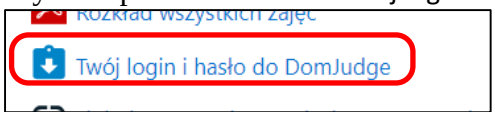
Wstęp.

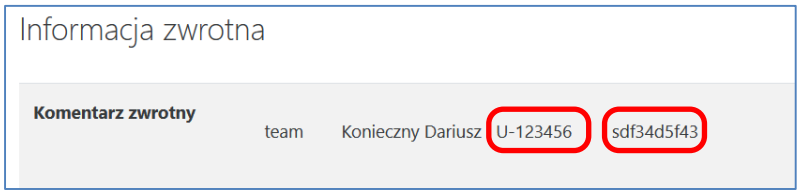
Podczas laboratoriów będziemy korzystać z automatycznego systemu sprawdzania DomJudge. Aby przetestować rozwiązanie z jego użyciem należy wykonać następujące kroki.

1. Pseudo-zadanie " Twój login i hasło do DomJudge".

Wybierz pseudo-zadanie "Twój login i hasło do DomJudge":



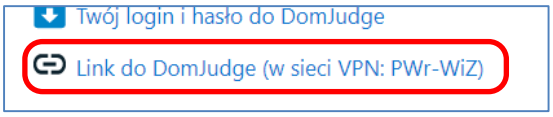
Następnie przejdź w dół, aby zobaczyć „Informacja zwrotna", która może wyglądać następująco:



Ciąg z prefiksem „U-" to nazwa loginu, więc w przykładzie jest to „U-123456". Drugie to hasło do tego konta – „sdf34d5f43".

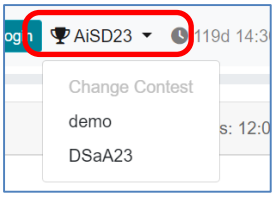
2. Używanie systemu DomJudge

Teraz możesz zalogować się do systemu sprawdzania rozwiązania DomJudge. Serwer jest dostępny tylko w sieci PWr-WiZ, więc poza zajęciami należy zalogować się do VPN-Wiz. Na stronie głównej kursu i wybierz „Link do DomJudge (w sieci VPN: PWr-WiZ)":

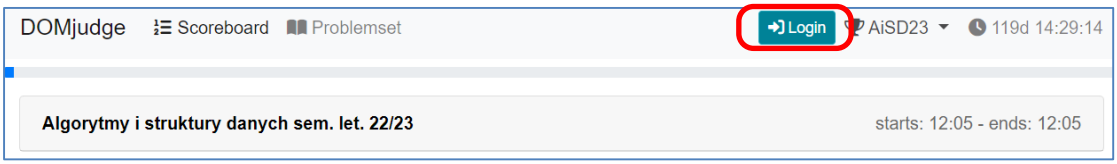


Spowoduje to otwarcie nowej strony internetowej z adresem URL http://10.108.42.19/domjudge/public.

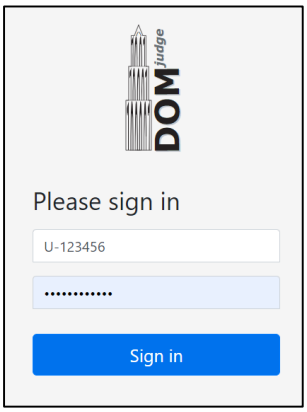
Pamiętaj, aby wybrać właściwy konkurs „AiSD23":



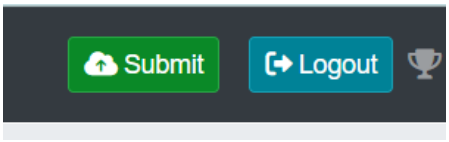
A następnie naciśnij przycisk „Login”:



Do logowaniu użyj loginu i hasła z punktu 1 tej instrukcji.

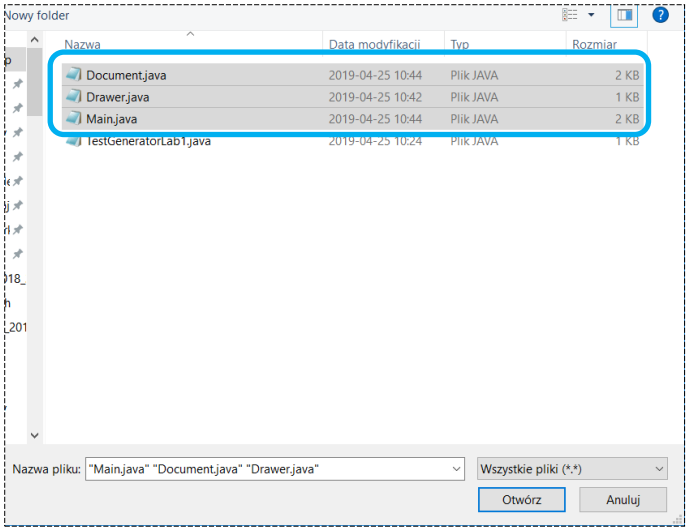


Teraz możesz wysłać rozwiązanie listy zadań. Na przykład, jeśli chcesz wysłać rozwiązanie zadania 1.1, musisz nacisnąć przycisk „Submit”:

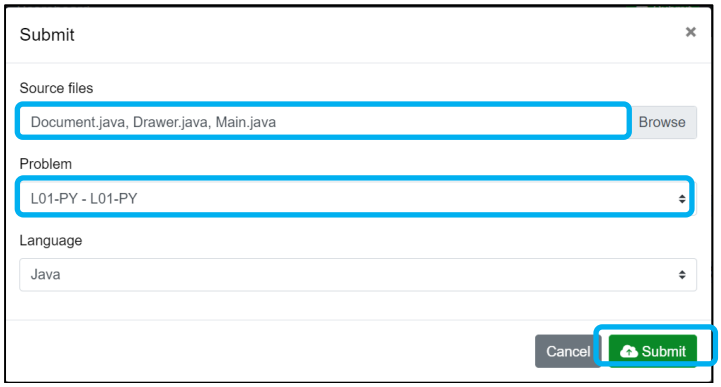


Następnie wciśnij przycisk „Browse”:

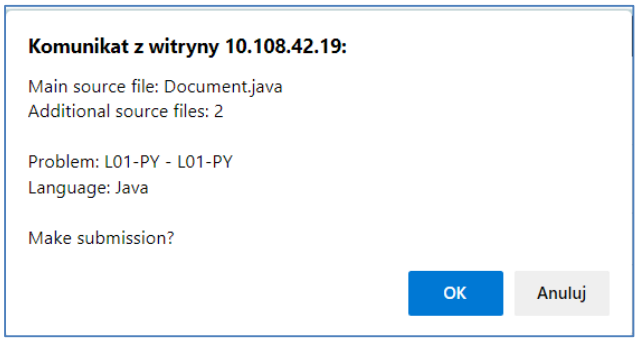
Wybierz wszystkie potrzebne do kompilacji pliki źródłowe:



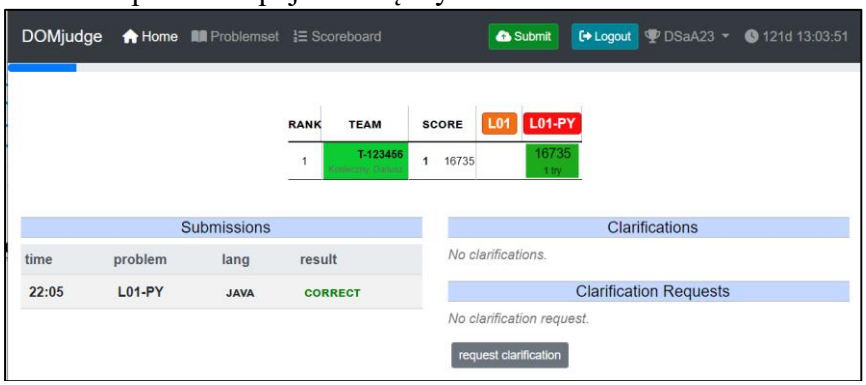
Wybierz oznaczenie problem i język Java oraz wciśnij "Submit":



Potwierdź wysłanie rozwiązania poprzez przycisk "OK" i poczekaj na wynik kompilacji i testowania:



Po chwili powinien pojawić się wynik:



Wynik testowania może być też negatywny, np.:

- Compile-error
- Wrong-answer
- Run-error
- Time-limit

Przejście testów automatycznych jest pierwszym krokiem oceniania rozwiązania. Dopiero potem w trakcie zajęć, zachodzą kolejne punkty poniższego procesu:

1) Rozwiązanie listy zadań (LZ) z wykorzystaniem testowania automatycznego - wykonuje student, również z domu.
2) Na początku zajęć nowe, krótkie zadanie analogiczne jak na zajęciach lub jego rozwinięcie/modyfikacja (NZ).
3) Zgłoszenie rozwiązania NZ – kolejka oczekujących.
4) Prezentacja rozwiązania NZ (kod i działanie).
5) Prezentacja kodu listy zadań LZ, który wcześniej przeszedł testy automatyczne.
6) Ocena punktowa za listę LUB poprawienie kodu i powrót do kroku 1 lub 2.

Lista zadań

1. Zaimplementuj metody:

a) `drawPyramid(int n)` która przyjmuje jako wejściową jedną liczbę całkowitą `n`, a następnie wyprowadza na konsoli piramidę jak na poniższym rysunku (na przykład dla *n*=4):

```
...X...
..XXX..
.XXXXX.
XXXXXXX
```

b) `drawChristmasTree(int n)` która przyjmuje jako wejściową jedną liczbę całkowitą `n`, a następnie wyprowadza na konsoli choinkę, w której wysokość ostatniej części jest równa n. Drzewo składa się z piramid o wysokości od 1 do `n`. Kształt musi być taki, jak pokazano poniżej (dla *n*=4):

```
...X...
...X...
..XXX..
...X...
..XXX..
.XXXXX.
...X...
..XXX..
.XXXXX.
XXXXXXX
```

2. Napisz procedurę `loadDocument(String name)`, która będzie wczytywać kolejne wiersze i analizować wiersz po wierszu, szukając linku w każdym wierszu. Format linku jest następujący: 5 znaków „`link=`” (dowolna kombinacja wielkich i małych liter), po których jest poprawny identyfikator. Poprawny identyfikator zaczyna się od litery (małej lub dużej), a następnie posiada nieprzerwany ciąg zero lub więcej liter lub cyfr lub znaków podkreślenia „_”. Procedura musi wypisać kolejno wszystkie poprawne identyfikatory, każdy w oddzielnej linii. Przed wypisaniem identyfikatory należy zmienić na **małe litery**. Dokument kończy się linią z tekstem „`eod`”, co oznacza koniec dokumentu.

Dla uzyskania maksymalnie 100 punktów zaprezentuj rozwiązanie do spotkania 2.
Dla uzyskania maksymalnie 50 punktów zaprezentuj rozwiązanie do spotkania 3.
Po spotkaniu 3 listę uznaje się za nierozwiązaną.

Rozwiązanie zostanie zautomatyzowane przetestowane testami z konsoli o przedstawionym poniżej formacie. Należy wykorzystać plik `Main.java` bez jego zmiany oraz wypełnić odpowiednie metody w plikach `Document.java` i `Drawer.java`.

Program zaczyna działanie od wypisania jednej linii ze napisem “START”.

Jeśli linia wejściowa zaczyna się od znaku „#” lub linia jest pusta, należy ją zignorować.
W przeciwnym razie linia wejściowa musi zostać skopiowana do linii wyjściowej z wykrzyknikiem przed pierwszym znakiem. Następnie należy wykonać właściwą operację.

Jeśli linia ma format:
PY *n*
twój program musi wywołać `drawPyramid(n)`. Możesz założyć (bez sprawdzania), że 1<=n<=20.

Jeśli linia ma format:
CT *n*
twój program musi wywołać `drawChristmasTree(n)`. Możesz założyć (bez sprawdzania), że 1<=n<=20.

Jeśli linia ma format:
LD *docName*
twój program musi wywołać `loadDocument(String docName)`.

Jeśli linia ma format:
`HA`
twój program musi zakończyć działanie, pisząc jako ostatnią linię "`END OF EXECUTION`". Każdy test kończy się tą komendą.

Na przykład dla pliku testowego:

```
py 3
ct 3
ld qwert
nnothing is here
link=abc link=qWe link=asd
link= broken li nk=wrong link =not
link=ok123_23sd what is here link=12wRong asdad link=_what12
dfasfdsdfsd
and now start LINK=$2323 LiNk=Ok
eod
ha
```

Wynikiem powinno być:

```
START
!py 3
..X..
.XXX.
XXXXX
!ct 3
..X..
..X..
.XXX.
..X..
.XXX.
XXXXX
!ld qwert
```

```
abc
qwe
asd
ok123_23sd
ok
!ha
END OF EXECUTION
```