowi jego imię/nick oraz punkty.

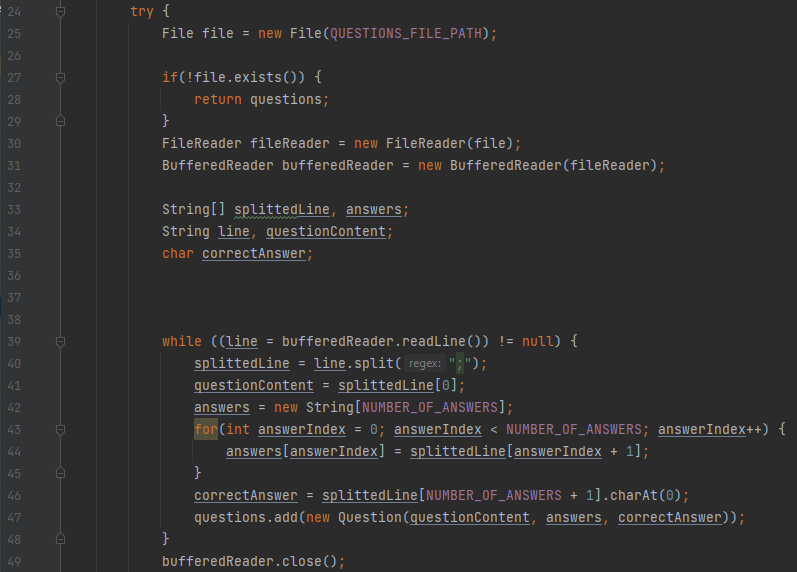
Rys. Konstruktory klasy Score

Dalej widzimy porównanie obiektów typu według ich punktów. Pozwala to m.in. na łatwe sortowanie listy z wynikami.

Rys. Dalsza część klasy Score

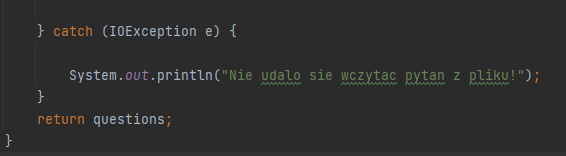
Klasa QuizFileReader dotyczy odczytywania z pliku. Tworzona jest pusta lista przechowująca pytania.

Rys. Klasa QuizFileReader

 Plik dzieli się na części oddzielone średnikiem, a z każdej linii pliku wczytywane są pytania. Do pierwszego średnika jest to część pytania (linijka kodu nr 41). Potem tworzona jest tablica odpowiedzi o stałej wartości 4, a każdy obrót pętli przypisuje do tablicy answers kolejną odpowiedź z pliku po średniku. Później następuje przypisanie poprawnej odpowiedzi z pliku, a do listy dodaje się nowy obiekt typu Question o parametrach odczytanych z pliku

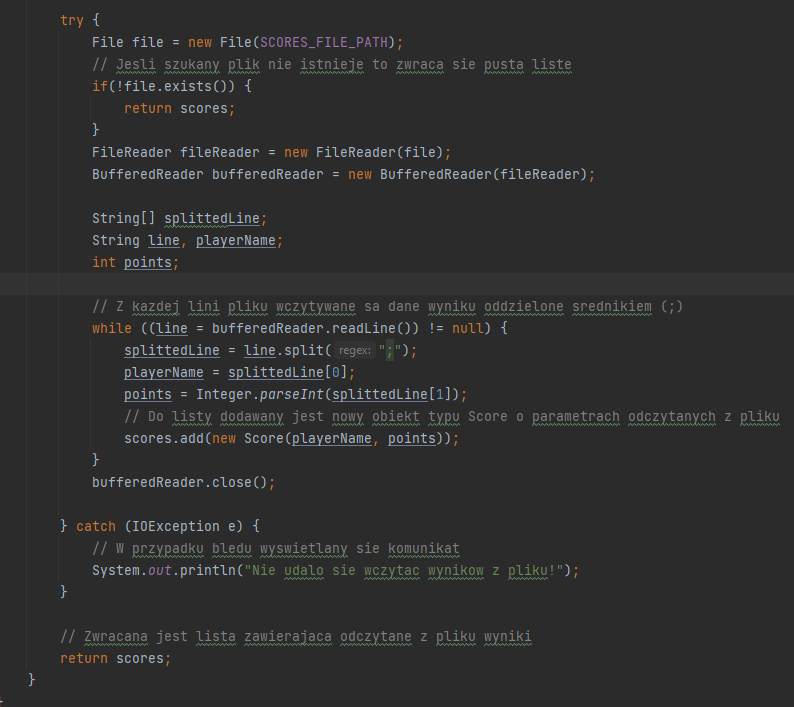
Rys. Dalsza część QuizFileReader

W przypadku błędu wyświetlany jest komunikat „Nie udało się wczytać pytań z pliku!”. Na końcu zwracana jest lista zawierająca odczytane z pliku pytania.



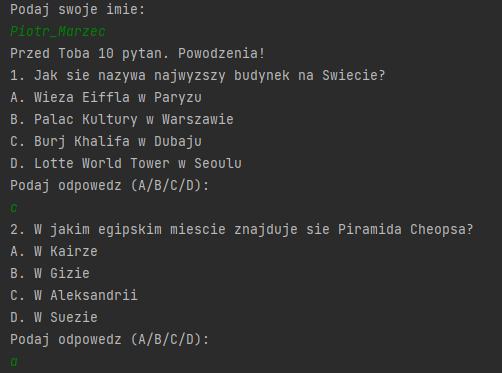
Rys. Kod błędu w klasie QuizFileReader oraz zwracanie listy

Pobranie wyników z pliku jest bardzo podobne (Rys. 16)

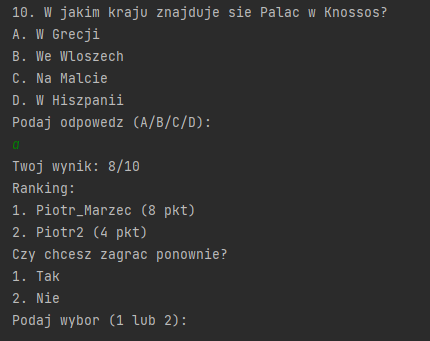


Rys. Pobranie wyników z pliku

# Prezentacja warstwy użytkowej

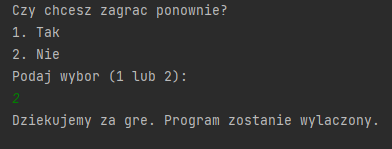
Gra typu QUIZ stworzona jest tak by była możliwie minimalistyczna, czytelna i prosta w obsłudze dla użytkownika. Po włączeniu gry wyświetla się nam komunikat o podaniu imienia/nicku gracza. Dalej możemy zobaczyć informacje o ilości pytań oraz już same pytania na które należy udzielić odpowiedzi.

Rys. Uruchomienie gry typu QUIZ oraz prezentacja pytań i odpowiedzi

Po udzieleniu odpowiedzi na wszystkie pytania wyświetla się uzyskany przez nas wynik, ranking oraz pytanie o ponowną rozgrywkę (Rys. 18)

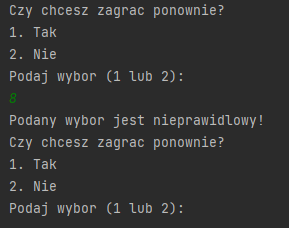
Rys. Dalsza cześć gry

Możemy wybrać jedną z dwóch opcji. Jeśli klikniemy „Tak” to gra rozpocznie się ponownie, natomiast jeśli „Nie” – program zakończy działanie.



Rys. Koniec programu

Przy wyborze innych cyfr niż 1 i 2, zostanie wyświetlony błąd a pytanie o ponowną rozgrywkę pojawi się ponownie (Rys. 20).

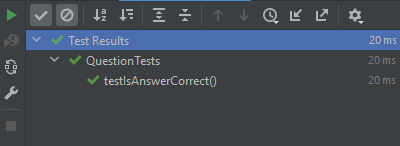


Rys. Błąd przy wyborze ponownej rozgrywki

# Raporty z testów jednostkowych

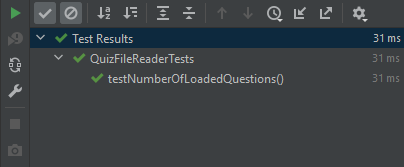
W celu sprawdzenia poprawności działania niektórych elementów gry zostały wykonane testy jednostkowe za pomocą JUnit w wersji 5.

Test klasy Question (Rys. 21)



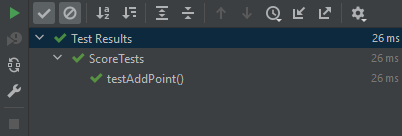
Rys. 21 Test klasy Question

Drugim testem jest QuizFileReaderTests (Rys.22)



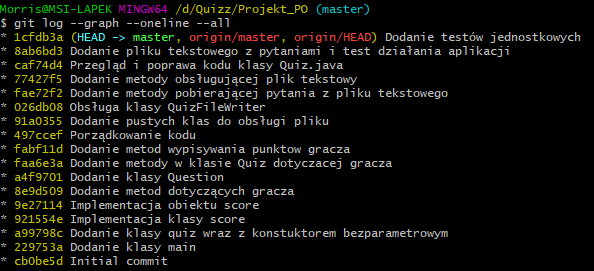
Rys. Test klasy QuizFileReader

Ostatnim testem, który został wykonany jest ScoreTests na klasie ScoreTest (Rys. 23)



Rys. Test klasy Score

# System kontroli wersji

Projekt prowadzony był z wykorzystaniem systemu kontroli Git, który zabezpiecza prace w kodzie oraz służy do zarządzania historią zmian. Poniżej (Rys. 24) prezentuję historie commitów

Rys. Historia commitów

# Dokumentacja projektu

Do projektu gry został stworzony system komentarzy z wykorzystaniem Javadoc. Cała dokumentacja wraz z plikami źródłowymi znajduje się na platformie GitHub pod adresem: <https://github.com/Marzec77/Projekt_PO.git>.

# 10. Literatura

1. https://www.samouczekprogramisty.pl/kurs-programowania-java/ (data dostępu: 3.02.2002)

#### 2. Cay S. Horstmann, „Java. Podstawy”, wyd. Helion

3. https://www.w3schools.com/java/default.asp (data dostępu: 10.02.2022)

4. https://www.youtube.com/watch?v=wk1Fbqh7Tew (data dostępu: 9.02.2022)

5. https://strefakursow.pl/ (data dostępu: 9.02.2022)

6. Wykłady online z przedmiotu „Programowanie Obiektowe”, Wyższa Szkoła Informatyki i Zarządzania w Rzeszowie, prowadzący: dr inż. Barbara Fryc (data dostępu: 10.02.2022)