

GOLD RUSH



Podstawowe wymagania i ogólny projekt

Jaki jest plan?

Określamy wymagania i zakres realizacji.

Musimy spróbować przewidzieć, jak finalnie ma wyglądać gra. Ponieważ nie znamy wielu szczegółów, na pewno nam się to nie uda.

Projektujemy bazową strukturę programu i planujemy interakcję między elementami.

Skupiamy się na tym, co wiemy, i staramy się zaprojektować podstawową strukturę, czyli "fundament" naszej aplikacji.

Planujemy zadania.

Na podstawie ustalonych wymagań i zakresu ich realizacji musimy zaplanować pracę.

Wyznaczamy zadania i decydujemy „co i kiedy”. Najlepiej uporządkować pracę, dzieląc czas na iteracje.

Zaczynamy programować!

Trzeba być cierpliwym. Najpierw trzeba wszystko przemyśleć, zaplanować, zamodelować, zweryfikować...

Ale ile można czekać!?

Wymagania

Przypomnę to, co zostało powiedziane na początku:

Z założenia ma to być sieciowa gra typu multiplayer. Gra rozgrywa się na kwadratowej planszy. Cel jest prosty: zdobyć jak najwięcej złota i pozbyć się konkurencji. Obowiązuje system turowy. Na polach rozmieszczane jest złoto oraz inne żetony odpowiadające pożywieniu, narzędziom, bonusom itp.

Rozbijmy to na punkty i włączmy krytyczne myślenie...

Wymagania

Ma to być **sieciowa** gra typu **multiplayer**.

Jeśli sieciowa, to przydałby się jakiś serwer gry. Co z nim?

Jakiś system rejestrowania, logowania? Wolny dostęp?

Gra rozgrywa się na **kwadratowej planszy**.

Jaki rozmiar planszy? Czy zawsze taki sam?

Cel jest prosty: **zdobyć** jak najwięcej złota i **pozbyć się** konkurencji.

Jak gracz wchodzi w interakcję z innymi elementami gry?

Obowiązuje **system turowy**.

W jakim sensie...?

Jakie zasady? Jak poruszają się gracze? W jakiej kolejności? Jak zrealizować spotkanie dwóch graczy?

Na polach **rozmieszczane jest złoto oraz inne żetony** odpowiadające pożywieniu, narzędziom, bonusom itp.

Kto i kiedy rozmieszcza te żetony? Jakie mają właściwości?

Czy wpływają tylko na stan gracza, czy też na jego zachowanie?

Fundamenty

Centralnym punktem jest **plansza**.

Plansza jest podzielona na **pola**.

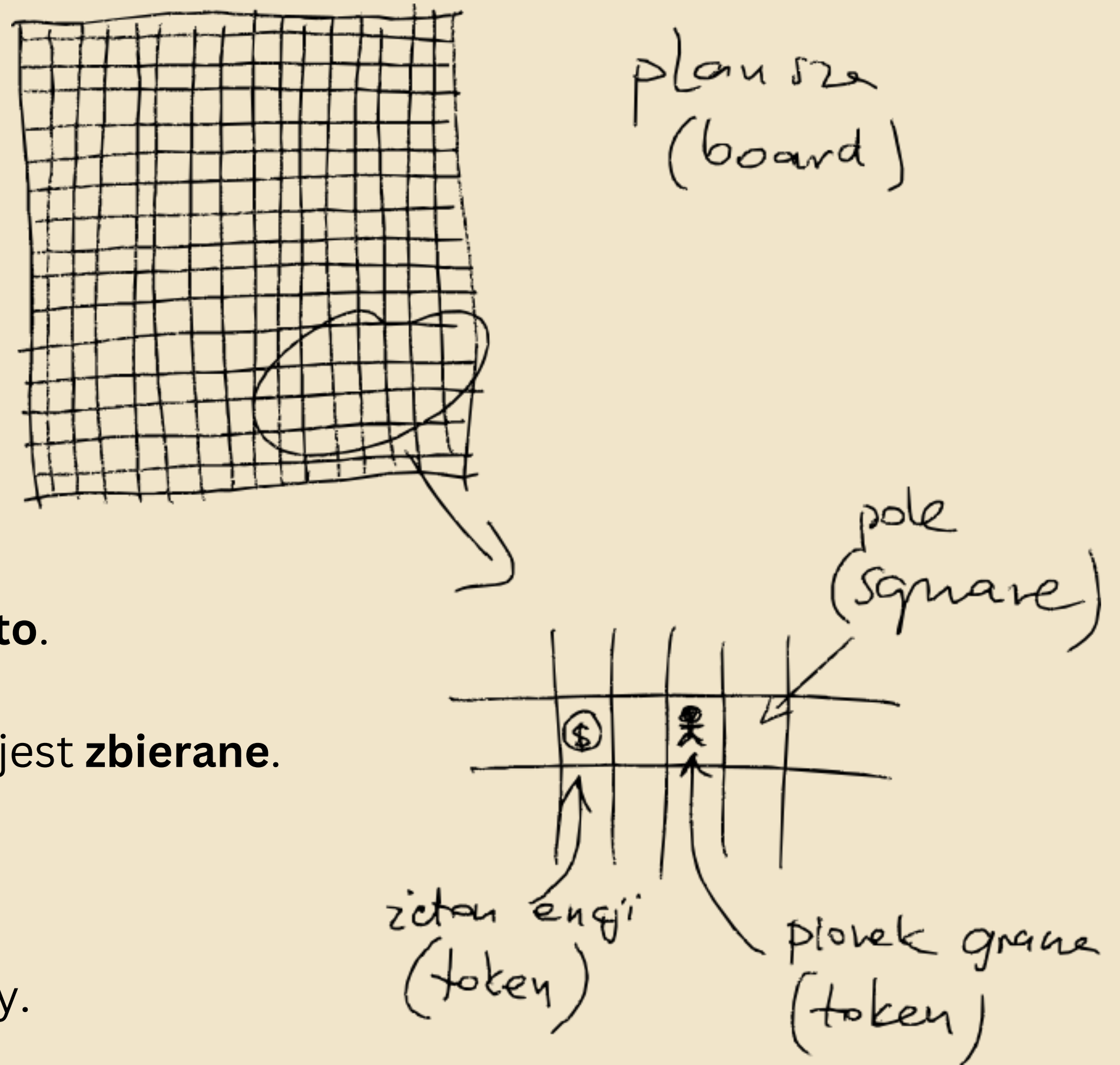
Na polu może znaleźć się **żeton**.

Żetonem może być **pionek gracza** albo **złoto**.

Po wejściu gracza na pole ze złotem, złoto jest **zbierane**.

Tyle na początek.

Wydzielmy na tej podstawie kluczowe klasy.



Może tak?



Za co mają odpowiadać poszczególne klasy?

mechanizm przesuwania pionka gracza

wyświetlanie planszy

pozycja żetonu (pionka)

cechy/zachowania encji (np. złota)

mechanizm rozmieszczania żetonów

właściwości gracza

inwentarz gracza

Pominęliśmy **Square**, zostały **Board** i **Token**:

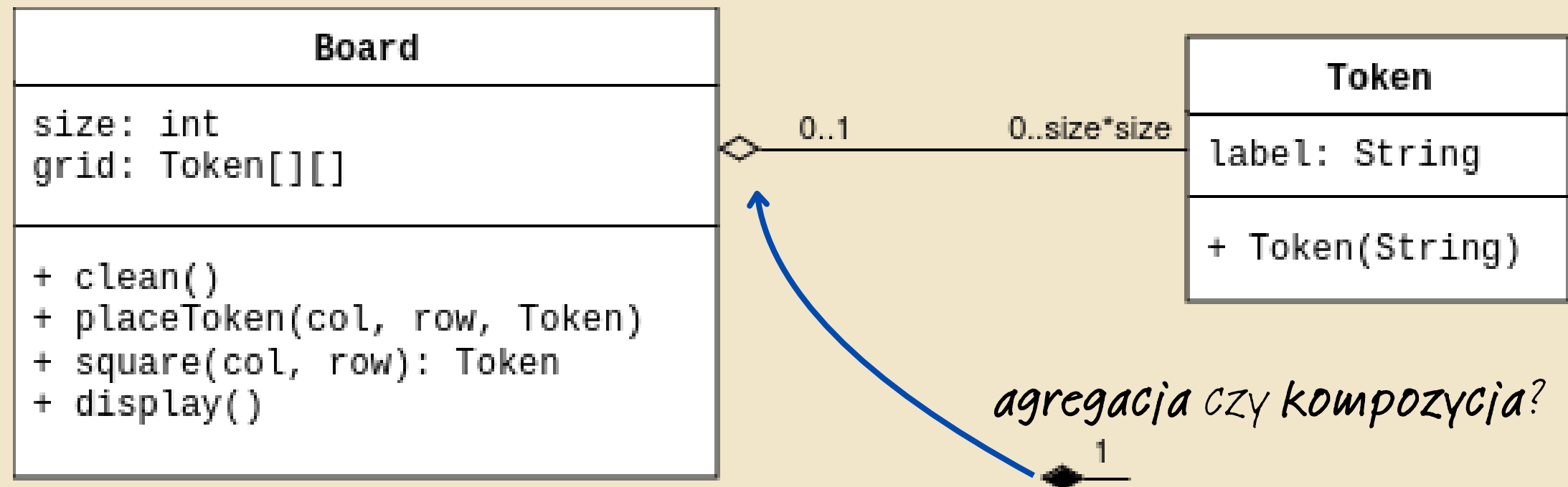


To zostawiamy na później:

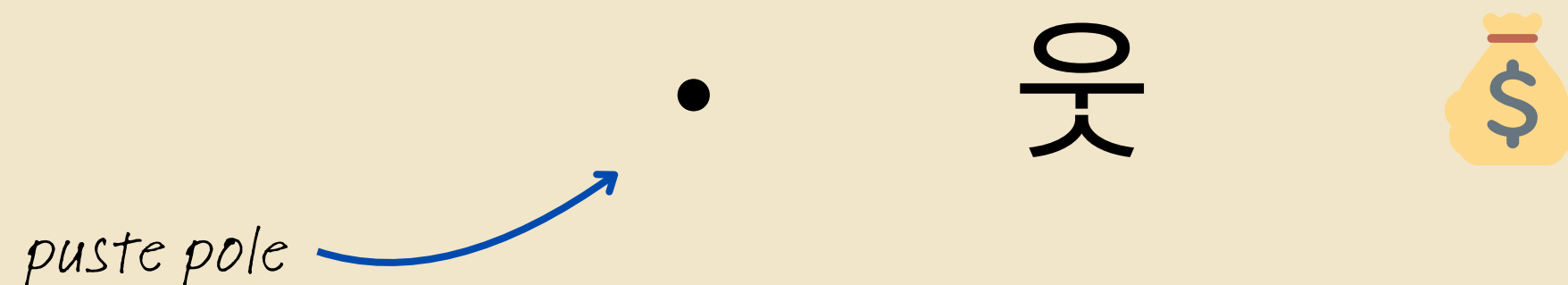
inwentarz gracza
właściwości gracza

Stworzymy osobną klasę na gracza

Więcej szczegółów:



Symbole* żetonów:



* Symbole i ich kody znajdziecie w repozytorium "resources"

Zadania

- Zdefiniować klasy **Board** i **Token**.
- W Board zaimplementować metody: **clean()**, **placeToken()**, **square()** i **display()**.
- Testowo umieścić na planszy **żeton złota** i **pionek gracza**.
- Uruchomić (i przejść) pierwsze testy jednostkowe (*i01w01*).

```
> Task :edu.io.Main.main()
```

```
• • • • • • • •
• • • • • • • •
• • • • • • • •
• • 웃 • • • • • •
• • • • • $ • •
• • • • • • • •
• • • • • • • •
• • • • • • • •
```

```
BUILD SUCCESSFUL in 169ms
```

Pozostałe zadania na pierwszą iterację:

- zaimplementować **poruszanie pionkiem** gracza
- wprowadzić **postać** gracza (a nie tylko pionka)
- zaimplementować **zbieranie złota**
- refaktoryzacja
- przygotować plan pracy na kolejną iterację