




## Tema - GwentStone



- Responsabili:  Florian-Luis Micu,  Andreea-Rebecca State,  Andrei Oțetea
- Deadline hard: 24 noiembrie, ora 23:59
- Data publicării: 7 noiembrie, ora 23:15
- Ultima modificare a cerinței: 18 noiembrie (clarificare rând și coloană pentru input)
- Ultima modificare a scheletului: 13 noiembrie (adăugare a coordonatelor "x" și "y" pentru "getCardAtPosition")

## Obiective

- Familiarizarea cu Java și conceptele de bază ale POO.
- Fundamentarea practică a constructorilor și a agregării/moștenirii.
- Dezvoltarea unor abilități de bază de organizare și design orientat obiect.
- Scrierea unui cod cât mai generic, ce va permite ulterior adăugarea de noi funcționalități.
- Respectarea unui stil de codare și de comentare.
- Dezvoltarea aptitudinilor de depanare a problemelor în cod.

## Introducere

Tocmai v-ați făcut primul vostru start-up. Fiind foarte pasionați de jocuri și de genul fantasy, ideea voastră de 1M de dolari este crearea unui joc de cărți pe calculator. Norocul vă surâde pentru că studioul CD Projekt Red pare că nu mai este în vogă în materie de jocuri așa că le puteți fura jocul de cărți, mai exact Gwent. Deoarece doriți ca jocul vostru să nu fie o copie care va fi ușor uitată, ați adăugat mecanisme noi care vor fi asemănătoare cu alte jocuri de cărți populare. Drept urmare, jocul vostru preia cele mai bune lucruri din **Hearthstone** și din **Gwent**, fiind mult mai ușor de implementat, dar și de jucat. De asemenea, ca în orice joc de calitate, o să ne asigurăm că avem și o poveste pe cînte pentru personaje noastre create.

## Date tehnice ale jocului

Dorim ca jocul nostru să poată fi jucat de doi jucători în sistem amical. Cu toate acestea, pentru că încă suntem în etapa de dezvoltare a jocului în cadrul studioului nostru OOP-Everything, vom folosi un AI care va simula cei doi jucători. AI-ul vă va da comenzile de joc în input, împreună cu server-ul care vă va trimite și comenzi de debugging. Comenzile de debugging sunt făcute pentru a verifica statutul jocului în diverse etape, astfel vom putea depana mai ușor problemele care pot apărea în cadrul implementării. De asemenea, server-ul vă solicită și date pe care le va folosi pentru a crea statistici, cum ar fi numărul de câștiguri ale unui jucător și numărul total de jocuri terminate de cei doi jucători. La output dorim să printăm orice informație ne cere server-ul sau orice eroare întâmpinăm în cadrul unei acțiuni invalide.

## Prezentare scurtă a gameplay-ului

Jocul vostru va fi jucat de către doi jucatori. Fiecare jucător are mai multe pachete de cărți numite deck-uri, iar fiecare dintre aceștia va trebui să își aleagă un pachet cu care va intra în joc. La începutul jocului, fiecărui jucător îi este asignat un erou care îl va reprezenta și se aruncă o monedă pentru a se stabili care jucător va începe. Jocul este "turn-based", deci fiecare jucător va avea câte o tură pe care o încheie explicit. După ce fiecare jucător își încheie tura, va începe o nouă rundă. La începutul rundei fiecare jucător primește "mana" care reprezintă valuta jocului apoi se trage o carte din deck și se introduce în mâna jucătorului. Cărțile din mâna unui jucător pot fi plasate pe masă pentru a fi folosite, dar pentru asta e nevoie de mana, deoarece fiecare carte "costă" mana pentru a putea fi plasată pe masa de joc. Masa de joc este un loc în care se plasează cărțile pentru a putea interacționa între ele. Fiecare jucător are 2 rânduri pe care poate pune o carte, cărțile fiind astfel făcute să poată fi plasate doar pe anumite rânduri. Cărțile care pot fi în mâna unui jucator sunt de doua tipuri: normale și environment. Cele normale pot fi plasate pe masă, cele environment afectează toate cărțile de pe un anumit rând. Eroii jucătorilor sunt prezenți pe masă de la începutul jocului într-un loc special și aceștia au abilități speciale care pot afecta rândurile cu cărți ale jucătorilor. De asemenea, cărțile normale de pe masă pot ataca eroii, astfel dacă un erou ajunge sa nu mai aibă viață se va termina jocul. Fiecare jucator are o metrică proprie în care se reține câte jocuri a câștigat și de câte ori a jucat un joc.

## Masa de joc

Masa de joc este locul unde cărțile vor interacționa direct între ele. Ea poate fi reprezentată ca o matrice de **4x5**, fiecare jucător având **2 rânduri** asignate lui. Cărțile se plasează pe câte un rând conform restricțiilor impuse mai jos, fiind introduse pe rând de la **stânga la dreapta**, mai exact adăugăm mereu o carte la dreapta ultimei cărți adăugate până se umple rândul. Dacă o carte este omorâtă, atunci toate cărțile din dreapta ei vor fi mutate cu o poziție la stânga (fenomen cunoscut ca **shiftare la stânga**). De asemenea, pe un rând pot exista **maxim 5** cărți.

Rândurile 0 și 1 sunt asignate jucătorului 2, iar rândurile 2 și 3 sunt asignate jucătorului 1, conform imaginii de mai jos. Rândurile din față vor fi reprezentate de rândurile 1 și 2, iar rândurile din spate vor fi 0 și 3 (jucătorii vor fi așezați față în față). Totodată, eroii jucătorilor vor avea un loc special în care vor fi așezați de la începutul jocului.

### Administrativ

- Regulament
- Calendar
- Echipa
- Orar
- Recomandări cod
- Indicații pentru teme
- Catalog

### Laboratoare

- Setup environment
- Laboratorul 1: Java basics
- Laboratorul 2: Constructori și referințe
- Laboratorul 3: Agregare și moștenire
- Laboratorul 4: static și final; Singleton pattern
- Laboratorul 5: Abstractizare
- Laboratorul 6: Clase interne
- Laboratorul 7: Overriding, overloading & Visitor pattern
- Laboratorul 8: Colecții
- Laboratorul 9: Design patterns - Factory, Strategy, Observer
- Laboratorul 10: Design patterns - Command și Builder
- Laboratorul 11: Genericitate
- Laboratorul 12: Java features
- Laboratorul 13: Excepții

### Teme

- Tema - GwentStone

### Teste

- Teste grilă

### Resurse utile

- Instalare IntelliJ Idea
- Activare IntelliJ Idea
- Demo proiect IntelliJ Idea
- Tutorial Git in IntelliJ Idea
- Tutorial Git
- Tutorial checklist
- Script pentru LambdaChecker

### Alte resurse

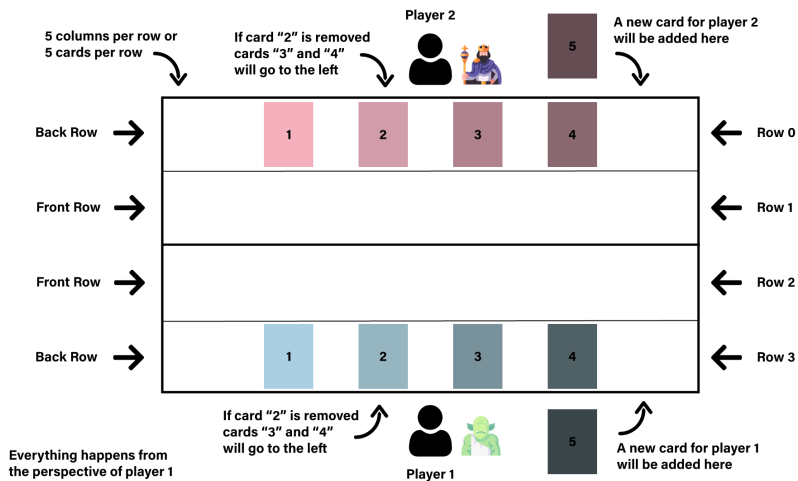
- Laborator recapitulare
- Exerciții vechi
- POO și Java
- JUnit5
- Organizarea surselor și controlul accesului
- Tutorial I/O
- JSON & Jackson
- Double Dispatch - scurt tutorial
- Reflection

### Arhiva

- 2021-2022
- 2020-2021
- 2019-2020
- 2018-2019
- 2017-2018
- 2016-2017
- 2015-2016
- 2014-2015
- 2013-2014

### Table of Contents

- Tema - GwentStone
  - Obiective
  - Introducere
  - Date tehnice ale jocului
  - Prezentare scurtă a gameplay-ului
  - Masa de joc
    - Formatul cărților de joc
      - Cartea Minion
      - Cartea Environment



În teste anumite acțiuni necesită coordonate ale cărților de pe tablă. Acest coordonatele vă sunt date folosind parametrii "x" (rând) și "y" (coloană).

## Formatul cărților de joc

Există mai multe tipuri de cărți pe care un jucător le poate deține, mai exact un jucător poate avea o carte de tipul **minion** sau o carte de tipul **environment**. De asemenea, fiecare jucător va avea o carte specială de tipul **erou** care îl va reprezenta în joc.

### Cartea Minion

Aceste cărți au la bază atribute precum: **mana** (intreg), **health** (intreg), **attackDamage** (intreg), **description** (String), **colors** (String/Enum) și **name** (String). *Mana* este **costul** pe care îl va avea o carte pentru a putea fi folosită în timpul jocului, *health* reprezintă viața unei cărți, *attackDamage* reprezintă punctele de atac cu care o carte poate ataca altă carte, *description* este o descriere scurtă a cărții, *colors* reprezintă culorile care alcătuiesc design-ul grafic al cărții și *name* este numele cărții. Aceste atribute **le veți primi la input**, fiecare carte având un set de valori specifice care vor fi **randomizate**. Drept urmare, două cărți cu același nume pot avea proprietăți diferite.

Din acest tipar fac parte cărțile următoare: **Sentinel**, **Berserker**, **Goliath** și **Warden**.



- O carte nu poate ataca de două ori în cadrul aceleiași ture a jucătorului.
- Nu se poate folosi atacul unei cărți care este "frozen" (frozen = a sta o tură).
- Atacul unei cărți trebuie să fie aplicat doar pe o carte a adversarului.
- Erorile de mai sus sunt valide și pentru cazul când o carte de pe masă dorește să atace eroul inamicului.

Pe lângă aceste cărți, mai există și cărți care posedă abilități speciale. Aceste cărți sunt: **Miraj**, **The Ripper**, **Disciple** și **The Cursed One**. Abilitățile speciale ale acestor cărți vor fi targetate către alte cărți pentru a activa următoarele efecte:

#### The Ripper

**Weak Knees:** -2 atac pentru un minion din tabăra adversă.

#### Miraj

**Skyjack:** face swap între viața lui și viața unui minion din tabăra adversă.

#### The Cursed One

**Shapeshift:** face swap între viața unui minion din tabăra adversă și atacul aceluiași minion

#### Disciple

**God's Plan:** +2 viață pentru un minion din tabăra lui.



- Cărțile **The Cursed One** și **Disciple** au atacul 0, deoarece în povestea noastră ei reprezintă niște druzii. În schimb, **The Ripper** și **Miraj** au atacul diferit de 0 fiind minioni legendari. Chiar dacă druzii au atacul 0, ei pot ataca pe oricine.
- Abilitățile nu sunt date la input, ele fiind fixate pentru fiecare carte. Trebuie ca voi să implementați efectul dorit atunci când se folosește abilitatea specială a unei cărți.
- Cărțile **The Ripper**, **Miraj**, **Goliath** și **Warden** trebuie puse pe rândul din **față** asociat jucătorului curent.
- Cărțile **Sentinel**, **Berserker**, **The Cursed One** și **Disciple** trebuie puse pe rândul din **spate** asociat jucătorului curent.
- Cărțile cu aceste abilități pot să atace sau să își folosească abilitatea, dar nu pot să le facă pe ambele în cadrul aceluiași "turn" al jucătorului.

- Cartea Erou
- Gameplay detaliat
  - Pregătirea jucătorilor
  - Pregătirea jocului
  - Începutul unei runde
  - Sfârșitul unei ture
  - Plasarea unei cărți pe masă
  - Comanda de atac a unei cărți
  - Comanda de utilizare a abilității unei cărți
  - Comanda de utilizare a atacului unei cărți asupra eroului
  - Comanda de utilizare a abilității unui erou
  - Comanda de utilizare a unei cărți environment
- Comenzi de debug
  - Aflarea cărților din mână
  - Aflarea cărților din pachet
  - Afișarea cărților de pe masă
  - Aflarea jucătorului activ
  - Aflarea eroului unui jucător
  - Afișarea unei cărți la o poziție dată
  - Afișarea manei unui jucător
  - Afișarea cărților de environment din mână unui jucător
  - Afișarea cărților de pe masă care sunt "frozen"
- Comenzi pentru statistici
  - Afișarea numărului total de jocuri jucate
  - Afișarea numărului total de jocuri câștigate
- Scheletul de cod
- Execuția temei
- Recomandări
- Evaluare
  - Checkstyle
  - Upload temă
  - FAQ
- Resurse și linkuri utile

- **Weak Knees, Skyjack** și **Shapeshift** trebuie să fie aplicate doar pe o carte a adversarului.
- **Weak Knees** aplicat pe o carte cu atacul < 2 scade până la 0 atacul(adică atacul nu poate fi negativ)
- **God's Plan** se va folosi doar pe o carte a jucătorului curent.
- **Shapeshift** aplicat pe o carte care are atacul 0 va face ca viața minionului pe care este aplicată abilitatea să devină 0, astfel cartea va fi **omorâtă**.
- Nu se poate folosi abilitatea unei cărți care este "frozen".
- Jucătorul trebuie să aibe destulă mana pentru a plasa o carte minion pe masă.

Cărțile **Goliath** și **Warden** au o abilitate specială pasivă a unor cărți numite adesea în jocuri "Tank", mai exact când sunt plasate pe masă, ele vor trebui mereu **atacate prima oară**, deoarece în povestea noastră ei reprezintă campioni care au fost mereu loiali celorlalți și care sunt antrenati să reziste loviturilor.



ex. Dacă player-ul 1 vrea să atace sau să folosească o abilitate pe o carte a player-ului 2, trebuie să verificăm dacă pe rândul 1 se află cel puțin un Tank. Dacă se află un Tank, trebuie selectat oricare dintre aceștia. Pentru ca player 1 să poată să atace alți minioni care nu sunt Tank, trebuie mai întâi să se omoare toți minionii adversarului de tipul Tank. Totodată, dacă player 1 vrea să atace eroul player-ului 2 trebuie să verificăm dacă nu există un Tank pe rândul 1. Eroul inamicului poate fi atacat doar dacă acesta nu are cărți de tipul Tank pe masă.

## Cartea Environment

Aceste cărți au la bază atribute precum: **mana, description, colors** și **name**. Ele reprezintă efecte care pot fi aplicate terenului de joc, în povestea noastră fiind efecte naturale ale unei lumi din povești. Fiecare efect ia în considerare **toate cărțile de pe rândul targetat**. Cărțile de tip environment sunt următoarele:

### Firestorm

Scade cu 1 viața tuturor minionilor de pe rând.

### Winterfell

Toate cărțile de pe rând stau o tură.

### Heart Hound

Se fură minionul adversarului cu cea mai mare viață de pe rând și se pune pe rândul "oglinduit" aferent jucătorului.



- Cărțile de acest tip **nu sunt puse pe masă**, sunt folosite direct.
- După ce cartea a fost folosită, aceasta va fi **ștearsă din mâna jucătorului** și nu va reveni în deck-ul jucătorului.
- Abilitățile nu sunt date la input, ele fiind fixate pentru fiecare carte de tipul environment. Trebuie ca voi să implementați efectul dorit atunci când se folosește abilitatea specială a cărții.
- Toate aceste abilități trebuie aplicate doar pe un **rând al inamicului**.
- Jucătorul trebuie să aibe destulă mana pentru a juca o carte de tip environment.
- Pentru cartea **Heart Hound**, trebuie verificat dacă jucătorul are loc să plaseze cartea furată pe rândul său aferent tipului de carte furat (ex. dacă s-a furat o carte de pe rândul 1, aceasta va trebui pusă pe rândul 2).

## Cartea Erou

Există 4 eroi pe care un jucator îi poate avea. Fiecare erou are următoarele atribute: **mana, description, colors** și **name**. Fiecare erou are parametrul **health** fixat la 30. Dacă eroul ajunge în timpul jocului să fie atacat și să aibe viața 0, jocul se va opri și va câștiga atacatorul. Eroul fiecărui jucător este specificat în input la începutul jocului. Totodată, ei au abilități speciale care sunt declanșate pe un rând:

### Lord Royce

**Sub-Zero**: îngheață cartea cu cel mai mare atac de pe rând.

### Empress Thorina

**Low Blow**: distruge cartea cu cea mai mare viață de pe rând.

### King Mudface

**Earth Born**: +1 viață pentru toate cărțile de pe rând.

### General Kocioraw

**Blood Thirst**: +1 atac pentru toate cărțile de pe rând.



- Eroul jucătorului este ales random de către joc și vă va fi comunicat în input.
- Abilitățile nu sunt date la input, ele fiind fixate pentru fiecare carte. Trebuie ca voi să implementați efectul dorit atunci când se folosește abilitatea specială a unei cărți de tipul erou.
- Abilitățile **Sub Zero** și **Low Blow** trebuie targetate doar pe un rând al inamicului.
- Abilitățile **Earth Born** și **Blood Thirst** trebuie targetate doar pe un rând aliat.

- Jucătorul trebuie să aibe destulă mana pentru a folosi abilitatea unui erou.
- Eroul jucătorului poate să atace o singură dată per tură.



Pentru aceste abilități se iterează de la stânga la dreapta și pentru **Low Blow** și **Sub-Zero** se ia prima carte de pe rând care respectă criteriul specificat mai sus (ex. dacă cartea cu indexul 0 și cartea cu indexul 3 au atacul maxim de pe rând, se va selecta cartea cu indexul 0).

## Gameplay detaliat

### Pregătirea jucătorilor

Fiecare jucător are mai multe pachete cu care poate începe jocul, prima lui sarcină fiind să aleagă pachetul cu care dorește să joace. Veți primi la input numărul de pachete de cărți, numărul de cărți din fiecare pachet, o listă cu pachetele de cărți și jucătorul pentru care se aplică acești parametri.

Click pentru input-ul asociat acestui pas ↗



Acest pas se întâmplă doar când se începe un test nou. Dacă jucătorii doresc să joace alt joc în cadrul aceluiași test, AI-ul vă va specifica doar ce pachet doresc să folosească pentru următorul meci, astfel **repetându-se pasul următor**. Ordinea pachetelor rămâne aceeași.



La finalul unei partide dintre cei doi jucători aceștia trebuie să aibă posibilitatea să aleagă același pachet ca în jocul trecut pentru un nou joc, deci va trebui ca pachetul folosit să poată să revină **exact** la starea inițială, înainte de procesul de amestecare.

### Pregătirea jocului


Fiecare jucător își alege un deck, acesta fiind specificat de AI în input. Deck-ul ales de fiecare jucător va fi amestecat la începutul jocului, deoarece dorim ca jocul să nu fie previzibil. Pentru a ne asigura că rezultatele vor fi deterministe, server-ul va trimite la input și un **seed** folosit ca parametru pentru procesul de amestecare.

Server-ul alege câte un erou aleator pentru fiecare jucător, aceștia fiind specificați la input.

Se alege un jucător care să înceapă prima rundă, fiind specificat la input.

Click pentru input-ul asociat acestui pas ↗



Pentru amestecarea cărților, vă recomandăm să folosiți metodele din clasa Random  Shuffle.



Pentru a avea pachetul amestecat cum trebuie de fiecare dată trebuie să **reinstantiați** clasa Random **la fiecare amestecare**.

### Începutul unei runde

La începutul fiecărei runde, ambii jucători primesc prima carte disponibilă din pachetul de cărți ales. Cartea primită de fiecare jucător va fi adăugată în "mâna" acestuia. Cartea primită este adăugată la sfârșitul listei de cărți din mână.



**Dacă nu mai există cărți în pachet, nu mai trebuie trasă nicio carte în mână.**

De asemenea, ambii jucători primesc mana la începutul fiecărei runde. Mana pe care o primesc jucătorii în prima rundă este 1, urmând ca aceasta să fie incrementată până la 10. În momentul în care se ajunge la 10 mana, aceasta nu va mai fi incrementată.



**Exemplu primire mana** Să presupunem că jucătorii nu folosesc deloc mana. În prima rundă fiecare jucător primește 1 mana. După ce își fac cei 2 tura, începe runda a 2-a. Fiecare jucător primește 2 mana, ceea ce înseamnă că acum ambii jucători au **3 mana**. După ce își fac ambii tura, începe runda a 3-a. Cei 2 jucători primesc 3 mana, ceea ce înseamnă că fiecare are **6 mana** acum ș.a.m.d.



Prima carte care va fi trasă de jucător va fi cea de la **începutul** listei de cărți amestecate ale jucătorului. Mai exact, cea cu index-ul 0 în lista de cărți din deck.



- O rundă este formată din două ture, una per jucător.
- În cadrul unei ture pot fi făcute mai multe acțiuni de către jucătorul curent.

## Sfârșitul unei ture

La finalul turei unui jucător, cărțile acestuia care au fost marcate ca fiind "frozen" sunt demarcate, întrucât acestea au stat o tură.

Totodată, se verifică dacă ambii jucători și-au sfârșit turele în cadrul rundei curente. Dacă această condiție este adevărată, se trece la următoarea rundă și se aplică pașii descriși mai sus.

Marcarea sfârșitului unei ture va fi precizată explicit de AI în input pentru jucătorul curent.

Click pentru input-ul asociat acestui pas ↗

## Plasarea unei cărți pe masă

Cărțile din mână pot fi plasate pe masă, cu condiția ca acestea să nu fie cărți de tipul environment și ca acestea să nu aibă costul de mana mai mare decât mana jucătorului.

Cărțile trebuie plasate pe masă, pe rândul aferent jucătorului, dar și al tipului de carte (conform secțiunilor de mai sus).

AI-ul vă va preciza în input indexul cărții din mână pe care dorește să o plaseze pe masa.

Cazurile invalide trebuie testate la acest pas în această ordine:

1. Se verifică dacă cartea nu este de tipul environment, altfel se printează: **"Cannot place environment card on table."**
2. Se verifică dacă cartea are costul mai mic decât mana jucătorului, altfel se printează: **"Not enough mana to place card on table."**
3. Se verifică dacă rândul pe care trebuie plasată cartea nu este plin, altfel se printează: **"Cannot place card on table since row is full."**

Click pentru input-ul asociat acestui pas ↗

Click pentru output în cazul unei erori ↗



- Un jucător poate plasa toate cărțile din mână în cadrul aceleiași ture dacă îndeplinește condițiile precizate.
- În input nu se precizează mâna cărui jucător trebuie actualizată, drept urmare aceasta comandă va fi activată pentru jucătorul curent.

## Comanda de atac a unei cărți

Fiecare carte plasată pe masă poate ataca altă carte care se află pe masa. În acest pas, se dorește ca viața minion-ului adversar țintit să scadă cu atâtea puncte câte puncte are cartea atacatorului în câmpul **attackDamage**. Dacă viața cărții atacate ajunge să fie  $\leq 0$ , aceasta este **eliminată** de pe masă. O carte poate ataca o altă carte o singură dată per tură. După ce cartea și-a folosit atacul, aceasta nu va mai putea ataca altă carte sau erou decât tura următoare.

AI-ul va specifica în input coordonatele de pe masă a cărții atacate și a cărții atacatorului.

Cazurile invalide care trebuie testate în acest pas sunt:

1. Se verifică dacă cartea atacată este o carte a inamicului, altfel se printează: **"Attacked card does not belong to the enemy."**
2. Se verifică dacă cartea atacatorului nu a atacat/folosit abilitatea specială deja tura aceasta, altfel se printează: **"Attacker card has already attacked this turn."**
3. Se verifică dacă cartea atacatorului nu este "frozen", altfel se printează: **"Attacker card is frozen."**
4. Se verifică dacă există o carte Tank pe rândurile adversarului.
  - a. Se verifică mai departe dacă aceasta a fost selectată pentru atac, altfel se printează: **"Attacked card is not of type 'Tank'."**

Click pentru input-ul asociat acestui pas ↗

Click pentru output în cazul unei erori ↗



- Dacă o carte este omorâtă, aceasta nu se introduce înapoi în deck-ul jucătorului atacat.
- Dacă adversarul are mai multe cărți de tipul Tank pe rândurile lui, atunci atacatorul trebuie să atace una din cărțile de tipul Tank.

## Comanda de utilizare a abilității unei carti

Fiecare carte plasată pe masă își poate folosi abilitatea pe altă carte care se află pe masă. Fiecare abilitate specială va avea un efect diferit asupra celui atacat, abilitățile fiind descrise și asociate cu anumiți minioni conform secțiunilor de mai sus.

AI-ul va specifica în input coordonatele de pe masă a cărții atacate și a cărții atacatorului.

Cazurile invalide trebuie testate la acest pas în această ordine:

1. Se verifică dacă cartea atacatorului nu este "frozen", altfel se printează: **"Attacker card is frozen."**
2. Se verifică dacă cartea atacatorului nu a atacat/folosit abilitatea specială deja tura aceasta, altfel se printează: **"Attacker card has already attacked this turn."**
3. Se verifică dacă atacatorul este Disciple.
  - a. Se verifică dacă cartea atacată este o carte aliată, altfel se printează: **"Attacked card does not belong to the current player."**
4. Se verifică dacă atacatorul este The Ripper, Miraj sau The Cursed One.
  - a. Se verifică dacă cartea atacată este o carte a inamicului, altfel se printează: **"Attacked card does not belong to the enemy."**

- b. Se verifică dacă există o carte Tank pe rândurile adversarului.

1. Se verifică mai departe dacă aceasta a fost selectată pentru atac, altfel se printează: **"Attacked card is not of type 'Tank'."**

Click pentru input-ul asociat acestui pas ↗

Click pentru output în cazul unei erori ↗



- Se garantează că în input atacatorul este o carte care are abilitate specială.
- Dacă atacatorul este Disciple, nu trebuie să se verifice că minionul atacat este Tank.
- Dacă o carte este omorâtă (caz posibil pentru abilitatea **Shapeshift**), aceasta nu se introduce înapoi în deck-ul jucătorului atacat.
- Dacă adversarul are mai multe cărți de tipul Tank pe rândurile lui, este suficient ca cel atacat să fie oricare dintre aceste cărți.

## Comanda de utilizare a atacului unei cărți asupra eroului

Fiecare carte plasată pe masă poate ataca eroul inamicului. În acest pas, se dorește ca viața eroului să scadă cu atâtea puncte câte are atacatorul atributul **attackDamage**. La finalul comenzii se verifică dacă eroul inamicului a murit, în acest caz jocul fiind câștigat de jucătorul curent.

AI-ul va specifica în input coordonatele de pe masă a cărții atacatorului.

Cazurile invalide trebuie testate la acest pas în această ordine:

- Se verifică dacă cartea atacatorului nu este "frozen", altfel se printează: **"Attacker card is frozen."**
- Se verifică dacă cartea atacatorului nu a atacat/folosit abilitatea specială deja tura aceasta, altfel se printează: **"Attacker card has already attacked this turn."**
- Se verifică dacă există o carte Tank pe rândurile adversarului.
  - Se verifică mai departe dacă aceasta a fost selectată pentru atac, altfel se printează: **"Attacked card is not of type 'Tank'."**



În urma atacului, se va verifica dacă eroul adversar mai este în viață. Dacă eroul adversar a fost învins, atunci se va afișa în funcție de caz: **"Player one killed the enemy hero."** sau **"Player two killed the enemy hero."**

Click pentru input-ul asociat acestui pas ↗

Click pentru output în cazul unei erori ↗

Click pentru output în cazul în care unul din eroi este învins ↗



- Se permit doar atacuri directe către eroi, nu și folosirea abilităților speciale pe eroi.
- Nu se permite atacarea propriilor eroi.



Dacă un erou a fost omorât, jocul curent se termină instant. Mai exact, nu se va mai schimba nici tura și nici runda, iar fiecare jucător nu va mai primi mana sau cărți.

## Comanda de utilizare a abilității unui erou

Fiecare erou poate ataca un rând de pe masă. Abilitățile eroilor sunt descrise în secțiunile de mai sus, aceștia având posibilitatea să atace doar minioni ai adversarului sau minioni aliați în funcție de abilitate.

AI-ul vă specifică în input rândul ales pentru abilitate.

Cazurile invalide trebuie testate la acest pas în această ordine:

- Se verifică dacă jucătorul are destulă mana ca să folosească eroul, altfel se printează: **"Not enough mana to use hero's ability."**
- Se verifică dacă eroul nu a atacat deja tura aceasta, altfel se printează: **"Hero has already attacked this turn."**
- Se verifică dacă eroul este **Lord Royce** sau **Empress Thorina**.
  - Se verifică mai departe dacă rândul selectat pentru abilitate este un rând al adversarului, altfel se printează: **"Selected row does not belong to the enemy."**
- Se verifică dacă eroul este **General Kocioraw** sau **King Mudface**.
  - Se verifică mai departe dacă rândul selectat pentru abilitate este un rând al jucătorului curent, altfel se printează: **"Selected row does not belong to the current player."**

Click pentru input-ul asociat acestui pas ↗

Click pentru output în cazul unei erori ↗



- Se garantează în input că eroii nu vor ataca rânduri goale.
- Abilitățile eroilor nu iau în considerare prezența cărților de tip Tank.
- Dacă se folosește abilitatea **Sub-Zero** pe o carte care este deja marcată ca fiind frozen, aceasta va rămâne tot frozen doar o tura.
- Dacă o carte este omorâtă (caz posibil pentru abilitatea **Low Blow**), aceasta nu se introduce înapoi în deck-ul jucătorului.

## Comanda de utilizare a unei cărți environment

Cărțile de tip environment sunt ținute în mâna jucătorilor, acestea putând fi aplicate asupra unui rând al mesei. Abilitatea unei cărți de environment este fixată pentru fiecare carte de acest tip și este descrisă în secțiunile de mai sus.

AI-ul va specifica în input rândul ales pentru efectul cărții de environment și indexul cărții din mână.

Cazurile invalide trebuie testate la acest pas în această ordine:

1. Se verifică dacă indexul ales corespunde unei cărți de tipul environment în lista de cărți din mână, altfel se printează: **"Chosen card is not of type environment."**
2. Se verifică dacă jucătorul are destulă mana ca să folosească cartea de environment, altfel se printează: **"Not enough mana to use environment card."**
3. Se verifică dacă rândul afectat este al inamicului, altfel se printează: **"Chosen row does not belong to the enemy."**
4. Pentru cartea **Heart Hound**, se verifică dacă rândul oglindă al cărții targetate nu este deja full pentru jucătorului curent, altfel se printează: **"Cannot steal enemy card since the player's row is full."**

Click pentru input-ul asociat acestui pas ↗

Click pentru output în cazul unei erori ↗



- După ce cartea de environment a fost folosită, aceasta va trebui să fie ștearsă din mâna jucătorului și nu va fi reintrodusă în deck-ul acestuia.
- Dacă se folosește abilitatea **Winterfell** pe o carte care este deja marcată ca fiind frozen, aceasta va rămâne tot frozen doar o tură.
- Dacă o carte este omorâtă (caz posibil pentru abilitatea **Firestorm**), aceasta nu se introduce înapoi în deck-ul jucătorului.
- Dacă se folosește abilitatea **Heart Hound** și rândul aferent cărții furate este plin pentru jucătorul curent, cartea nu va mai fi furată și cartea "Heart Hound" folosită nu va fi ștearsă din mâna jucătorului.

## Comenzi de debug

Pe parcursul jocului, server-ul vă poate cere oricând anumiți parametri pentru a verifica implementarea voastră.

### Aflarea cărților din mână

Pentru această comandă trebuie să puneți în output toate cărțile care se află în mâna jucătorului dat ca parametru la input.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

### Aflarea cărților din pachet

Pentru această comandă trebuie să puneți în output toate cărțile care se află în deck-ul jucătorului dat ca parametru la input.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

### Afișarea cărților de pe masă

Pentru această comandă trebuie să afișați la output toate cărțile prezente pe masă. Cărțile trebuie printate începând de la rândul 0, coloana 0 până la rândul 3, coloana 4, conform pozei de mai sus.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗



- Cărțile de pe masă se vor printa sub forma unei liste începând de la rândul 0 până la rândul 3, de la stânga la dreapta.

### Aflarea jucătorului activ

Pentru această comandă trebuie să printați la output jucătorul activ (adică jucătorul care nu și-a încheiat încă tura).

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

### Aflarea eroului unui jucător

Pentru această comandă trebuie să printați la output eroul jucătorului dat ca parametru la input.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

### Afișarea unei cărți la o poziție dată

Pentru această comandă trebuie să printați la output cartea aflată la poziția dată în input.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

Click pentru un exemplu de eroare pentru acest pas ↗



Dacă nu există o carte la poziția dată, veți printa "No card available at that position."



## Afișarea manei unui jucător

Pentru această comandă trebuie să printați la output mana unui jucător primit la input.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

## Afișarea cărților de environment din mâna unui jucător

Pentru această comandă trebuie să printați la output toate cărțile de environment prezente în mâna jucătorului dat ca input. Ordinea cărților de environment trebuie să fie identică cu ordinea lor din mâna jucătorului.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗



Dacă nu există cărți de environment în mană, veți afișa o listă goală.

## Afișarea cărților de pe masă care sunt "frozen"

Pentru această comandă trebuie să printați la output toate cărțile de pe masă care au flag-ul "frozen" setat. Ordinea printării trebuie să respecte ordinea cărților de pe masă, mai exact se începe de la linia 0, coloana 0 și se termină la linia 3, coloana 4.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗



Dacă nu există cărți pe masă care să fie "frozen", veți afișa o listă goală.

## Comenzi pentru statistici

Statisticile sunt reprezentate de numărul total de jocuri la care a participat fiecare jucator și numărul total de câștiguri

### Afișarea numărului total de jocuri jucate

Pentru această comandă va trebui să afișați la output numărul total de jocuri jucate de ambii jucători.

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

### Afișarea numărului total de jocuri câștigate

Pentru această comanda va trebui să afișați la output numărul total de jocuri câștigate de un jucător. Vor exista două comenzi pentru fiecare jucător ("getPlayerOneWins" și "getPlayerTwoWins").

Click pentru input-ul asociat acestui pas ↗

Click pentru un exemplu de output pentru acest pas ↗

## Scheletul de cod

În rezolvarea temei va fi nevoie de folosirea unor obiecte pentru stocarea și maparea datelor primite în format JSON. Scheletul temei vă oferă mai multe clase utile pentru citirea și rularea temei. Acestea se regăsesc în cadrul scheletului astfel:

- Clasele de input din cadrul pachetului fileio : Acestea se vor folosi pentru parsarea datelor de input. Astfel preluați toate informațiile necesare pentru a crea propriile clase pentru rezolvarea temei.
- Clasa Main care este punctul de intrare pentru logica de rezolvare a temei.

Pentru a înțelege mai bine cum funcționează citirea/scrie în fișierele JSON vă recomandăm să citiți [JSON & Jackson](#).



Output-ul nu trebuie formatat ca în ref-uri, fiindcă se verifică conținutul obiectelor și array-urilor JSON, nu textul efectiv. Cu toate acestea, dacă folosiți Jackson, vă recomandăm să utilizați **PrettyPrinter** [Documentație](#) [PrettyPrinter](#). Totodată, pentru a înțelege cum se poate realiza **scrierea în fișierele JSON de output**, vă sugerăm să consultați [JSON Array](#).



Aveți în folder-ul **"lib"** toate dependențele necesare pentru rularea temei, mai exact bibliotecile Jackson.

## Execuția temei

1. Se încarcă datele citite din fișierul de test, în format JSON, în obiecte.
2. Se primesc secvențial acțiuni și se execută pe măsură ce sunt primite, rezultatul lor având efect asupra Repository-ului.
3. După executarea unei acțiuni, se afișează rezultatul ei în fișierul JSON de ieșire.
4. La terminarea tuturor acțiunilor se termină și execuția programului.





- După ce clonați repo-ul de pe GitHub, vă rugăm să vă faceți un repository propriu privat în care să vă puneți doar conținutul folder-ului **"tema"** de pe repo-ul echipei de POO. Dacă nu puneți folder-ul cu tema la alta cale, **nu o să puteți** să faceți schimbări în Git, deoarece vă aflați în rădăcina repository-ului echipei de POO.
- Pentru ca checker-ul să funcționeze trebuie să deschideți tema din IntelliJ la calea unde se află folderele **"src"**, **"lib"**, **"ref"**, **"input"**. Aveți folder-ul **.idea** pregenerat ca să vă ajute în acest sens. De asemenea, fișierul **.iml** conține calea către bibliotecile Jackson. Dacă aveți probleme stergeți folder-ul **.idea** și fișierul **.iml** și generațiile voi din nou din IntelliJ.

## Recomandări

- Tema a fost concepută pentru a vă testa cunoștințele dobândite în urma parcurgerii laboratoarelor 1-4, aceasta putând fi rezolvată doar cu noțiunile învățate din acele laboratoare. Cu toate acestea, vă **sugerăm** să vă folosiți și de noțiunile învățate în laboratorul 5, deoarece acestea vă pot ajuta în crearea/modelarea claselor și ierarhiilor.
- Pentru depanarea diferențelor dintre output-ul vostru și fișierele ref, vă recomandăm [acest site](#).
- Verificați periodic această pagină, deoarece scheletul/cerința pot suferi modificări în urma unor erori din partea noastră.

## Evaluare

Punctajul constă din:

- 80p implementare - trecerea testelor
- 10p coding style (vezi checkstyle)
- 5p README clar, concis, explicații axate pe design (flow, interacțiuni)
- 5p folosire git pentru versionarea temei



Pentru folosirea tool-ului **Git** vă punem la dispoziție un tutorial actualizat și amplu despre el la acest [link](#) și aveți de asemenea și un tutorial despre comenzile pe care puteți să le dați din IntelliJ la acest [link](#).



Pe pagina [Indicații pentru teme](#) găsiți indicații despre scrierea readme-ului și depunctările generale pentru teme

Depunctările pentru **designul și organizarea codului** se vor scădea din punctajul testelor. Dacă vor apărea depunctări specifice temei în momentul evaluării, nementionate pe pagina cu depunctări generale, ele se vor încadra în limitele de maxim 15 pentru design, 5p pentru readme. Dacă tema nu respecta cerințele, sau are zero design OOP atunci se pot face depunctări suplimentare.

Folosirea git pentru versionare va fi verificată din folderul **.git** pe care trebuie să îl includeți în arhiva temei. Punctajul se va acorda dacă ați făcut **minim 3 commit-uri relevante și cu mesaj sugestiv**. **NU** este permis să aveți repository-urile de git publice până la deadline-ul hard.

**Bonusuri:** La evaluare, putem oferi bonusuri pentru design foarte bun, cod bine documentat dar și pentru diverse elemente suplimentare alese de voi.



- Temele vor fi testate împotriva plagiatului. Orice tentativă de copiere va duce la **anularea punctajului** de pe parcursul semestrului și **repetarea materiei** atât pentru sursă(e) cât și pentru destinație(ii), fără excepție.
- **Aveți grijă să nu puneți pe Vmchecker fișiere .idea sau .iml.**

## Checkstyle

Unul din obiectivele temei este învățarea respectării code-style-ului limbajului pe care îl folosiți. Aceste convenții (de exemplu cum numiți fișierele, clasele, variabilele, cum indentați) sunt verificate pentru temă de către tool-ul [checkstyle](#).

Pe pagina de [Recomandări cod](#) găsiți câteva exemple de coding style.

Dacă numărul de erori depistate de checkstyle depășește 30, atunci punctele pentru coding-style nu vor fi acordate. Dacă punctajul este negativ, *acesta se trunchiază la 0*.

Exemple:

- `punctaj_total = 100 și nr_erori = 200 ⇒ nota_finala = 90`
- `punctaj_total = 100 și nr_erori = 29 ⇒ nota_finala = 100`
- `punctaj_total = 80 și nr_erori = 30 ⇒ nota_finala = 80`
- `punctaj_total = 80 și nr_erori = 31 ⇒ nota_finala = 70`

## Upload temă

Arhiva pe care o veți urca pe [VMChecker](#) va trebui să conțină în directorul rădăcină:

- fișierul `README`
- folder-ul `src` cu pachetele și cu fișierele `.java`
- folderul `.git`

## FAQ

**Q:** Pot folosi biblioteca "X"?

**A:** Dacă doriți să folosiți o anumită bibliotecă vă recomandăm să întrebați pe forum, ca apoi să o validăm și să o includem și pe Vmchecker.

**Q:** Am descoperit edge case-ul "Y", trebuie să îl tratez?

**A:** Nu. Toate datele necesare pentru soluționarea temei vă sunt date în cerință. Dacă totuși am omis ceva ne puteți contacta pe forum.

**Q:** Pot folosi clase de tip "Enum" pentru constante?

**A:** Da.

**Q:** Ce JDK recomandați?

**A:** Momentan, orice JDK de la versiunea 11 până în prezent. Dacă vor apărea probleme cu Vmchecker vă vom anunța ca să schimbați JDK-ul cu unul compatibil.

**Q:** Pot să fac în orice ordine testele?

**A:** Da! Testele au fost concepute să fie cât mai decuplate și să testeze câte o funcționalitate în întregime. Cu toate astea, vă recomandăm să implementați mai întâi testele 01, 02, 03 deoarece reprezintă preambulul fiecărui joc. De asemenea, vă recomandăm să implementați și testele 04 și 05 pentru că unele teste verifică dacă ați tratat edge case-ul unei cărți înghețate și pentru asta e nevoie să puteți să înghețați o carte mai întâi.

## Resurse și linkuri utile

- [Schelet de cod](#)
- [Forum](#)
- [Hearthstone short gameplay](#)
- [Gwent short gameplay](#)
- [Indicații pentru teme](#)
- [Recomandări coding style & javadoc](#)

poo-ca-cd/teme/tema.txt - Last modified: 2022/11/18 01:31 by florian\_luis.micu

 Old revisions

 Media Manager  Back to top

