



**WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA**
z siedzibą w Rzeszowie

Dokumentacja projektu

Przedmiot: Programowanie obiektowe

Tytuł projektu:

„Quiz wiedzy o Polsce ”

Prowadzący:

mgr inż. Ewa Żesławska

Wykonawca:

Kamil Markowski

w64152

Semestr, symbol kierunku i grupa: 3IIZ/GP02

Rzeszów 2022

1.	Opis założeń projektu	3
2.	Specyfikacja Wymagań	3
2.1.	Wymagania funkcjonalne	3
2.1.	Wymagania нефunkcjonalne	3
3.	Diagram przypadków użycia	3
4.	Opis techniczny projektu	4
5.	Prezentacja warstwy użytkowej projektu	12
6.	Raporty z testów jednostkowych	14
7.	System kontroli wersji	15
8.	Literatura	15

1. Opis założeń projektu

Gra „Quiz wiedzy o Polsce ” polega na sprawdzeniu podstawowej wiedzy gracza z zakresu historii, geografii oraz wiedzy ogólnej o Polsce. Quiz składa się z 10 pytań. Każde pojedyncze pytanie jest wyświetlane na ekranie. Osoba biorąca udział w quizie wpisuje odpowiedź a, b, c lub d. Gracz na koniec quizu widzi na ile pytań odpowiedział poprawnie, wyświetlone zostają informacje o ocenie jaką uzyskał oraz o sumie zdobytych punktów. Wyświetlone zostają progi punktacyjne na poszczególne oceny. Za każdą poprawną odpowiedź gracz otrzymuje 2 punkty maksymalnie otrzymać może 20 punktów.

2. Specyfikacja Wymagań

2.1. Wymagania funkcjonalne

- Gracz po zakończeniu gry ma dostęp do swojego wyniku.
- Pytanie jest wyświetlane, gdy udzielona jest odpowiedź na poprzednie pytanie.
- Do pytań nie można wracać.
- Poprawna jest tylko jedna z czterech odpowiedzi.

2.1. Wymagania нефunkcjonalne

- Możliwość zmiany treści pytania, czterech możliwych odpowiedzi oraz poprawnej odpowiedzi w pliku txt z poziomu pliku.
- Quiz działa z plikiem txt i używa danych w nim zawartych.
- Gra jest prosta w użyciu oraz przyjemna dla użytkowników.
- Gra działa na systemach MacOS.
- Quiz powstał w języku Java z frameworkiem JUnit.

3. Diagram przypadków użycia

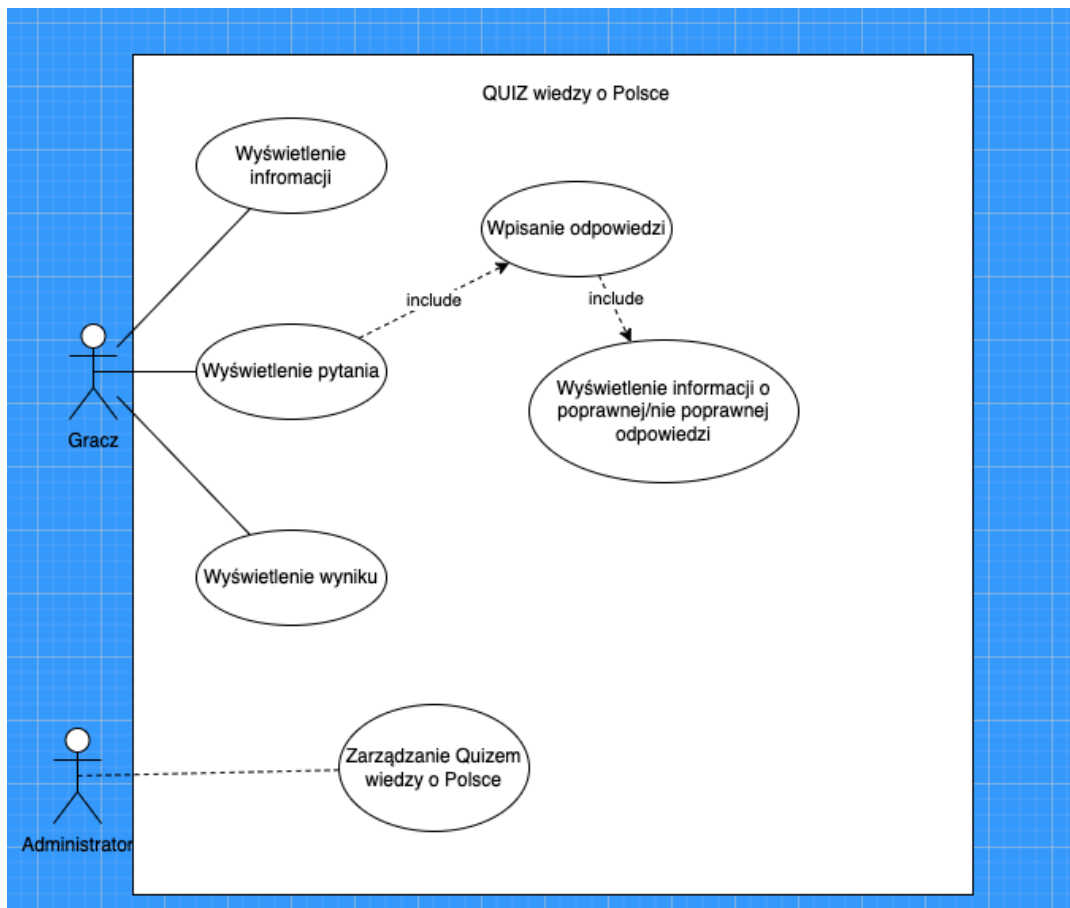


Diagram przypadków użycia

4. Opis techniczny projektu

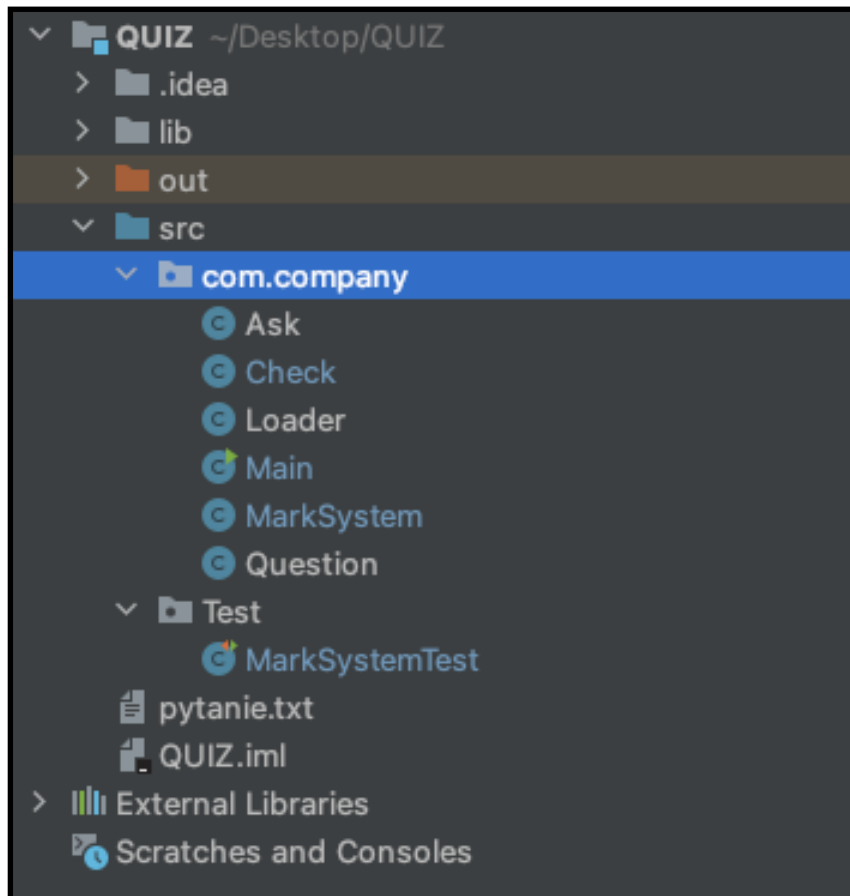
- Środowisko programistyczne Javy: IntelliJ Idea 2021.2.2 ultimate edition
- Biblioteki: java.util.Scanner, java.io.File
- Wersja Javy: 17

Gra quiz została napisana w języku Java obiektowo. Z podziałem na klasy oraz osobną klasę do testów jednostkowych. Metody wykorzystane są statyczne, gdyż nie jest tworzona instancja klasy dla każdej klasy z osobna tylko jedna dla wszystkich klas.

Projekt „Quiz wiedzy o Polsce” składa się z plików:

- **Main.java**
- **Question.java**
- **Ask.java**
- **Loader.java**
- **Check.java**

- **MarkSystem.java**
- **MarkSystemTest.java**
- **pytanie.txt**



Rysunek 1. Struktura quizu

Klasa **Question.java** - przechowuje zmienne potrzebne do działania gry (patrz Rysunek 2). Pola są statyczne, gdyż są one współdzielone przez wszystkie obiekty tej klasy w przeciwieństwie do pól niestatycznych, których własne egzemplarze ma każdy obiekt klasy.

```

1 package com.company;
2
3
4 public class Question {
5
6     public static String contents; //tresc pytania
7     public static String a, b, c, d; //mozliwe odpowiedzi
8     public static String true_answer; //poprawna odpowiedz
9     public static String user_answer; //odpowiedz od uzytkownika
10    public static int number_question; //numer pytania
11    public static int score; //punkty zdobyte na koniec gry zwiekszane o 1 gdy gracz odpowie poprawnie
12 }

```

Rysunek 2. Klasa Question.java

Klasa **Loader.java** (dziedziczy z klasy **Question.java**) odpowiada za wczytanie danych z pliku tekstowego **pytanie.txt**. Klasa ta wykorzystuje bibliotekę **java.util.Scanner** oraz **java.io.File** do pracy na plikach. Posiada statyczną metodę **void load()**, która korzysta ze zgłaszania wyjątków **FileNotFoundException** (linia 9). Pętla while wykorzystuje instancje klasy **Scanner** (linia 18) i wczytuje po kolei linie pliku **pytanie.txt**, przy czym za linie uznaje koniec linii ze znakiem „Enter”. W linii 15 zadeklarowana jest zmienna przechowująca wzór, dzięki której co 7 linia jest treścią pytania .(patrz Rysunek 3).

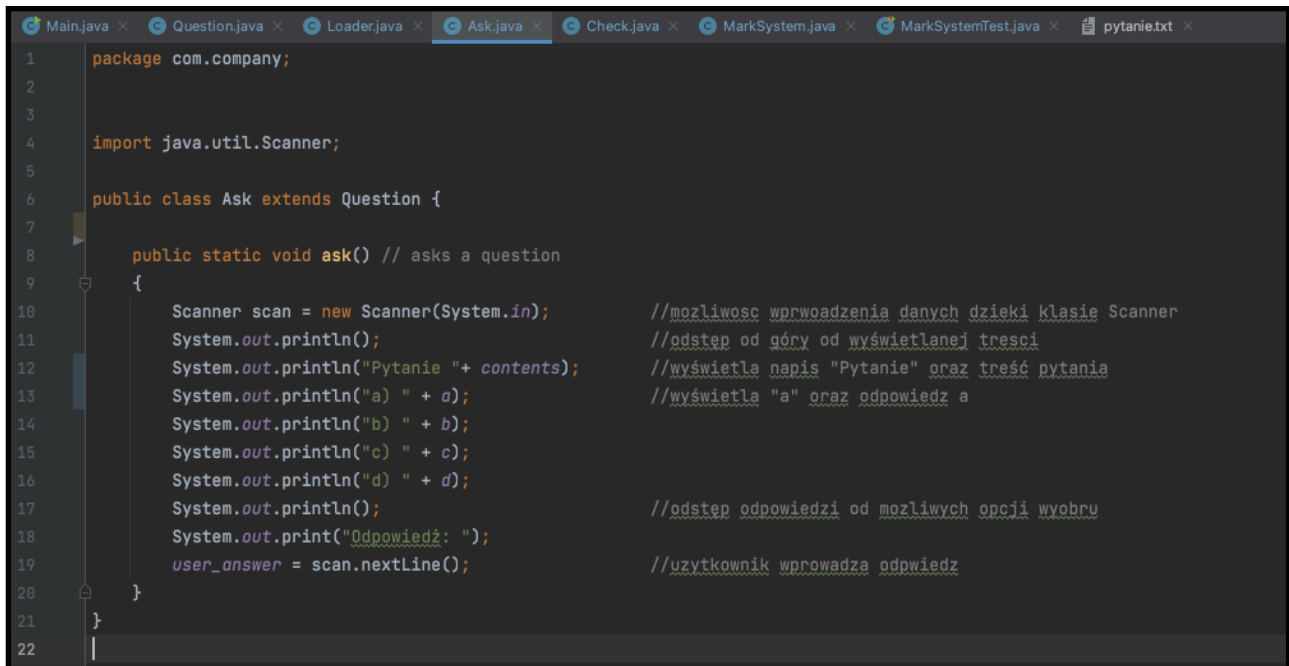
```

1 package com.company;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5 import java.util.Scanner;
6
7 public class Loader extends Question {
8
9     public static void load() throws FileNotFoundException // funkcja void, które wczytuje plik pytanie.txt oraz wyświetla dane w nim
10    {
11        File file = new File(pathname: "pytanie.txt"); // klasa file jest to zacząp do pliku
12        Scanner scanner = new Scanner(file); //tworze nowy obiekt scanner klasy Scanner
13
14        String line;
15        int line_number = (number_question - 1) * 6 + 1; //wzór według którego co 7 linia to pytanie
16        int current_line = 1; // aktualny nr linii
17
18        while(scanner.hasNext()) { //petla while wykorzystuje obiekt scanner z klasy Scanner i wczytuje dopóki ma następną linię do wczytania
19            line = scanner.nextLine(); //linia to następna linia w metodzie scanner ???
20            if (current_line == line_number) contents = line; //Jeżeli nr aktualnie czytanej linii zgadza się z numerem pierwszej linii dla pytania to tresc to nr linii
21            if (current_line == line_number + 1) a = line; //w tej linii wiekszej o 1 jest odpowiedz a
22            if (current_line == line_number + 2) b = line; //w tej linii wiekszej o 2 jest odpowiedz b
23            if (current_line == line_number + 3) c = line;
24            if (current_line == line_number + 4) d = line;
25            if (current_line == line_number + 5) true_answer = line;
26            current_line++; //aktualny nr linii
27        }
28        scanner.close(); //zamknięcie połączenia z plikiem
29    }
30 }
31

```

Rysunek 3. Klasa Loader.java

Klasa **Ask.java** (dziedziczy z klasy **Question.java**) posiada ona metodę statyczną **void ask()**, która służy do wyświetlenia w menu konsoli zawartości pliku **pytanie.txt** w kolejności: treść pytania (linia 12), 4 możliwe odpowiedzi (linia od 13 do 16) oraz do wpisania odpowiedzi gracza (linia 19). Korzysta z biblioteki **java.util.Scanner** do wprowadzenia odpowiedzi (patrz Rysunek 4).



```
1 package com.company;
2
3
4 import java.util.Scanner;
5
6 public class Ask extends Question {
7
8     public static void ask() // asks a question
9     {
10         Scanner scan = new Scanner(System.in); //możliwość wprowadzenia danych dzięki klasie Scanner
11         System.out.println(); //odstęp od góry od wyświetlanej treści
12         System.out.println("Pytanie " + contents); //wyświetla napis "Pytanie" oraz treść pytania
13         System.out.println("a) " + a); //wyświetla "a" oraz odpowiedź a
14         System.out.println("b) " + b);
15         System.out.println("c) " + c);
16         System.out.println("d) " + d);
17         System.out.println(); //odstęp odpowiedzi od możliwych opcji wyboru
18         System.out.print("Odpowiedź: ");
19         user_answer = scan.nextLine(); //użytkownik wprowadza odpowiedź
20     }
21 }
22
```

Rysunek 4. Klasa Ask.java

Klasa **Check** (dziedziczy z klasy **Question.java**) posiada metodę statyczną **check()** (linia 7) odpowiada w grze za sprawdzenie poprawności odpowiedzi gracza z poprawną odpowiedzią z pliku **pytanie.txt**. Zawiera instrukcję warunkową **if** wraz z metodą porównującą dwa **Stringi** (linia 9). Jeżeli wynik poprawny to do zmiennej **score** wpisze 2 i wyświetli komunikat (linia 11 i 12) w przeciwnym wypadku do zmiennej **score** wpisze 0 i wyświetli komunikat (linia 14 i 15). Klasa ta dodaje za każdą poprawną odpowiedź 2 punkty. (patrz Rysunek 5).

```
1 package com.company;
2
3 import java.util.Objects;
4
5 public class Check extends Question {
6
7     public static void check(){                                //metoda statyczna sprawdza odpowiedz gracza na pytanie
8
9         if(Objects.equals(user_answer, true_answer))          // porownywanie Stringow czy pierwszy jest rowny drugiemu
10        {
11            score = 2;                                          //jeżeli równe to do punktów przypisz 2
12            System.out.println("To poprawna odpowiedź !!!");    //wyświetl poprawna odpowiedz
13        } else {
14            score = 0;                                          //w przeciwnym wypadku do punktów przypisz 0
15            System.out.println("Zła odpowiedź ");              //wyświetl zła odpowiedz
16        }
17    }
18 }
19
```

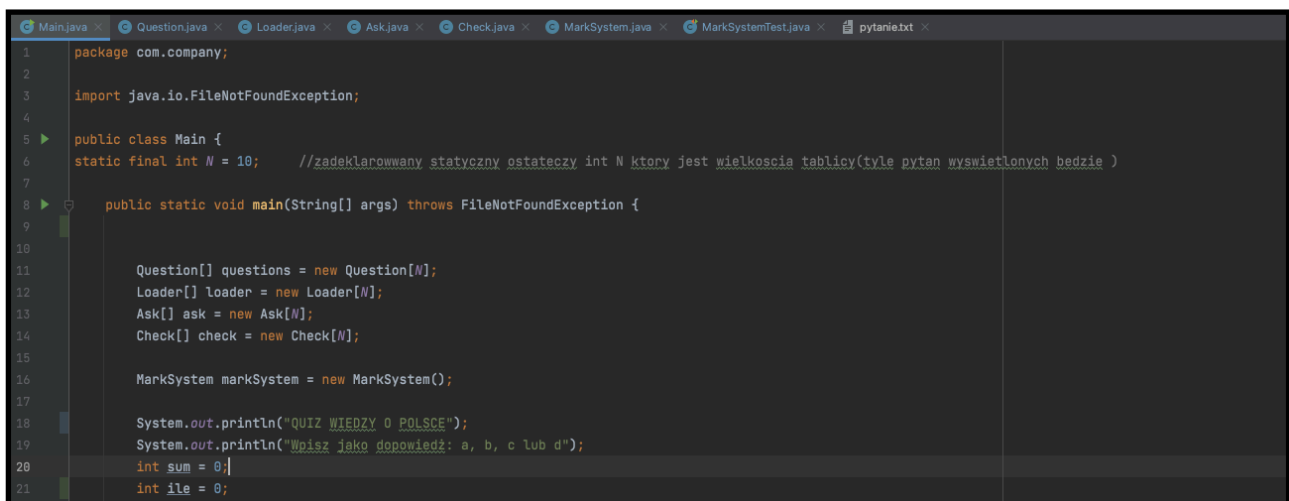
Rysunek 5 Klasa Check.java

Klasa **MarkSystem** (dziedziczy z klasy **Question.java**) posiada metodę **addScore()** przyjmuje ona parametr typu **int** o nazwie **sum** (linia 5). Służy do wystawienia oceny w zależności od ilości zdobytych punktów. (patrz Rysunek 6).

```
1 package com.company;
2
3 public class MarkSystem extends Question{
4
5     public int addScore(int sum)                               //metoda dodaj punkt przyjmuje parametr typu int o nazwie sum
6     {
7         if(sum <= 20 && sum >= 18) return 5;                  //jeżeli wynik gracza jest pomiędzy 20 a 18 to zwróć 5
8
9         if(sum <= 16 && sum >= 14) return 4;                  //jeżeli wynik gracza jest pomiędzy 16 a 14 to zwróć 4
10
11        if(sum <= 12 && sum >= 10) return 3;                    //jeżeli wynik gracza jest pomiędzy 12 a 10 to zwróć 3
12
13        if(sum <= 8 && sum >= 6) return 2;                     //jeżeli wynik gracza jest pomiędzy 8 a 6 to zwróć 2
14
15        if(sum <= 4) return 1;                                  //jeżeli wynik gracza jest mniejszy niż 4 to zwróć 1
16
17        return sum;                                             // zwraca sume
18    }
19 }
20
```

Rysunek 6. Klasa MarkSystem.java

Klasa **Main.java** jest to główny plik posiada publiczną statyczną metodę **void main()**. Korzysta z wyjątku **FileNotFoundException** (linia 8). Zadeklarowana jest zmienna statyczna, finalna typu **int** o nazwie **N = 10** służy ona do określenia wielkości tablicy (Ilość pytań, która zostanie wyświetlona). (linia 6). W liniach od 11 do 14 utworzone są obiekty tablicowe o rozmiarze podanym w „N”. W linii 16 utworzona jest instancja klasy **MarkSystem** odpowiedzialna za wystawianie oceny na koniec gry. Linia 18 oraz 19 są to wyświetlenia informacji o quizie (patrz Rysunek 7).



```
1 package com.company;
2
3 import java.io.FileNotFoundException;
4
5 public class Main {
6     static final int N = 10; //zadeklarowany statyczny ostateczny int N który jest wielkoscia tablicy(tyle pytan wyswietlonych bedzie )
7
8     public static void main(String[] args) throws FileNotFoundException {
9
10
11         Question[] questions = new Question[N];
12         Loader[] loader = new Loader[N];
13         Ask[] ask = new Ask[N];
14         Check[] check = new Check[N];
15
16         MarkSystem markSystem = new MarkSystem();
17
18         System.out.println("QUIZ WIEDZY O POLSCE");
19         System.out.println("Wpisz jako odpowiedz: a, b, c lub d");
20         int sum = 0;
21         int ile = 0;
```

Rysunek 7. Klasa Main.java

Klasa **Main.java** posiada pętlę **for** służącą do wypełnienia tablicy obiektami, gdyż tablica automatycznie wypełniona jest wartościami **null**. W linii 29 jest instrukcja warunkowa **if**, która sprawdza, czy wartości obiektów są różne od **null**. Jeżeli są to uruchamiane zostają metody z poszczególnych klas bez tworzenia instancji klas, gdyż metody w nich są statyczne. Zmienna **sum** przechowuje ilość zdobytych punktów (linia 35). Zmienna **ile** przechowuje liczbę pytań, na którą gracz odpowiedział poprawnie (linia 36). W Linii 39 wyświetlone zostają informacje o ilości poprawnych pytań. Linia 40 wyświetla informacje o uzyskanej ocenie na koniec quizu. Linia 41 wyświetla sumę zdobytych punktów. Linie od 42 do 46 wyświetlają informacje o progach na poszczególne oceny (patrz Rysunek 8).

```

23     for (int i = 0; i < questions.length; i++)
24     {
25         questions[i] = new Question(); //przypisany obiekt do tablicy
26         loader[i] = new Loader();
27         ask[i] = new Ask();
28         check[i] = new Check();
29         if (questions[i] != null && loader[i] != null && ask[i] != null && check[i] != null) //sprawdzenie czy tablica nie jest wypełniona wartościami null
30         {
31             Question.number_question = i + 1; //bez tworzenia obiektu static
32             Loader.load();
33             Ask.ask();
34             Check.check();
35             sum += Question.score; //dodaje 2pkt za kazda dobra odpowiedz do zmiennej suma
36             ile = sum/2; //liczy na ile pytań dobrze odpowiedział gracz
37         }
38     }
39     System.out.println("Odpowiedziałeś poprawnie na: " + ile + " / " + N + " pytań");
40     System.out.println("Twoja ocena: " + markSystem.addScore(sum));
41     System.out.println("Gratuluję zdobycia: " + sum + " pkt");
42     System.out.println("ocena 5 - 20/18 pkt");
43     System.out.println("ocena 4 - 16/14 pkt");
44     System.out.println("ocena 3 - 12/10 pkt");
45     System.out.println("ocena 2 - 8/6 pkt");
46     System.out.println("ocena 1 - < 4 pkt");
47 }
48 }

```

Rysunek 8. Klasa Main.java

Plik tekstowy **pytanie.txt** stworzony jest tak, że pierwsza linia to treść pytania następne 4 linie to możliwe odpowiedzi a linia 6 to poprawna odpowiedź na zadane pytanie. (Jest wczytywany w klasie Loader.java); (patrz Rysunek 9).

Struktura pliku:

1. Treść pytania.
2. Możliwa odpowiedź „a”.
3. Możliwa odpowiedź „b”.
4. Możliwa odpowiedź „c”.
5. Możliwa odpowiedź „d”.
6. Poprawna odpowiedź

Plik skonstruowany jest tak, że co siódma linia jest pytaniem.

```
1 1. Z iloma państwami graniczy Polska?
2 6
3 7
4 8
5 9
6 b
7 2. Jakie jest najgłębsze jezioro w Polsce?
8 Śniardwy
9 Hańcza
10 Mamry
11 Solina
12 b
13 3. Z jakim państwem Polska ma najdłuższą granicę?
14 Ukraina
15 Niemcy
16 Rosja
17 Czechy
18 d
19 4. Kto napisał słowa do hymnu Polski?
20 Józef Dąbrowski
21 Józef Wybicki
22 Stanisław Kosciuszko
23 Nie wiadomo
24 b
25 5. Ostateczny kształt granic Polski, obowiązujący do dziś, ustalono w roku:
26 1945
27 1989
28 1991
29 1951
30 d
```

Rysunek 9. Plik tekstowy pytanie.txt

Klasa **MarkSystemTest.java** służy do wykonywania testów jednostkowych. (patrz Rysunek 10).

```
1 package Test;
2 import com.company.MarkSystem;
3 import org.junit.Assert;
4 import org.junit.jupiter.api.*;
5
6 public class MarkSystemTest { //Klasa, którą testuje
7     private MarkSystem markSystem; //instancja klasy MarkSystem
8
9     @BeforeEach // @BeforeEach Wykona się zawsze przed testami
10    void beforeEach() { //metoda void
11        markSystem = new MarkSystem(); // utworzenie instancji klasy
12    }
13
14    @RepeatedTest(value = 4) //test powtarzalny sprawdzi 4 razy
15    public void MarkSystem_IsCorrect_Mark() { //sprawdzanie poprawności wysawiania ocen zależnie od ilości podanych pkt
16        //given
17        int a = 16; //ilość punktów zdobytych przez gracza
18
19        //when
20        int result = markSystem.addScore(a); //zapisanie do zmiennej 'result' oraz wywołanie logiki
21
22        //then
23        Assert.assertEquals("expected: 4, result); //sprawdzenie oczekiwany wynik to zwrócenie liczby 4 dla 16 pkt (dla a = 16) przyjmuje dwa parametry wartość oczekiwana
24    }
25
26
27
28
```

Rysunek 10. Klasa MarkSystemTest.java

5. Prezentacja warstwy użytkowej projektu

Po uruchomieniu quizu przedstawiony jest ekran początkowy. Wyświetlone zostaje menu konsolowe z podstawowymi informacjami o grze. Użytkownik widzi wyświetlane pytanie na temat ogólnej wiedzy o Polsce. Odpowiada na pytanie wpisując odpowiednią literę (a, b, c, d) oraz zatwierdzając klawiszem „Enter” (patrz Rysunek 1).

```
QUIZ WIEDZY O POLSCE
Wpisz jako odpowiedź: a, b, c lub d

Pytanie 1. Z iloma państwami graniczy Polska?
a) 6
b) 7
c) 8
d) 9

Odpowiedź:
```

Rysunek 1. Ekran początkowy quizu

W przypadku złej odpowiedzi gracz zostaje poinformowany o złym wyborze. Za złą odpowiedź przyznawane jest 0 pkt. (patrz Rysunek 2).

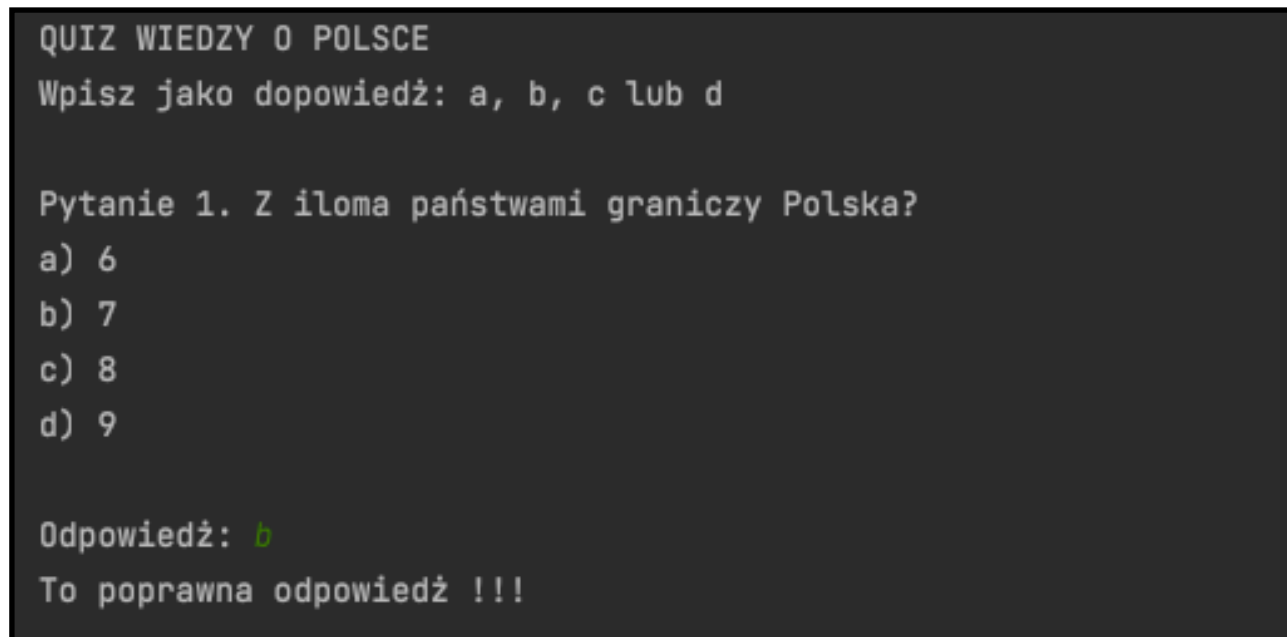
```
QUIZ WIEDZY O POLSCE
Wpisz jako odpowiedź: a, b, c lub d

Pytanie 1. Z iloma państwami graniczy Polska?
a) 6
b) 7
c) 8
d) 9

Odpowiedź: a
Zła odpowiedź
```

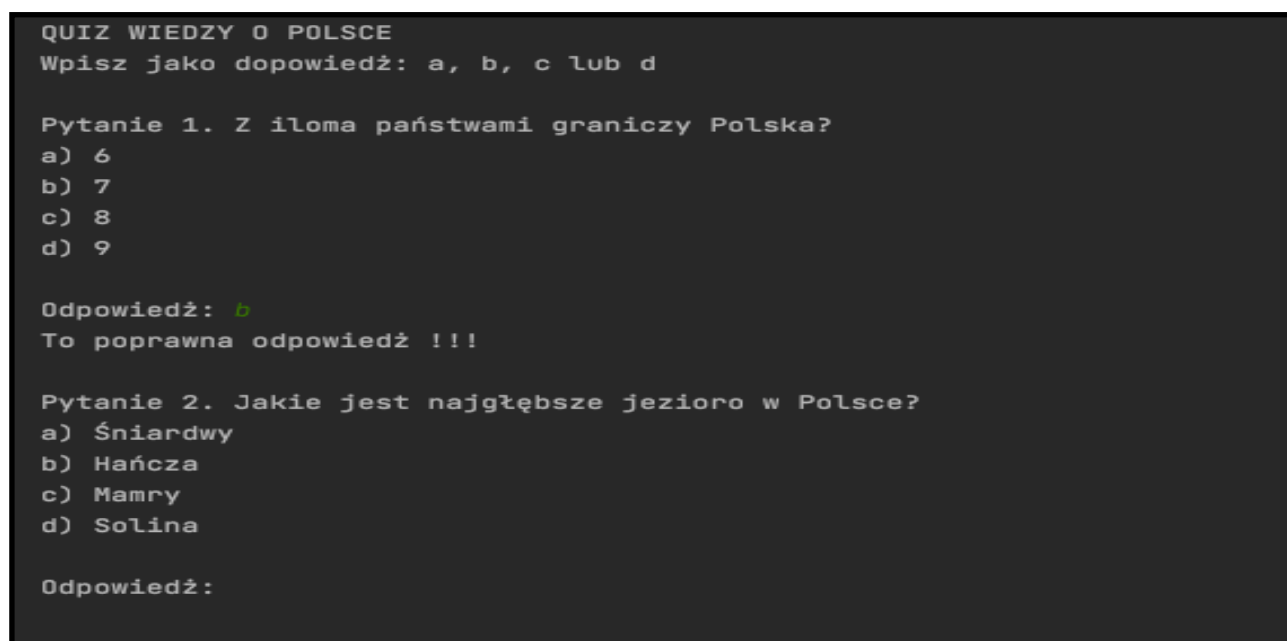
Rysunek 2. Ekran po złej odpowiedzi

W przypadku poprawnej odpowiedzi wyświetlony zostaje komunikat o poprawnej odpowiedzi oraz przyznane zostaje 2 pkt (patrz Rysunek 3).



Rysunek 3. Ekran po poprawnej odpowiedzi

Niezależnie od tego czy wynik odpowiedzi jest poprawny czy też błędny, zostaje wyświetlone kolejne pytanie (patrz Rysunek 4).



Rysunek 4. Wyświetlone kolejne pytanie

Po zakończonym quizie gracz dostaje informacje o (patrz Rysunek 5) :

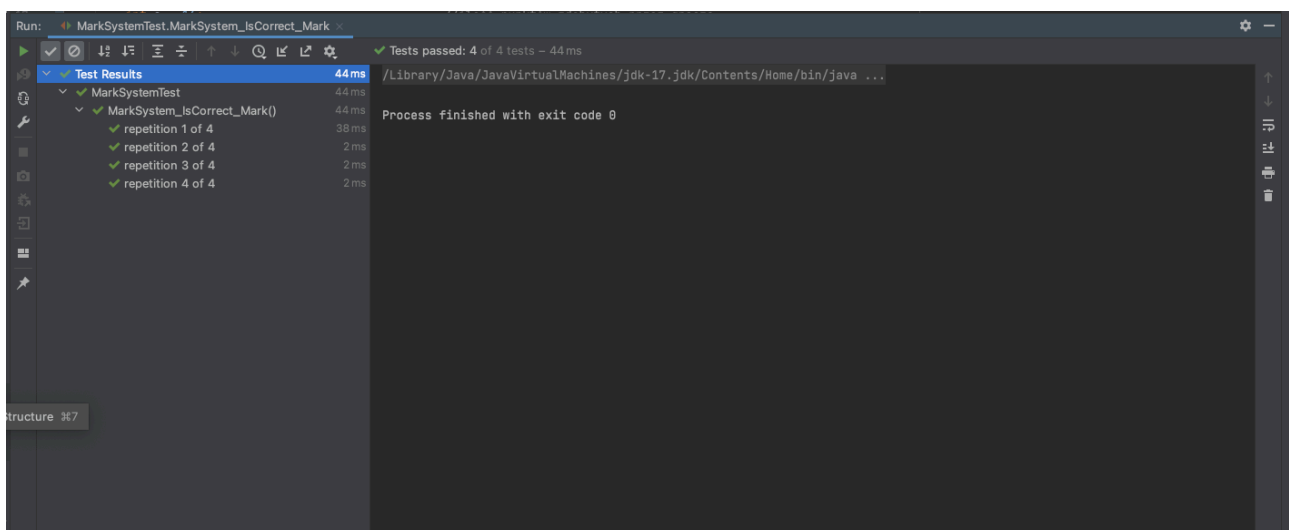
- Ilości poprawnych odpowiedzi.
- Ocenie.
- Zdobytych punktach.
- Skali punktów na poszczególne oceny.

```
Odpowiedziałeś poprawnie na: 7 / 10 pytań
Twoja ocena: 4
Gratuluję zdobyłeś: 14 pkt
ocena 5 - 20/18 pkt
ocena 4 - 16/14 pkt
ocena 3 - 12/10 pkt
ocena 2 - 8/6 pkt
ocena 1 - < 4 pkt
```

Rysunek 5. Końcowy informacje

6. Raporty z testów jednostkowych

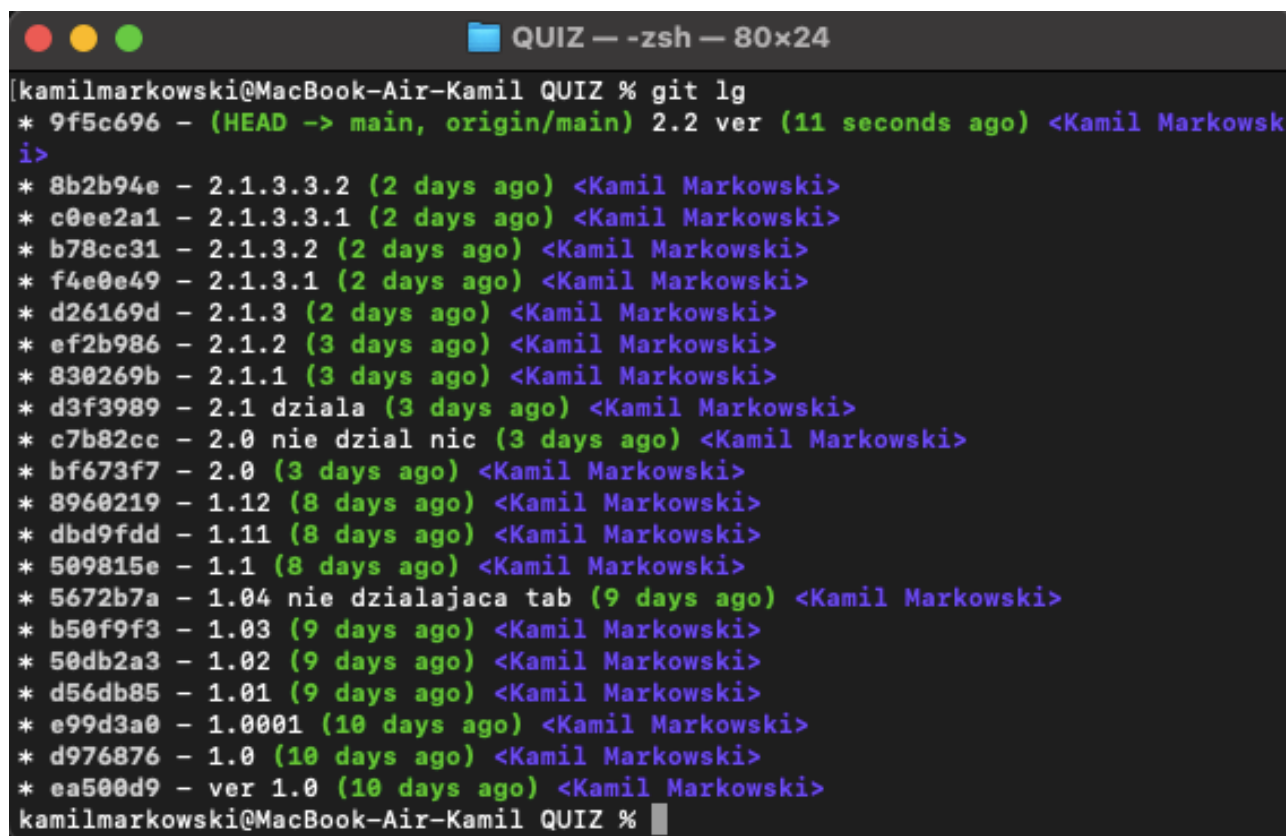
Przeprowadzony został test jednostkowy mający sprawdzić, czy dla podanej liczby punktów metoda **addScore(int sum)** w klasie **MarkSystem.java** wystawia poprawną ocenę. Sprawdza 4-krotnie otrzymany wynik. (Przykładowo dla 10 pkt poprawna wartość to 3 a dla 16 pkt to 4); (Patrz rysunek 1).



Rysunek 1. Test jednostkowy

7. System kontroli wersji

Projekt realizowany był z wykorzystaniem systemu kontroli wersji Git, a wszystkie pliki źródłowe projektu znajdują się pod adres: <https://github.com/gugol9/QUIZ> . Na rysunku 1 przedstawiono zrzut ekranu pokazujący historię commitów.



```
kamilmarkowski@MacBook-Air-Kamil QUIZ % git lg
* 9f5c696 - (HEAD -> main, origin/main) 2.2 ver (11 seconds ago) <Kamil Markowski>
* 8b2b94e - 2.1.3.3.2 (2 days ago) <Kamil Markowski>
* c0ee2a1 - 2.1.3.3.1 (2 days ago) <Kamil Markowski>
* b78cc31 - 2.1.3.2 (2 days ago) <Kamil Markowski>
* f4e0e49 - 2.1.3.1 (2 days ago) <Kamil Markowski>
* d26169d - 2.1.3 (2 days ago) <Kamil Markowski>
* ef2b986 - 2.1.2 (3 days ago) <Kamil Markowski>
* 830269b - 2.1.1 (3 days ago) <Kamil Markowski>
* d3f3989 - 2.1 dziala (3 days ago) <Kamil Markowski>
* c7b82cc - 2.0 nie dziala nic (3 days ago) <Kamil Markowski>
* bf673f7 - 2.0 (3 days ago) <Kamil Markowski>
* 8960219 - 1.12 (8 days ago) <Kamil Markowski>
* dbd9fdd - 1.11 (8 days ago) <Kamil Markowski>
* 509815e - 1.1 (8 days ago) <Kamil Markowski>
* 5672b7a - 1.04 nie dzialajaca tab (9 days ago) <Kamil Markowski>
* b50f9f3 - 1.03 (9 days ago) <Kamil Markowski>
* 50db2a3 - 1.02 (9 days ago) <Kamil Markowski>
* d56db85 - 1.01 (9 days ago) <Kamil Markowski>
* e99d3a0 - 1.0001 (10 days ago) <Kamil Markowski>
* d976876 - 1.0 (10 days ago) <Kamil Markowski>
* ea500d9 - ver 1.0 (10 days ago) <Kamil Markowski>
kamilmarkowski@MacBook-Air-Kamil QUIZ %
```

Rysunek 1. Historia commitów

8. Literatura

9. <https://javastart.pl/baza-wiedzy/wyjatki/nullpointerexception> (data dostępu: 26.12.2021).
10. <https://javastart.pl/baza-wiedzy/java-zadania/zadanie-wczytywanie-danych-scanner#rozwiązanie> (data dostępu: 26.12.2021).
11. <https://kursjava.com/klasy/metody-i-pola-statyczne/> (data dostępu: 03.01.2022).
12. <https://stormit.pl/testy-jednostkowe-junit/> (data dostępu: 05.01.2022).