PROJEKT: BATTLESHIPS

Autor: Jakub Orłowski

Data oddania: 30.10.2023r.

1. Opis projektu:

Projekt ten jest stworzony jako zadanie do zrealizowania na przedmiot Komunikacja Człowiek-Komputer. Treść zadania wygląda następująco:

„Zadanie polega na stworzeniu aplikacji działającej w trybie tekstowym. Tematyka aplikacji może być w pełni dowolna - od relatywnie prostych aplikacji tekstowych na niższe oceny po skomplikowane gry i pomysłowe projekty na wyższe.

Oceniane będą następujące aspekty:
- pomysł,
- atrakcyjność aplikacji dla użytkownika,
- ilość opcji, parametrów konfiguracyjnych, trybów działania itp.,
- interfejs użytkownika (najmniej punktowane wydawanie komend z linii poleceń, wyżej oceniane stworzenie wizualnie atrakcyjnego i wygodnego menu tekstowego, po którym użytkownik porusza się np. za pomocą strzałek),
- animacje (ASCII art),
- sposób odświeżania widoku (najmniej punktowane odświeżanie całego widoku z każdą najmniejszą zmianą, wyżej odświeżanie jedynie elementów, które uległy zmianie),
- ogólne wrażenie użytkowe.

Proszę zwrócić szczególną uwagę na kwestię interfejsu użytkownika i sposobu poruszania się po aplikacji, korzystania z niej.”

Projekt: BattleShips to gra konsolowa w Javie używająca biblioteki Lanterna w celu wyboru i edycji terminala. Oferuje ona rozgrywkę w statki przeciwko AI oraz przeciwko innym graczom w sieci (w przyszłej wersji).

2. Opis funkcjonalności:

- Gra przeciwko AI poziom 1 ✓
- Gra przeciwko AI poziom 2 ✗
- Gra przeciwko AI poziom 3 ✗
- Gra przeciwko graczowi w sieci ✗

3. Szczególnie interesujące zagadnienia projektowe:

System jest stworzony na wzorcu MVC-podobnym gdzie kontroler odświeża model, a renderer go wyświetla.

W projekcie zostały użyte:

- interfejsy, które pozwalają na narzucenie wymogów klasom implementującym je, dzięki czemu jest możliwa modularność np. renderowania, która zamiast terminala w przyszłości weźmie zaimplementowany obiekt JFrame
- klasy abstrakcyjne, dzięki którym można zgrupować wspólne funkcjonalności jak zaatakowanie planszy AI, a oddzielić poszczególne implementacje takie jak ruch AI przeciwko graczowi
- pętla gry wspierająca zmianę rozmiaru ekranu, pracująca na stanach i odświeżająca ekran tylko kiedy została wykonana jakaś akcja
- singleton, którym jest silnik gry jako przykład wykorzystania wzorca projektowego
- metoda fabrykująca zwracająca poszczególne kontrolery zależnie od odebranego stanu poprzedniego kontrolera
- operacje na bitach, które były niezbędne do następnego punktu czyli:
- Wave Function Collapse – własny pomysł oraz implementacja algorytmu wykorzystującego powyższą ideę. Użyte to zostało do generowania losowego ustawienia statków na planszy AI. Załóżmy, że mamy liczbę 8-bitową. Niech jej bity oznaczają rozmiar statku: 2 od lewej to długość 4, kolejne 2 to długość 3, kolejne to 2 i kolejne to długość 1. Lewy z nich oznacza orientację poziomą a prawy pionową. Następnie jest ustawiany stan początkowy całej tablicy czyli gdzie jaki statek może zostać wstawiony. Randomizer losuje pozycję i sprawdza czy zakolejkowany statek można tam postawić poprzez operację bitową, jeśli nie to losuje nową pozycję, jeśli tak to go stawia na finałowej planszy i modyfikuje ją tak, że ustawia 0 tam gdzie jest statek i wokół niego, a po lewej i górnej stronie tej wody modyfikuje bity komórek odpowiednio do możliwych stanów. Pierwsze dwa zdjęcia poniżej (lewo, prawo) reprezentują tablicę stanów i finałową planszę na początku. Po wylosowaniu pola D7 i orientacji pionowej, zmiany ukazują następne zdjęcia.

	1	2	3	4	5	6	7	8	9	10
A	255	255	255	255	255	255	255	191	175	171
B	255	255	255	255	255	255	255	191	175	171
C	255	255	255	255	255	255	255	191	175	171
D	255	255	255	255	255	255	255	191	175	171
E	255	255	255	255	255	255	255	191	175	171
F	255	255	255	255	255	255	255	191	175	171
G	255	255	255	255	255	255	255	191	175	171
H	127	127	127	127	127	127	127	63	47	43
I	95	95	95	95	95	95	95	31	15	11
J	87	87	87	87	87	87	87	23	7	3

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										

	1	2	3	4	5	6	7	8	9	10
A	255	255	255	255	255	95	95	31	175	171
B	255	255	255	255	255	87	87	23	175	171
C	255	255	191	175	171	0	0	0	175	171
D	255	255	191	175	171	0	0	0	175	171
E	255	255	191	175	171	0	0	0	175	171
F	255	255	191	175	171	0	0	0	175	171
G	255	255	191	175	171	0	0	0	175	171
H	127	127	63	47	43	0	0	0	47	43
I	95	95	95	95	95	95	95	31	15	11
J	87	87	87	87	87	87	87	23	7	3

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D							1			
E							1			
F							1			
G							1			
H										
I										
J										

4. Instrukcja instalacji:

Prosimy o poinformowanie twórcy o chęci instalacji gry ze względu na prywatne repozytorium. Wtedy twórca będzie mógł dodać taką osobę w celu klonowania repozytorium.

Po dodaniu do repozytorium możemy sklonować je poprzez polecenie:

```
~> git clone https://github.com/JakiuDeus/Battleships.git
```

Następnie wykonujemy poniższe polecenia:

```
~> cd Battleships
~> mvn clean package
~> cd target
Windows:
~> javaw -jar Battleships-0.0.jar
Linux:
~> java -jar Battleships-0.0.jar
```

Ważne informacje:

- Są problemy z odpaleniem tego programu w konsoli PowerShell lub też poprzez Windows 11 co wymaga dokładniejszego sprawdzenia. Jest to wina Fabryki Terminali w Lanternie, która przy prośbie o nowy obiekt dostosowanego Screena rzuca wyjątek IOException, którego nie da się w inny sposób obejść.
- Ten problem prawdopodobnie będzie można obejść przy przejściu na wersję graficzną z własną implementacją Swinga

5. Instrukcja konfiguracji:

Odpalanie z konsoli:

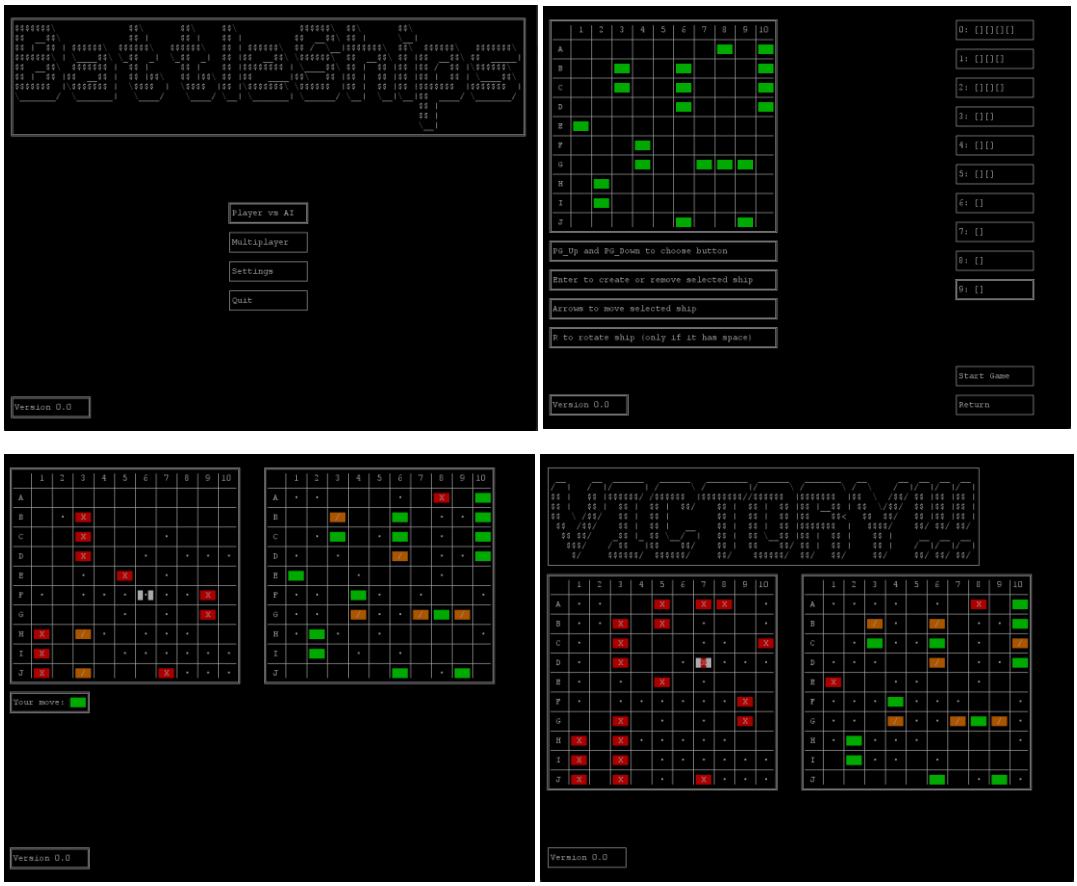
```
Windows:
~> javaw -jar Battleships.jar
Linux:
~> java -jar Battleships.jar
```

6. Instrukcja użytkownika:

Uwaga: Multiplayer oraz Settings są tymczasowo wyłączone ze względu na brak implementacji.

Widoki można podzielić na cztery rodzaje:

- Widok typu Menu
- Widok budowy
- Widok gry
- Widok wyniku



Gdy jesteśmy na widoku typu Menu poruszamy się po przyciskach strzałkami do góry i w dół, a przyciskiem Enter przechodzimy do dalszej części. Wybrany przycisk różni się obwódką od niewybranych przycisków.

W widoku budowania mamy następujące akcje:

- Page_UP oraz Page_DOWN służą do poruszania się po przyciskach z prawej strony
- Enter w przypadku statków kładzie go albo zabiera z planszy.
- Strzałki służą do ruszania wybranego statku, który jest na planszy
- R – służy do obracania statku. Obrót jest możliwy gdy możemy obrócić statek trzymając go za lewy górny róg. Można to przyrównać do zegara między godzinami 3 i 6.

Gdy statki najeżdżają na siebie lub się stykają bokami bądź rogami lub gdy nie wszystkie statki zostały użyte, gra przy starcie poinformuje nas że statki nie są poprawnie rozłożone. Należy je wtedy odpowiednio poprawić aby przestrzegały te reguły.

W widoku gry mamy widoczne dwie plansze. Pierwsza jest przeciwnika i my na niej wykonujemy ruchy, a druga jest nasza i przeciwnik wykonuje na niej ruchy. Pod planszą jest oznaczenie czyj jest ruch. Jeśli jest zielone to możemy strzelać, jeśli czerwone to jest to ruch przeciwnika. AI myśli przez sekundę przed ruchem. Poziom 1 strzela losowo, poziom 2 wyłapuje statki obok, poziom 3 ma własną mapę prawdopodobieństwa, którą z każdym ruchem aktualizuje prowadząc go do najszybszej wygranej.

W tym widoku sterujemy strzałkami po planszy gdzie rusza się nasz kursor. Gdy pole nie zostało odgadnięte możemy kliknąć Enter. Wtedy oddamy strzał i zostaniemy poinformowani poprzez kolor i

znak co się stało. Czarny i kropka oznacza pudło, Pomarańczowy i ‘/’ oznacza trafienie ale nie zatopienie. Jak statek zostanie zatopiony zmieni się on na kolor czerwony i znaki ‘X’.

W widoku wyniku widzimy obracający się napis VICTORY!!! lub DEFEAT!!! zależnie od wyniku, oraz tablice pokazujące ostatni wygląd przed zakończeniem. Po wciśnięciu Enter przejdziemy do ekranu głównego

7. Wnioski:

Projekt zajął powyżej 100 godzin „luźnej” roboty i nie wszystko co zostało planowane mogło zostać zrealizowane w podanym czasie. Mimo to struktura jest rozdzielona w taki sposób, że przy tworzeniu implementacji graficznej będzie można łatwo podmienić metody bez ingerencji w głębszy kod. Dodatkowo udało się wykorzystać bardziej zaawansowane koncepty programistyczne typu Wave Function Collapse oraz mapa prawdopodobieństwa do AI poziomu 3. Terminal też odświeża się tylko w częściach, które tego wymagają oraz reaguje na zmianę rozmiaru poprzez ponowne wyświetlenie zawartości.

8. Samoocena:

Co prawda nie udało się osiągnąć celu na 5 czyli funkcjonalności multiplayer, która wymagałaby oddzielnej aplikacji serwerowej zintegrowanej na serwisach AWS, natomiast jestem dumny ze struktury projektu oraz jego poszczególnych zaawansowanych konceptów jak i modularności. Docelowo wszystko co było planowane będzie zaimplementowane w wersji graficznej więc liczę na wysoki wynik.