

Uniwersytet Ekonomiczny w Krakowie

Projekt zaliczeniowy

Z przedmiotu

„Programowanie współbieżne i równoległe”

Wykonali:

Grupa KrDZIs3013Io

Dawid Wedman, 188618

Vladyslav Mostovych, 187516

Kraków 2017

Spis treści

1. Opis zagadnienia	3
2. Proponowane rozwiązanie	4
3. Wykorzystane biblioteki i klasy.....	5
4. Sposób obsługi	8

1. Opis zagadnienia.

Zaprojektowany program pozwala użytkownikom gry komputerowej „World of Tanks”¹ dowiedzieć się o stanie serwerów, ilości graczy połączonych z serwerami oraz sprawdzić stan połączenia z serwerem gry (ping do serwera) bez potrzeby uruchamiania klienta gry.

Program jest stworzony w języku Java.

¹ World of Tanks – gra komputerowa z gatunku MMO wyprodukowana i wydana w 2011 roku przez Wargaming.net. Gra oparta jest na modelu płatności free-to-play. ([Wikipedia link](#))

2. Proponowane rozwiązanie

Rozwiązaniem problemu otrzymania danych o stanie serwerów jest Wargaming.net public API dla deweloperów aplikacji. Za pomocą tego serwisu jest możliwe uzyskanie danych o stanie serwerów gier przedsiębiorstwa „Wargaming.net”.

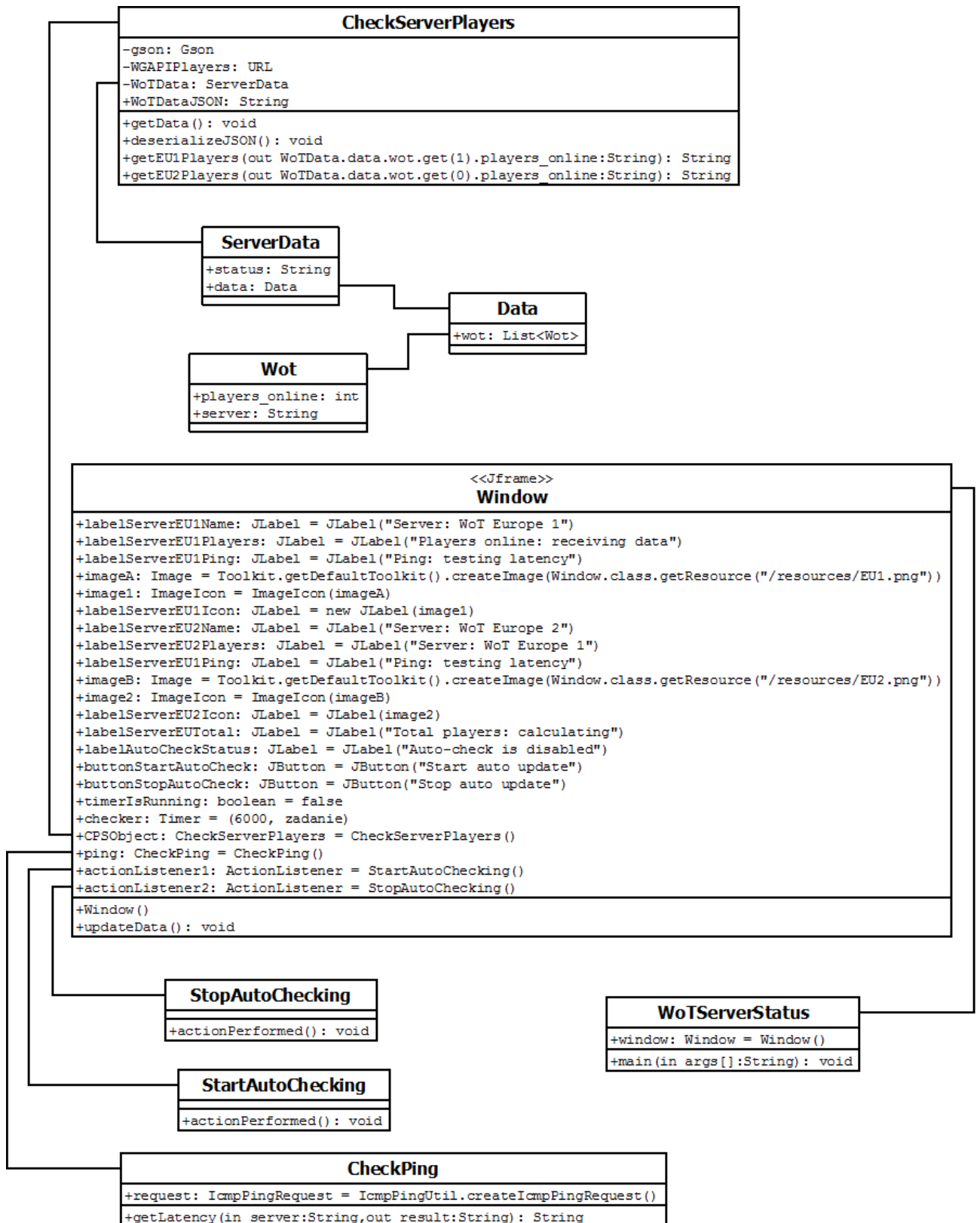
Wszystkie dane można otrzymać za pomocą serwisu w postaci pliku w formacie JSON.

Dla sprawdzania stanu połączenia z serwerami gry użyjemy postronną bibliotekę z możliwościami diagnostyki połączeń sieciowych.

Dla zaprojektowania interfejsu graficznego użytkownika użyjemy bibliotek standardowych „java.awt*” oraz „java.awt.event*”.

Dla stworzenia możliwości sprawdzania automatycznego stanu serwerów użyjemy biblioteki standardowej „javax.swing.*”.

3. Wykorzystane biblioteki i klasy



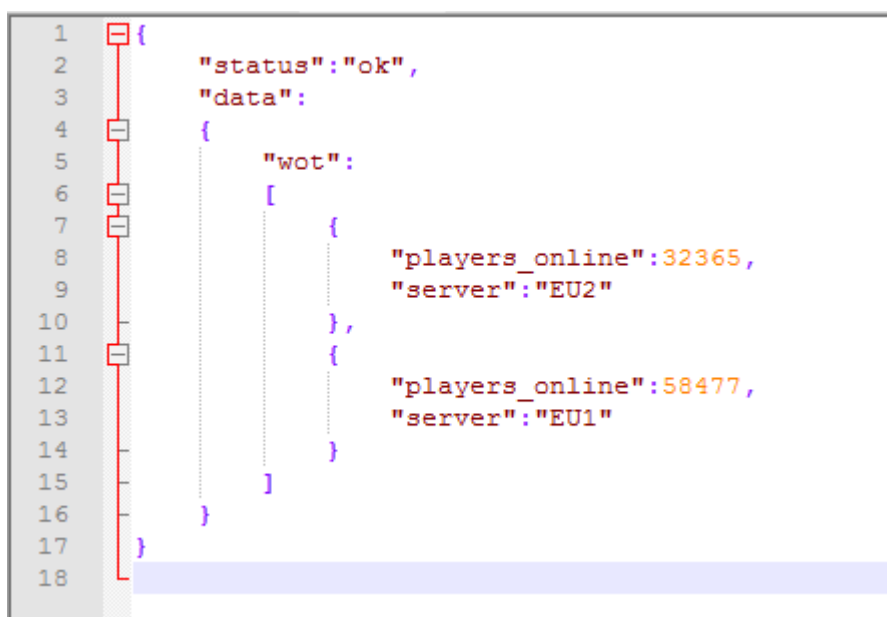
Dla otrzymania informacji o stanie serwerów zaprojektowaliśmy klasę *CheckServerPlayers*.

Metoda `getData()` klasy `CheckServerPlayers` pozwala wysłać zapytanie do serwera i otrzymać odpowiedź z danymi w postaci pliku JSON.

Przykład zapytania:

https://api.worldoftanks.eu/wgn/servers/info/?application_id=demo&game=wot

Przykład odpowiedzi:



```
1 {
2   "status": "ok",
3   "data":
4     {
5       "wot":
6         [
7           {
8             "players_online": 32365,
9             "server": "EU2"
10          },
11          {
12            "players_online": 58477,
13            "server": "EU1"
14          }
15        ]
16      }
17    }
18 }
```

Żeby odczytać te dane zaprojektowano klasy `ServerData`, `Data` oraz `Wot`.

Przetwarzanie pliku JSON w postaci obiektu wykonujemy za pomocą stworzonej metody `deserializeJSON()` klasy `CheckServerPlayers`. Plik w postaci JSON deserializujemy w obiekt na podstawie danych klas za pomocą biblioteki „Google GSON²” wersji 2.8.0.

Przykład polecenia:

```
WoTData = gson.fromJson(WoTDataJSON, ServerData.class);
```

gdzie: `WoTData` – obiekt klasy `ServerData`;

`gson.fromJson` – metoda biblioteki Google Gson;

² Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of. ([GitHub.com](https://github.com/google/gson))

WoTDataJSON – plik w postaci JSON;

ServerData.class – podstawowa klasa(szablon) dla deserializacji.

Dla odczytania danych o ilości graczy na serwerach zaprojektowano metody *getEU1Players()* oraz *getEU2Players()* klasy *CheckServerPlayers*. Te metody zwracają dane o typie danych String z liczbą graczy na serwerach.

Przykład polecenia:

```
return "" + WoTData.data.wot.get(0).players_online;
```

Do otrzymania informacji o stanie połączenia z serwerami zaprojektowano klasę *CheckPing*. Metoda tej klasy *getLatency(String server)* otrzymuje adres serwera i zwraca czas otrzymania odpowiedzi od serwera za pomocą biblioteki ICMP4J. Funkcje *executePingRequest()* oraz *formatResponse()* zwracają dane w takim samym formacie jak polecenie ping w CMD(windows) oraz bash(OS X, Linux). Żeby otrzymać tylko liczbę usuwamy inny tekst za pomocą wyrażeń regularnych.

Przykład polecenia:

```
IcmpPingResponse response = IcmpPingUtil.executePingRequest(request); // Sending req
String formattedResponse = IcmpPingUtil.formatResponse(response); // And writing

Matcher matcher = Pattern.compile("time=(.*?) ([a-z]){2}").matcher(formattedResponse);
```

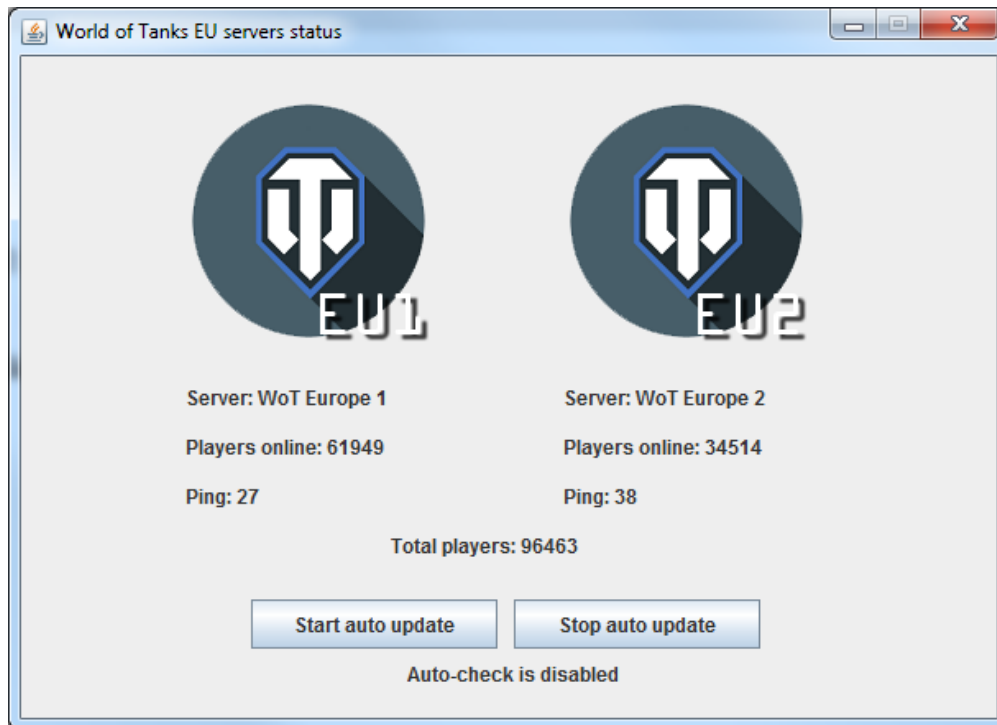
Dla interfejsu graficznego użytkownika zaprojektowano klasę *Window*. Za pomocą tej klasy wyświetlamy na ekranie przetworzoną informację co uzyskujemy podczas pracy programu w obiektach *JLabel*.

Sprawdzanie automatyczne stanu serwerów zaprojektowano za pomocą klasy *Timer* biblioteki standardowej „javax.swing.*”. Sprawdzanie automatyczne można włączyć albo wyłączyć za pomocą przycisków *JButton*.

4. Sposób obsługi

Dla uruchomienia aplikacji na komputerze musi być zainstalowane środowisko JRE ze strony <https://www.java.com/>.

Po uruchomieniu aplikacji WoTServerStatus.jar odtwarza się okno główne aplikacji.



Po uruchomieniu program automatycznie sprawdza stan serwerów oraz stan połączenia z serwerami.

Dla włączenia i wyłączenia automatycznego sprawdzania stanu serwerów należy użyć przycisków „Start auto update” oraz „Stop auto update”.

Jeżeli automatyczne sprawdzanie jest już włączone albo wyłączone użytkownik otrzyma odpowiedni komunikat.

