Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Implementace hry Hive

2022

Tomáš Večeřa

Vedoucí práce: doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Tomáš Večeřa
Název práce:	Implementace hry Hive
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2022
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	35
Přílohy:	1 CD
Jazyk práce:	český

Bibliograhic info

Author:	Tomáš Večeřa
Title:	Implementation of the game Hive
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2022
Study field:	Applied Computer Science, combined form
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	35
Supplements:	1 CD
Thesis language:	Czech

Anotace

Aplikace umožňuje hrát hru Hive® po síti. Servervová část je napsána v jazyce Java s použitím frameworku Spring Boot. Klientská část je HTML stránka. Oboustranná komunikace mezi serverem a klientem je zajištěna protokolem WebSocket a architekturou REST. Textová část práce je pojata jako dokumentace ke hře z pohledů uživatele i programátora.

Synopsis

The application allows to play game Hive® over the network. The server side is written in Java using Spring Boot framework. The client side is an HTML page. Two-way communication between the server and the client is provided by the WebSocket protocol and the architecture REST. The text part of the thesis is conceived as a documentation for the game from point of view of both user and programmer.

Klíčová slova: hra; Java; HTML; REST; WebSocket;

Keywords: game; Java; HTML; REST; WebSocket;

Děkuji svému vedoucímu práce doc. RNDr. Miroslavu Kolaříkovi, Ph.D. za vedení mé práce a volnost při upřesnění tématu a výběru technologií.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

1 Úvod **8**
- 1.1 Výběr tématu . . . 8
- 1.2 Výběr jazyků a technologií . . . 8

2 Základní informace o hře **8**

3 Pravidla **9**
- 3.1 Cíl . . . 9
- 3.2 Začátek hry a základní postup . . . 9
- 3.3 Pravidlo nasazování kamenů . . . 9
- 3.4 Pravidlo nasazení královny . . . 9
- 3.5 Nemožnost tahu a remíza . . . 9
- 3.6 Obecná pravidla pro přesun kamenů . . . 9
  - 3.6.1 Pravidlo jednoho úlu . . . 9
  - 3.6.2 Pravidlo volného pole . . . 10
- 3.7 Pravidla kamenů . . . 11
  - 3.7.1 Včelí královna . . . 11
  - 3.7.2 Brouk . . . 12
  - 3.7.3 Mravenec . . . 13
  - 3.7.4 Pavouk . . . 14
  - 3.7.5 Kobylka . . . 15
- 3.8 Časový limit na zahrání tahu . . . 16

4 Uživatelská dokumentace **16**
- 4.1 Spuštění serveru . . . 16
- 4.2 Legenda HTML stránek klienta . . . 17
  - 4.2.1 Menu . . . 17
  - 4.2.2 Hra . . . 18

5 Programátorská dokumentace **19**
- 5.1 Použité technologie . . . 19
  - 5.1.1 Spring Boot . . . 19
  - 5.1.2 Thymeleaf . . . 19
  - 5.1.3 TestNG . . . 19
  - 5.1.4 Gradle . . . 19
- 5.2 Přehled zdrojového kódu serveru . . . 19
- 5.3 Přehled zdrojového kódu klienta . . . 23
- 5.4 Převod interní reprezentace hrací desky na klientskou . . . 23
- 5.5 Automatizované testy . . . 25
- 5.6 Komunikační protokol . . . 26
  - 5.6.1 Klient-server . . . 27
  - 5.6.2 Server-klient . . . 31
  - 5.6.3 Validace uživatele . . . 31

Závěr 32

Conclusions 33

A Obsah přiloženého CD 34

Reference 35

Seznam obrázků

1 Pravidlo jednoho úlu (bílý mravenec se nesmí pohnout) 10
2 Pravidlo volného pole (bílý mravenec má jen dvě možnosti pohybu) 10
3 Včelí královna 11
4 Příklad pohybů včelí královny 11
5 Brouk 12
6 Brouk blokující jiný kámen 12
7 Bílý brouk blokující tři kameny při najetí myší 12
8 Mravenec 13
9 Příklad pohybů mravence 13
10 Pavouk 14
11 Příklad pohybů pavouka 14
12 Kobylka 15
13 Příklad pohybů kobylky 15
14 Legenda menu 17
15 Legenda hry 18
16 Diagram případů užití 20
17 Diagram tříd 21
18 Převod interní reprezentace hrací desky na klientskou 24
19 Simulace situace 1.7 25

Seznam tabulek

1 Případy užití 19
2 Třídy 22
3 Metoda zjištění stavu her 27
4 Metoda založení nové hry 28
5 Metoda pozvánky na připojení do hry 28
6 Metoda opětovného připojení do hry (verze po ztrátě spojení) . . 29
7 Metoda opětovného připojení do hry (verze po zavření hry) 29
8 Metoda zahrání tahu 30

Seznam zdrojových kódů

1 Automatizovaný test pro situaci 1.7 25
2 Hra převedená do formátu JSON 26
3 Odpověd na založení nové hry 28
4 Odpověď na opětovné připojení do hry 29
5 Tělo požadavku zahrání tahu 30
6 Příklad přihlášení klienta k odběru notifikací hry 31

1 Úvod

1.1 Výběr tématu

Chtěl jsem si vybrat téma, se kterým mám alespoň základní zkušenosti. I když jsem je s programováním her neměl, v průběhu výběru tématu bakalářské práce jsem absolvoval předměty Projektový seminář 1 a 2, kde jsem je získal. V oblasti architektury klient-(web)server mám zkušenosti ze zaměstnání, ale jen s použitím jednostranné komunikace (architektura REST a prokol SOAP), takže mi z pohledu teoretických znalostí zbývalo nastudovat oboustrannou komunikaci. To mi přišlo proveditelné, takže jsem s výběrem tohoto tématu dlouho neváhal.

1.2 Výběr jazyků a technologií

Výběr jazyků (Java pro serverovou část a HTML pro klientskou) a technologií jsem měl, stejně jako výběr tématu, ovlivněn předchozími zkušenostmi. Framework Spring (a jeho rozšíření Spring Boot) mě velmi zaujal dobrou dokumentací s příklady, popularitou a tím, že je jeho součástí konfigurovatelný web server Tomcat, což mi přišlo jako zajímává alternativa k nasazování aplikace na server. Pro implementaci klientské části jsem si vybral HTML, protože jediný požadavek na hráče je mít jakýkoliv moderní webový prohlížeč (hra byla odladěna pro prohlížeč Google Chrome).

2 Základní informace o hře

Hra Hive® je strategická hra pro dva hráče. Základní hra se skládá z 22 kamenů – ty se rozdělují podle barvy (11 bílých a 11 černých) a podle druhu hmyzu (každý hráč má k dispozici včelí královnu, 2 brouky, 3 mravence, 3 kobylky a 2 pavouky). Existují oficiální i fanouškovské rozšíření hry přidávající další druhy hmyzu, ty ale v mé implentaci hry použity nejsou. Autorem hry je John Yianni a vydavetelem je Gen42 Games.

Hive® je registrovaná ochranná známka společnosti Gen42 Games, London, UK https://www.gen42.com.

Obrázky kamenů v klientské části hry a v textu práce vycházejí ze zdrojů k návodu na platformě instructables [3].

3 Pravidla

Pravidla vycházejí ze zdrojů [1] a [2].

3.1 Cíl

Cílem hry je obklíčit protihráčovu královnu ze všech (šesti) stran a snažit se ušetřit tohoto osudu svoji královnu. Nezáleží při tom na typu ani barvě kamenů, které královnu obklíčí.

3.2 Začátek hry a základní postup

Na začátku hry je hrací plocha (také nazývána hive nebo úl) prázdná a mladší nebo náhodně vybraný hráč, začíná nasazením prvního kamene. Poté se hráči střídají v tazích a pokud mu to pravidla umožnují musí nasadit další kámen nebo pohnout některým ze svých nasazených kamenů. Nasazení i pohyb kamenů se řídí několika pravidly, která jsou popsána níže.

3.3 Pravidlo nasazování kamenů

Nově nasazený kámen musí sousedit s některým kamenem stejné barvy a nesmí sousedit s kamenem protivníka. Toto pravidlo neplatí pro první dva nasazené kameny (jeden černý a jeden bílý).

3.4 Pravidlo nasazení královny

Královna musí být nasazena nejpozději ve čtvrtém kole daného hráče. Dokud není královna nasazena, hráč nemůže přesouvat kameny. Z toho plyne, že nejpozději lze nasadit královnu, když jsou na hrací ploše tři kameny daného hráče.

3.5 Nemožnost tahu a remíza

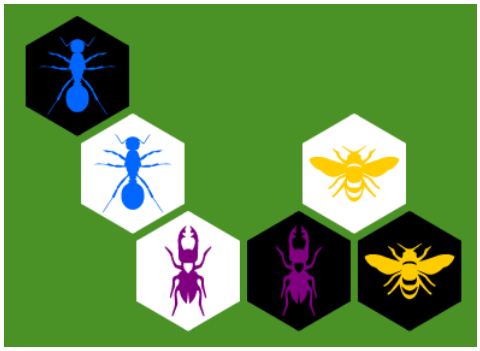
Pravidla umožňují situaci, kdy hráč nemůže pohnout žádným kamenem a nemůže nasadit nový kámen. V tom případě získává protivník další tah. Pokud jsou v této situaci oba hráči, nastává remíza. Ta nastane i v případě, že jedním tahem hráč plně obklíčí obě královny.

3.6 Obecná pravidla pro přesun kamenů

Pokud není uvedeno jinak, tak platí pro všechny kameny.

3.6.1 Pravidlo jednoho úlu

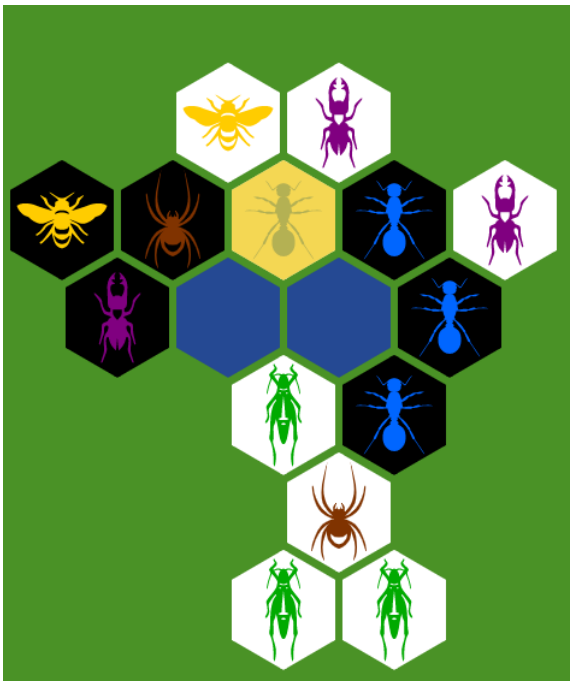
Po celou dobu hry (i během tahu) nesmí kámen ani skupina kamenů stát osamoceně (takzvané rozpojení úlu). Na obrázku 1 je situace, ve které se bílý mravenec nesmí pohnout (černý mravenec by byl osamocený).



Obrázek 1: Pravidlo jednoho úlu (bílý mravenec se nesmí pohnout)

3.6.2 Pravidlo volného pole

Po celý tah musí kámen zůstat v kontaktu s herní plochou. Z toho plyne, že s kamenem, který je obklíčen ze všech šesti stran nelze pohnout. Tah nelze provést pokud by to vyžadovalo posunutí (i když jen dočasné) jiných kamenů. Z toho plyne, že nelze pohnout ani kamenem obklíčeným z pěti stran. Také z toho plyne, že nelze provést tah, pokud na jeho cestě existují dvě sousední pole taková, že mají dva jiné společné sousedy. Toto pravidlo neplatí pro brouka a kobylku. Na obrázku 2 je situace, ve které má bílý mravenec pouze dvě možnosti pohybu.



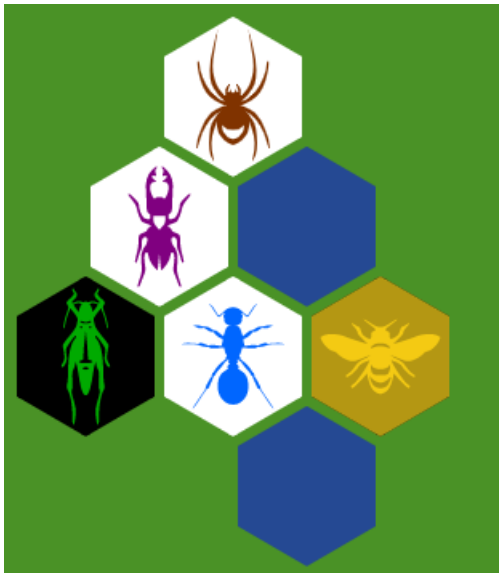
Obrázek 2: Pravidlo volného pole (bílý mravenec má jen dvě možnosti pohybu)

3.7 Pravidla kamenů

3.7.1 Včelí královna

Pohybuje se vždy o právě jedno pole.

Obrázek 3: Včelí královna



Obrázek 4: Příklad pohybů včelí královny

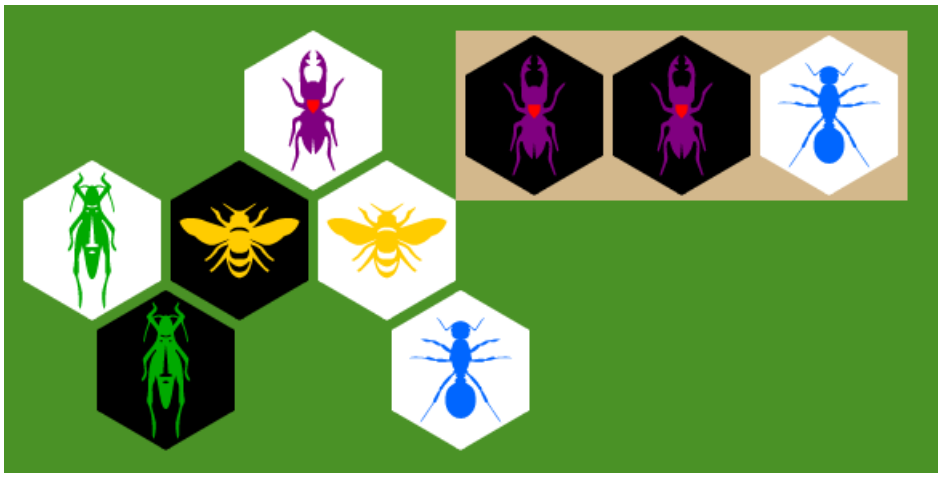
3.7.2 Brouk

Jako královna se pohybuje jen o jedno pole, ale jeho specialita je možnost vylézt na jiný kámen. Takto zablokovaným kamenem nelze pohnout, dokud se brouk nepřesune. Může vylézt i na pole už obsané jiným broukem, který má pod sebou jiný kámen, takže můžou být i všichni čtyři brouci na jednom poli. Nasazovat se ale musí na prázdné pole. Barva pole je určena barvou nejvýše umístěného brouka.

V mé implementaci klienta má brouk, který obsadí jiný kámen červenou značku a při najetí myší na tento kámen se zobrazí kameny, které jsou pod ním (vlevo je nejhořejší).

Obrázek 5: Brouk

Obrázek 6: Brouk blokující jiný kámen

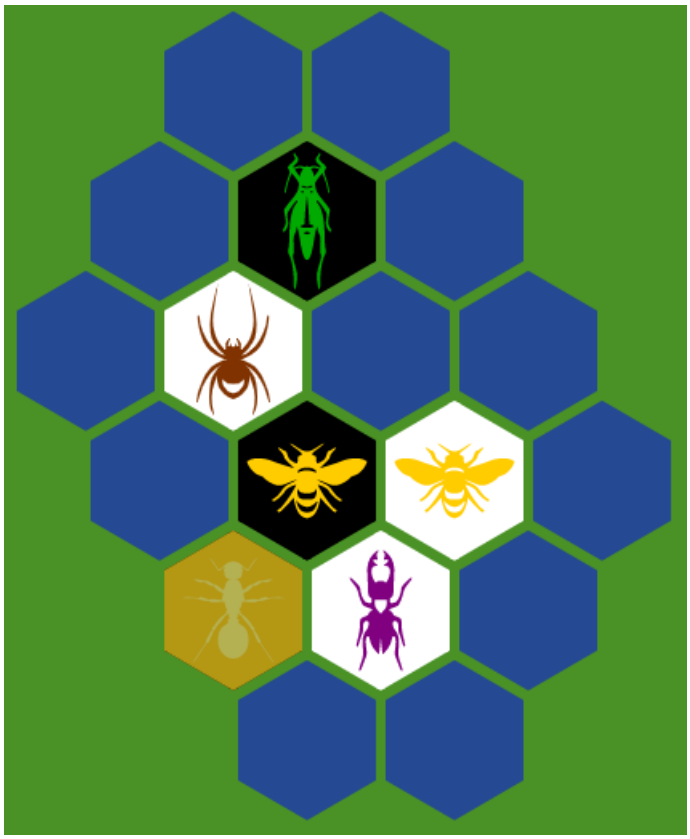


Obrázek 7: Bílý brouk blokující tři kameny při najetí myší

3.7.3 Mravenec

Může se pohybovat o libovolný počet polí.

Obrázek 8: Mravenec



Obrázek 9: Příklad pohybů mravence

3.7.4 Pavouk

Pohybuje se vždy o 3 unikátní pole.

Obrázek 10: Pavouk

Obrázek 11: Příklad pohybů pavouka

3.7.5 Kobylka

Skáče přes rovnou řadu společně propojených kamenů na další volné pole. Vždy musí skočit přes alespoň jeden kámen.

Obrázek 12: Kobylka

Obrázek 13: Příklad pohybů kobylky

3.8 Časový limit na zahrání tahu

V oficiálních pravidlech není definován časový limit na zahrání tahu, ale pro deskovou hru po síti je, dle mého názoru, potřeba. Proto jsem ho ve své implemntaci hry definoval, a to 5 minut. Po vypršení kolo propadne a na řadě je protihráč. Pokud hráč zmešká ještě jedno kolo, vyhrává protihráč. Výjimkou jsou první dva tahy (takzvané inicializační). Zde je limit pouze 1 minuta a pokud hráč tah zmešká, vyhrává protihráč. To zamezí zbytečnému čekání, pokud hráč čeká na připojení protihráče a nevšimne si, že už se tak stalo (protihráč by pro korektní ukončení hry musel čekat 10 minut).

4 Uživatelská dokumentace

4.1 Spuštění serveru

Předpoklady:

- Všechny příkazy z této podkapitoly jsou určené pro příkazovový řádek.
- Aplikace je otestována pro Javu 8, ale měla by fungovat i pro vyšší verze.

Pro spuštění serveru jsem vytvořil spustitelný `hive.jar` soubor, který je obsahem přiloženého CD. Je ho možné spustit příkazem:
`java -jar hive.jar`

Klient aplikace (určený pro otevření ve webovém prohlížeči) je poté dostupný na adrese `localhost` (konkrétně na portu 80, ten lze ale v adrese vynechat).

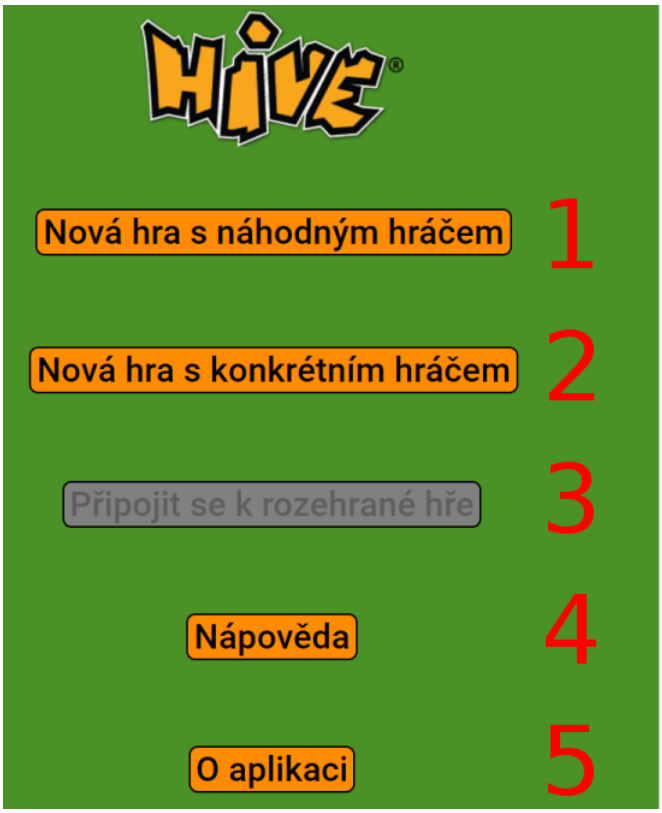
Při spuštění je možné konfigurovat server:

- jiný než výchozí port (80)
  `java -Dserver.port=1234 -jar hive.jar`
- vypisování veškeré komunikace mezi serverem a klienty
  `java -Dlogging.level.=INFO -jar hive.jar`

Nejedná se o jediný způsob jak server spustit, ale ostatní jsou vhodné spíše pro spouštění v průběhu vývoje aplikace a většina vyžaduje vývojové prostředí Eclipse.

4.2 Legenda HTML stránek klienta

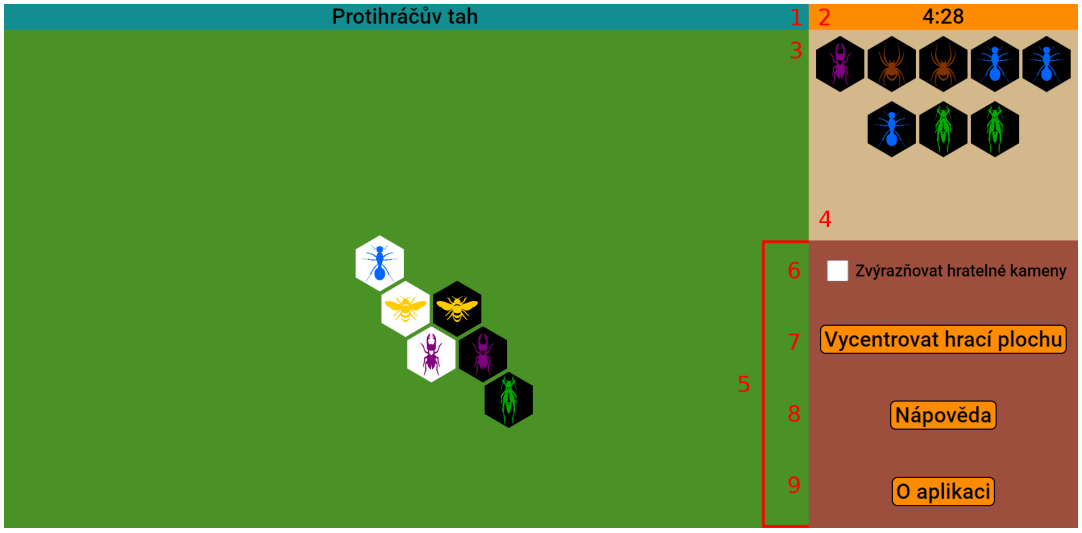
4.2.1 Menu



Obrázek 14: Legenda menu

1. Pokud jiný hráč čeká na hru s náhodným hráčem, tak se hráči propojí a hra se spustí. Pokud ne, hráči se zobrazí hrací deska, ale musí počkat na protihráče.
2. Spustí se nová hra a hráči se zobrazí tlačítko na zkopírování pozvánky do hry pro protihráče. Až protihráč odkaz s pozvánkou otevře, hra se spustí.
3. Pokud hráč během hry zavře okno, ve kterém byla hra spuštěna, ale nezavře prohlížeč a hra mezitím neskončila, má možnost se do hry vrátit. Po kliknutí se zobrazí okno s jednoduchou tabulkou rozehraných her (čas spuštění hry, barvou kamenů a tlačítko na připojení do hry).
4. Zobrazí okno se dvěma záložkami „Legenda“ a „Pravidla hry“ obsahující stejné informace jako kapitoly 3 a 4.2.
5. Zobrazí okno se jménem autora, odkazem na katedru informatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci a informace ohledně registrované značky `Hive`.

4.2.2 Hra



Obrázek 15: Legenda hry

1. Informační lišta
2. Zbývající čas kola. Po vypršení kolo propadne a je na řadě protihráč a u dalších tahů hráče je vedle časovače varování, protože při dvou propadnutích vyhrává protihráč (časový limit kola je popsán v podkapitole 3.8).
3. Hrací plocha s nasazenými kameny
4. Dosud nenasazené kameny
5. Postranní menu
6. Hratelné kameny jde poznat podle změny kurzoru při najetí na ně, ale pokud je toto políčko zaškrtnuto, hratelné kameny jsou zvýrazněny růžovou barvou. Hra je poté přístupnější pro začátečníky, kteří nemají vžitá všechna pravidla.
7. Vycentruje hrací plochu tak, aby byl v jejím středu prostřední kámen (vertikálně i horizontálně).
8. Stejné tlačítko, které je použito v hlavním menu (viz položka 4 v legendě menu).
9. Stejné tlačítko, které je použito v hlavním menu (viz položka 5 v legendě menu).

5 Programátorská dokumentace

5.1 Použité technologie

5.1.1 Spring Boot

Nejdůležitějším prvkem hry z pohledu použitých technologií je Spring. Jedná se o framework umožňující jednoduchý vývoj webových aplikací v Javě. Také jsem využil jeho rozšíření Spring Boot, které jak už bylo zmíněno v úvodu, obsahuje konnfigurovatelný Tomcat web server.

5.1.2 Thymeleaf

Při seznámování s frameworkem Spring Boot jsem zjistil, že se distribuuje s knihovnou Thymeleaf. Ta umožňuje obohacovat HTML stránky o Javascriptový objekt generovaný v Javě (překládá se do JSON formátu). Tím jsem vyřešil problém jak mít pro hru s náhodným hráčem a privátní hru stejnou HTML stránku (požadavky jsou rozdílné, ale oba dostanou odpověď v podobě přesměrování na stejnou HTML stránku, jen obohacenou o jiný Javascriptový objekt).

5.1.3 TestNG

Pro psaní automatizovaných testů jsem použil framework TestNG. Vybral jsem si ho, protože je velmi dobře podporován ve vývojovým prostředím Eclipse.

5.1.4 Gradle

Abych nemusel mít v git repositáři .jar soubory s těmito technologiemi, použil jsem nástroj Gradle. Ten umožňuje jednoduchým skriptem tyto soubory stáhnout z veřejného repositáře https://search.maven.org/. Další funkcionalita tohoto nástroje, kterou jsem využil, je jednoduché vytváření .jar souboru aplikace.

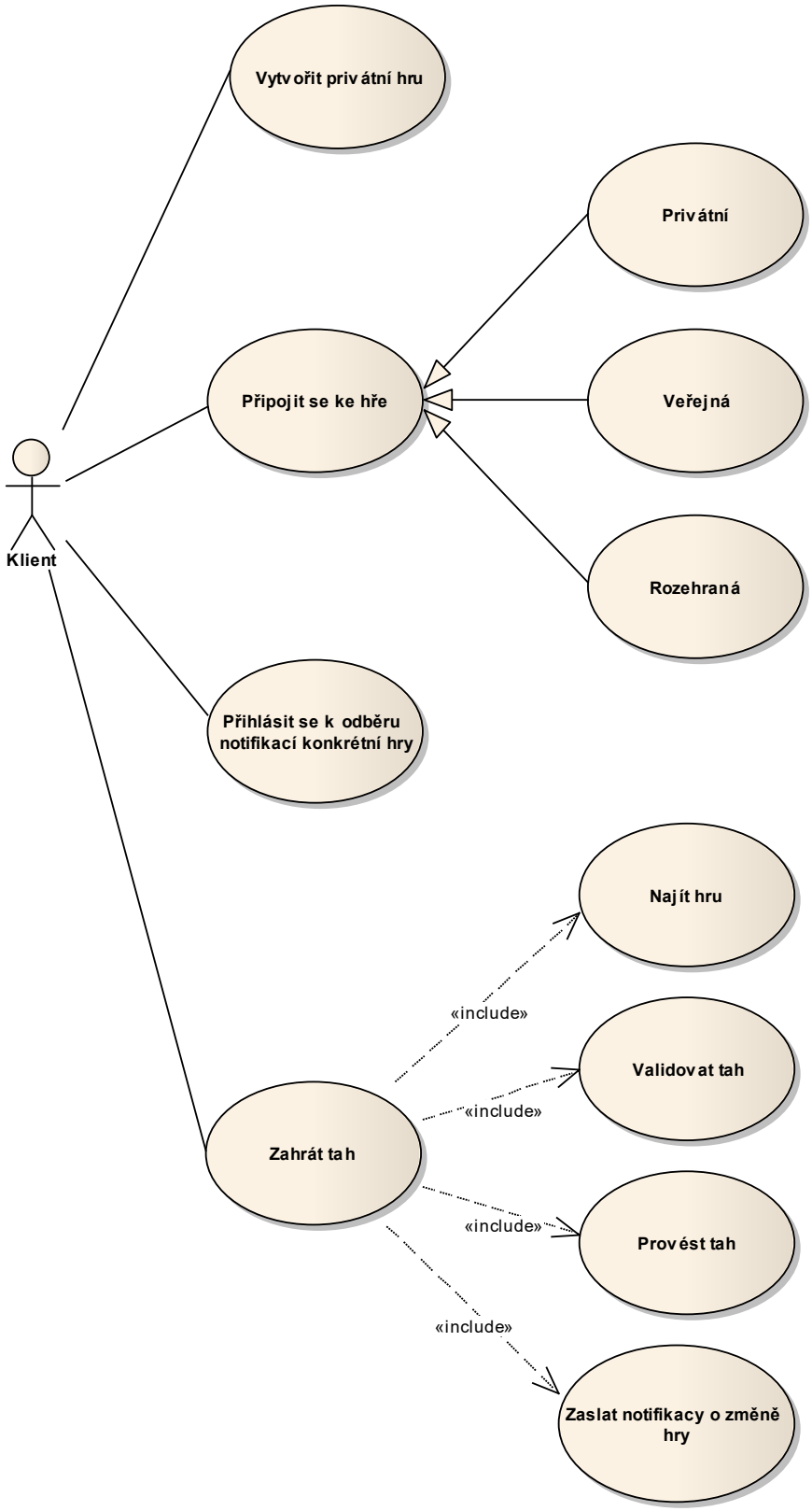
5.2 Přehled zdrojového kódu serveru

Tato podkapitola je pojata z pohledu návrhu a proto jsou některé nepodstatné případy užití, třídy a další detaily vynechány.

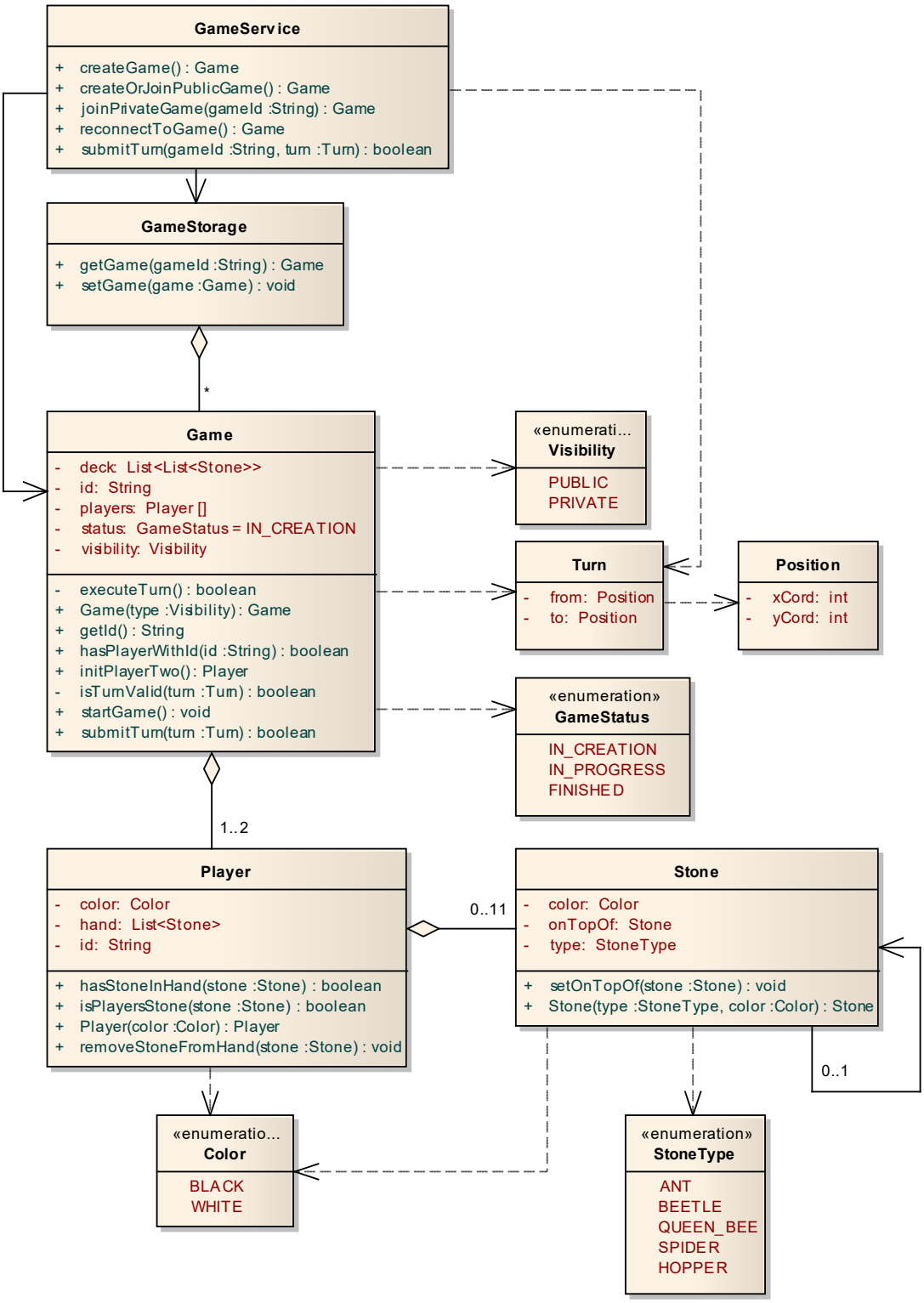
Tabulka 1: Případy užití

Případ užití	Zajišťuje	
	Třída	Operace
Vytvořit privátní hru	GameService	createGame
Připojit se ke hře	GameService	joinPrivateGame
		createOrJoinPublicGame
		reconnectToGame
Přihlásit se k odběru notifikací hry	WebSocketMessageBrokerConfigurer*	registerStompEndpoints
Zahrát tah	Game	submitTurn
Najít hru	GameStorage	getGame
Validatovat tah	Game	isTurnValid
Provést tah	Game	executeTurn
Zaslat notifikaci o změně hry	SimpMessagingTemplate*	convertAndSend

* třídy frameworku Spring



Obrázek 16: Diagram případů užití



Obrázek 17: Diagram tříd

Tabulka 2: Třídy

Operace	Popis	Spolupracuje	
		Třída	Operace
	GameService		
createGame	Vytvoří novou nespuštěnou hru, operací setGame ji přidá do GameStorage a hru vrátí.	GameStorage Game	setGame konstruktor+getId
joinPrivateGame	Operací getGame najde v GameStorage hru (Hra musí být typu PRIVATE), operací initPlayerTwo do ní přidá druhého hráče, operací startGame ji spustí a hru vrátí.	GameStorage Game	getGame initPlayerTwo+startGame
createOrJoinPublicGame	Pokud jiný hráč založil veřejnou hru a čeká na připojení druhého tak se chová stejně jako connectToPrivateGame. Pokud ne, tak vytvoří novou hru typu PUBLIC, operací setGame se přidá do GameStorage a hru vrátí.	GameStorage Game	getGame+setGame konstruktor+initPlayerTwo+startGame
submitTurn	V GameStorage pomocí operace getGame najde konkrétní hru a operací submitTurn na objektu Game se pokusí zahrát tah. V případě, že tah proběhl (byl validní) tak do GameStorage vloží operací setGame aktualizovanou hru. Vrací boolean na základě úspěšnosti.	GameStorage Game	getGame+setGame submitTurn
reconnectToGame	V GameStorage pomocí operace getGame najde konkrétní hru a pokud hra obsahuje hráče s poskytnutém id, hru vrátí.	GameStorage Game	getGame hasPlayerWithId
	GameStorage		
getGame	Vrátí hru na zakládě poskytnutého ID		
setGame	Aktualizuje (popřípadě přidá novou) hru do svého interního uložiště na základě id (getId oeprace).	Game	getId
	Game		
initPlayerTwo	Přidá do hry druhého hráče (včetně jeho kamenů) a vrátí ho.	Player Stone	konstruktor konstruktor
submitTurn	Zkontroluje jestli je tah podle pravidel (isTurnValid) a pokud ano tak ho provede (executeTurn). Vrací boolean.		
isTurnValid	Zkontroluje jestli je tah podle pravidel – kámen musí být hráče, který je na tahu (operace isPlayersStone) a hrací deska tento tah musí umožňovat (viz pravidla). Vrací boolean.	Player Stone Turn	isPlayersStone Gettery atributů Gettery atributů
executeTurn	Provede tah včetně všech vedlejších efektů: 1.kontrola výhry 2. přepnutí stavu hry 3. přepnutí hráče 4. Pokud byl kámen v ruce hráče tak se z ruky odstraní (operace removeStoneFromHand). 5. Pokud byl na nové pozici jiný kámen tak se operací setOnTopOf přesune pod kámen, kterým se táhlo.	Player Stone	removeStoneFromHand setOnTopOf
startGame	Přepne hru do stavu IN_PROGRESS.		
	Player		
hasStoneInHand	Zkontroluje, jestli má hráč daný kámen v ruce (na základě operací color a type). Vrací boolean.	Stone	getColor+getType
isPlayersStone	Zkontroluje, jestli je kámen daného hráče. (na základě operace getColor). Vrací boolean.	Stone	getColor
removeStoneFromHand	Odebere kámen z hráčovy ruky a kámen vrátí.		

5.3 Přehled zdrojového kódu klienta

resources/statis/js/communication.js Komunikace se serverem a obsluha vypršení časového limitu tahu.

resources/statis/js/game.js Inicializace hry, příprava pozvánky pro protihráče, ukládání rozehraných her do localStorage a obsluha tlačítka na zvýraznění hratelných tahů.

resources/statis/js/menu.js Obsluha připojení do rozehrané hry z menu.

resources/statis/js/popups.js Obsluha oken pro nápovědu a informace o aplikaci.

resources/statis/js/scroll.js Funkce pro podporu navigace na hrací desce.

resources/statis/js/status.js Obsluha informační lišty.

resources/statis/js/stones_manager.js Všechny funkcionality kamenů na hrací ploše i ještě nenasazených.

resources/statis/index.html HTML stránka pro úvodní stránku (menu).

resources/templates/game-template.html HTML stránka hry, kterou vrací server obohacenou o JavaScript objekt hry.

resources/templates/error.html HTML stránka pro zobrazování některých chyb (například při otevření neplatné pozvánky do hry).

resources/statis/css/ Složka se soubory kaskádových stylů.

resources/statis/svg/ Složka s svg obrázky.

resources/statis/img/ Složka s jinými než svg obrázky.

5.4 Převod interní reprezentace hrací desky na klientskou

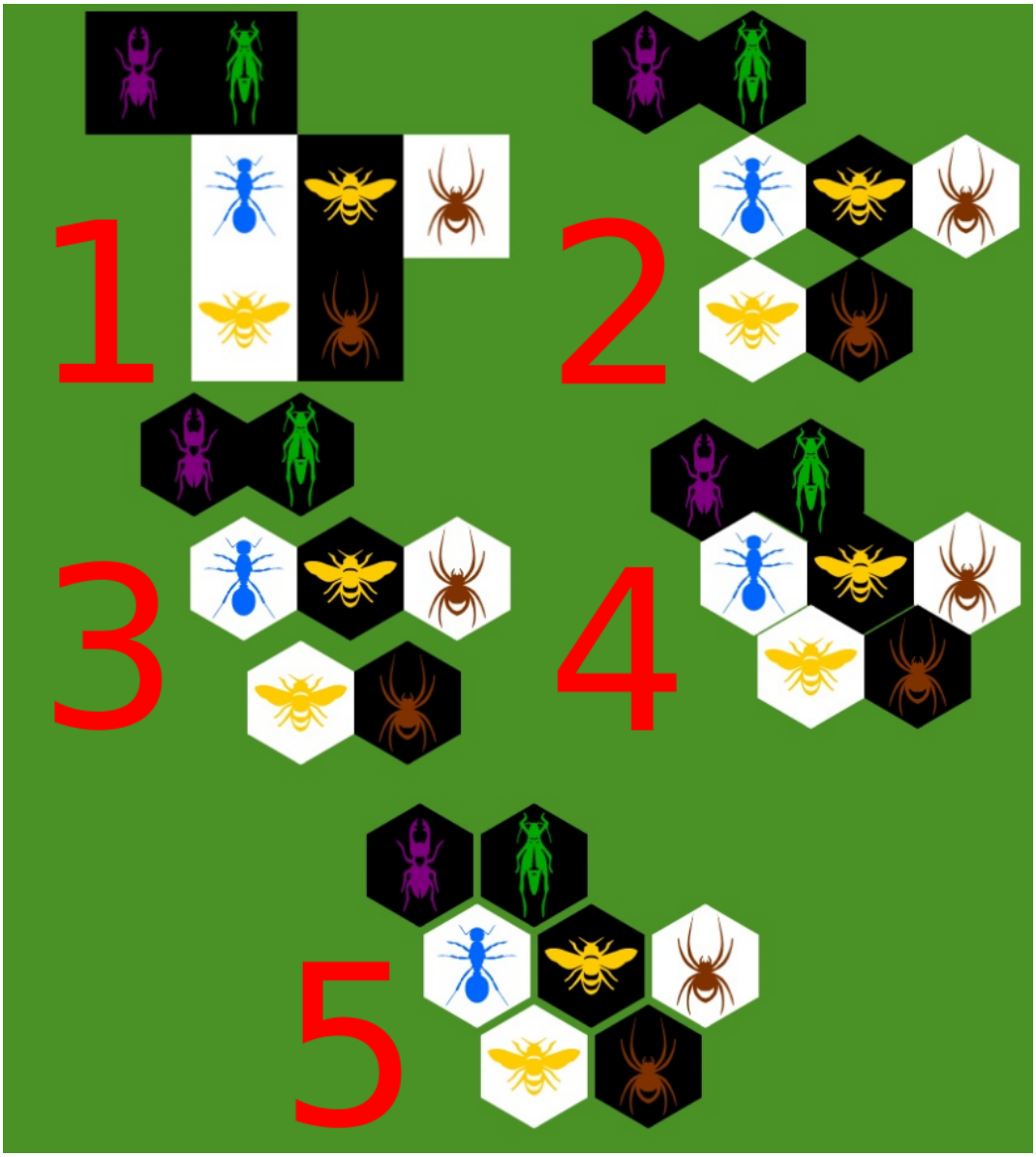
Vzhledem ke tvaru kamenů a jejich umístění na hrací ploše není hra Hive® vhodná pro převod do kódu (a do formátu JSON pro komunikaci s klienty) bez transformace na nějakou více standardní a strojově čitelnější reprezentaci. Já jsem zvolil dvourozměrné pole a jeho transformace na klientskou reprezentaci (prováděna klientem) probíhá takto:

1. Vstupní dvojrozměrné pole
2. Převod obdélníku na šestiúhelník. V mé implentaci klienta pomocí css vlastnosti `clip-path`.

3. Hra obsahuje boolean atribut `isFirstRowIndented` určující jestli jsou horizontálně odsazeny liché nebo sudé řádky (po přidání kamene do prvního řádku se jeho hodnota překlápí). Odsazení je doprava o polovinu šířky kamene.

4. Posunutí řádku nahoru o `výška kamene * 0,25 * číslo řádku`. Zde je vidět nepřesnost `clip-path` (závisí na rozlišení obrazovky a prohlížeči), ale pro moje účely je přípustná.

5. Kamenům se přidá rozestup.

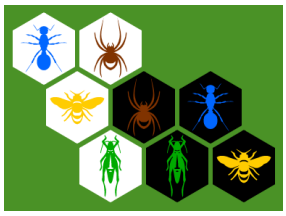


Obrázek 18: Převod interní reprezentace hrací desky na klientskou

5.5 Automatizované testy

Pro snažší implementaci hry jsou automatizované testy velmi užitečnou pomůckou. Usnadňují nejen počáteční implementaci, ale i případné opravování chyb v pozdějších fázích vývoje, kdy se programátor nemusí bát, že opravou něco rozbije.

V knize, ze které jsem čerpal pravidla [1] je ke každému pravidlu nejen slovní popis, ale i několik očíslovaných obrázků se situacemi ukazující pravidlo v praxi, takže jsem část automatizovaných testů napsal pro právě tato pravidla. Z důvodu autorských práv není následující obrázek z knihy, ale jedná se o stejnou situaci simulovanou v mé implementaci klienta (v knize má situace číslo 1.7), na které bylo vysvětleno pravidlo volného pole (černý pavouk se nesmí pohnout).



Obrázek 19: Simulace situace 1.7

```
1  @Test
2  public void test01_07() {
3    AutoGame game = new AutoGame(TestUtils.twoDimensionalArrayIntoDeck
         (new Stone[][] {
4      // @formatter:off
5        {null, null, null, null, null, null},
6      {null, new Stone(Color.WHITE, Type.ANT), new Stone(Color.WHITE,
         Type.SPIDER), null, null, null},
7      {null, new Stone(Color.WHITE, Type.QUEEN_BEE), new Stone(Color.
         BLACK, Type.SPIDER), new Stone(Color.BLACK, Type.ANT), null,
         null},
8      {null, null, new Stone(Color.WHITE, Type.HOPPER), new Stone(Color
         .BLACK, Type.HOPPER), new Stone(Color.BLACK, Type.QUEEN_BEE),
          null},
9      {null, null, null, null, null, null},
10     // @formatter:on
11   }), true, true, true);
12   assertEquals(game.getDeck().get(2).get(2).getType(), Type.SPIDER);
13   assertEquals(
14     game.getValidTurns().stream()
15         .filter(turn -> turn.getFrom() != null && turn.getFrom().
              equals(new Position(2, 2))).count(),
16     0);
17 }
```

Zdrojový kód 1: Automatizovaný test pro situaci 1.7

5.6 Komunikační protokol

Těla požadavků a odpovědí v této podkapitole jsou pouze ukázkou jak probíhá komunikace. Pokud by měla sloužit jako dokumentace pro napsaní nového klienta, musela by být, podle mého názoru, výrazně detailnější (více vysvětlivek a ukázek). To ale není předmětem této práce a pravděpodobně by předčila v množství textu zbytek práce.

```
1  {
2    "objectType": "GameReconnect",
3    "gameId": "9bbb49da-6bc5-4b6d-8a03-193a90f5a192",
4    "currentPlayerColor": "WHITE",
5  // pro přehlednost je uveden pouze jeden prvek z pole nenasazených
       kamenů hráče
6    "currentPlayerHand": [{ "color": "WHITE", "type": "BEETLE", "
       onTopOf": null}],
7    "currentPlayerTimeouts": 0,
8    "queenMandatory": false,
9  // pro přehlednost je uveden pouze jeden prvek z pole nenasazených
       kamenů protivníka
10   "enemyPlayerHand": [{ "color": "BLACK", "type": "BEETLE", "onTopOf
       ": null }],
11   "deck": [[null, null, null, null, null],
12     [null,null,{ "color": "BLACK", "type": "ANT", "onTopOf": null},
         null,null],
13     [null,null,{ "color": "BLACK" , "type": "QUEEN_BEE", "onTopOf":
         null},{ "color": "WHITE", "type": "QUEEN_BEE", "onTopOf":
         null}, null],
14     [ null, null, null, { "color": "WHITE", "type": "BEETLE", "
         onTopOf": null }, null],
15     [null, null, null, null, null]],
16   "gameStatus": "IN_PROGRESS",
17 // pro přehlednost je uveden pouze jeden prvek z pole validních tahů
18   "validTurns": [{
19      "from": null,
20      "to": {"xCord": 4, "yCord": 3 },
21      "stone": { "color": "WHITE", "type": "SPIDER", "onTopOf": null
           },
22      "timeout": false
23     }]
24   "firstRowIndented": false,"visibility": "PUBLIC", "timeout":
       1645950470073, "handshakeTurn": false, "allowedTimeouts": 1
25 }
```

Zdrojový kód 2: Hra převedená do formátu JSON

5.6.1 Klient-server

Komunikace z klienta na server probíhá pomocí architektury REST. Aplikace obsahuje dva typy požadavků:

1. Požadavky přijímající a vracející data ve formátu JSON. Jedná se o univerzální požadavky nezávislé na druhu klienta (například webová stránka, desktopová aplikace nebo aplikace pro telefony)

2. Thymeleaf požadavky. Jak už jsem zmínil v kapitole 5.1.2, využil jsem knihovnu Thymeleaf k vytvoření speciálních požadavků vracejících přesměrování na HTML stránku (hru) obohacenou o JavaScriptový objekt. Pro zachování možnosti vytvoření klienta nevyužívajícího HTML stánky, existuje pro každý Thymeleaf požadavek jeho univerzální varianta vracející JavaScriptový objekt, o který by byla obohacena HTML stránka. Tyto požadavky také přijímají data ve formátu JSON.

Tabulka 3: Metoda zjištění stavu her

Adresa	`/menu/games?saved={řetězec_her}`
Typ	GET
Thymeleaf verze	Ne
Popis	Klient by měl ukládat rozehrané hry (v mé implementaci klienta jsou uloženy v localStorage), aby v případě, že hráč hru zavře, měl možnost se do hry vrátit. Nemá ale smysl vracet se do už ukončené hry (hráč se nestihl vrátit včas). Proto existuje tato metoda, které se předá řetězec uložených her a vrátí pole her, které jsou rozehrané. Jednotlivé hry musí být ve formátu `id_hry:id_hráče` a hry musí být oddělené středníkem.
Příklad	`/menu/games/saved?=1234:abcd;5678:efgh;`

Tabulka 4: Metoda založení nové hry

Adresa	`/game/create`
Typ	POST
Thymeleaf verze	`/thymeleaf/game/create`
Tělo požadavku	`startMode` **PUBLIC** Nová hra s náhodným hráčem. **PRIVATE** Nová hra s konkrétním hráčem.
Tělo odpovědi	Objekt hry a data o připojeném hráči (id, barva kamenů a pole dostupných kamenů k nasazení).

```
{
  "lastPlayerJoinedId": "ff93131e-0820-46e8-bf1e-f4a3b857b6ec",
  "lastPlayerJoinedColor": "WHITE",
// pro přehlednost je uveden jen první prvek pole
  "lastPlayerJoinedHand": [
    { "color": "WHITE", "type": "SPIDER", "onTopOf": null }
  ],
// pro přehlednost je vynechán objekt hry
}
```

Zdrojový kód 3: Odpověd na založení nové hry

Tabulka 5: Metoda pozvánky na připojení do hry

Adresa	`/game/join/{id_hry}`
Typ	GET (i když modifikuje hru je potřeba typ GET, aby šla pozvánka jednoduše otevřít v prohlížeči)
Thymeleaf verze	`/thymeleaf/game/join/{id_hry}`
Tělo odpovědi	Stejné jako v předchozí metodě (viz tabulka 4)

Tabulka 6: Metoda opětovného připojení do hry (verze po ztrátě spojení)

Adresa	`/game/reconnect/{id_hry}/{id_hráče}`
Typ	GET
Thymeleaf verze	Ne
Popis	Účelem této metody je získát aktuální hru při opětovném navázání spojení po jeho ztrátě.
Tělo odpovědi	Hra převedená do formátu JSON (viz zdrojový kód 5.6).

Tabulka 7: Metoda opětovného připojení do hry (verze po zavření hry)

Adresa	`/menu/reconnect/{id_hry}/{id_hráče}`
Typ	GET
Thymeleaf verze	`/thymeleaf/menu/reconnect/{id_hry}/{id_hráče}`
Popis	Účel této metody je umožnit hráči opětovné připojení do hry po tom, co ji zavřel.
Tělo odpovědi	Objekt hry a data o hráči, který se chce připojit, a to jeho barvu, pole nenasazených kamenů a id. ID sice klient musí znát, aby se mohl připojit, ale v případě použití Thymeleaf verze metody je ho potřeba předat stránce hry, na kterou klient dostane přesměrování.

```
{
  "reconnectedPlayerId": "ff93131e-0820-46e8-bf1e-f4a3b857b6ec",
  "reconnectedPlayerColor": "WHITE",
  "reconnectedPlayerHand": [
    { "color": "WHITE", "type": "HOPPER", "onTopOf": null }
  ],
  // pro přehlednost je vynechán objekt hry
}
```

Zdrojový kód 4: Odpověď na opětovné připojení do hry

Tabulka 8: Metoda zahrání tahu

Adresa	`/game/submitturn`
Typ	POST
Thymeleaf verze	Ne
Tělo požadavku	Id hry, id uživatele a tah. Tah se skládá ze dvou souřadnic (`from` a `to`) a kamene. Ten se skládá z barvy a typu.
Tělo odpovědi	U této metody záleží pouze na stavovém kódu.
	200 Tah byl zahrán. Vedlejším efektem je WebSocket notifikace s aktualizovanou hrou pro oba hráče popsán v podkapitole 5.6.2).
	400 Tah není podle pravidel.

```
{
  "gameId": "41351243-788a-45a7-9913-1ce1a4a3b637",
  "playerId": "a680e4ab-5593-4f13-ba2f-84272bdac4b5",
  "turn": {
    "from": null,
    "to": {
      "xCord": 3,
      "yCord": 3
    },
    "stone": {
      "type": "HOPPER",
      "color": "BLACK"
    }
  }
}
```

Zdrojový kód 5: Tělo požadavku zahrání tahu

5.6.2 Server-klient

Komunikace ze serveru na klienta probíhá pomocí protokolu WebSocket, respektive nadstavbou STOMP.

WebSocket Protokol pro oboustranou komunikaci mezi klientem a serverem. V mé aplikaci je využit pouze směr ze serveru na klienta.

STOMP Protokol definující formát zpráv oboustranné komunikace. Nemusí se jednat o WebSocket.

STOMP pomocí WebSocket komunikaci jsem implementoval podle oficiálního Spring Boot tutoriálu [4].

Od klienta se očekává, že se po získání id hry přihlásí ke zdroji `/gameplay` a zprávám `/topic/game-progress/{id_hry}`, kde bude server posílat notifikace, po zahrání korektního tahu. Obsahem notifikace je hra převedená do formátu JSON (viz zdrojový kód 5.6). Pro toto připojení využívám ve své implementaci klienta knihovny sockjs a stompjs. Jsou také ve zmíněném tutoriálu, protože jsou plně kompatibilní se Spring Boot implementací protokolů STOMP a WebSocket a tudíž implementace připojení na straně klienta byla jednoduchá a takřka bezproblémová.

```
1 var gameId = "1234-abcd-456-efgh";
2 var socket = new SockJS(window.location.origin + "/gameplay");
3 var stompClient = Stomp.over(socket);
4 stompClient.connect({}, function () {
5   stompClient.subscribe("/topic/game-progress/" + gameId,
6     function (response) {
7       console.log("game progress msg: " + JSON.parse(response.body));
8     });
9 });
```

Zdrojový kód 6: Příklad přihlášení klienta k odběru notifikací hry

5.6.3 Validace uživatele

V odpovědích na požadavky připojení do hry získá klient id hráče. To se musí posílat při zahrání tahu a je validováno na serveru. V ostatních odpovědích id hráče není. Takže aby mohl někdo hráči škodit (zahrát jeho tah) musel by například odposlouchávat síťovou komunikaci mezi hráčem a serverem. Proto je dle mého názoru hra proti takovému škození dobře zabezpečena.

Závěr

Úspěšně jsem vytvořil aplikaci pro hraní hry Hive® po síti. Aplikace plně splňuje zadání a neobsahuje nedodělky nebo zásádní koncepční problémy. Komunikační protokol je dostatečně transparentní pro případné rozšíření hry o další typy klientů. Počáteční výběr technologií se ukázal jako vhodný. Na druhou stranu aplikace není přehlídkou nejmodernějších technologií a postupů. Prostor pro zlepšení vidím hlavně ve vzhledu HTML stránek klienta a jejich responzivnosti. Aplikace by také mohla být rozšířena o několik funkcionalit. Například další typy kamenů, nastavitelná délka tahu privátních her, historii tahů, počítačového hráče, režim diváka, systém uživatelských účtů s historií her.

Conclusions

I have successfully created an application to play the game Hive® over the network. The application fully meets the requirements and does not contain unfinished parts or major conceptual problems. The communication protocol is transparent enough to allow expansion of the game with other types of clients. The initial selection of technologies proved to be appropriate. On the other hand the application is not a showcase of the latest technologies and procedures. I see room for improvement mainly in the design of client's HTML pages and their responsivnes. The application could also be extended by several functionalities. For example other types of stones, adjustable turn length of private games, turn history, computer player, viewer mode, user account system with game history.

A Obsah přiloženého CD

`thesis.zip`
: Archiv všech souborů potřebných pro vygenerování PDF dokumentu práce.

`thesis.pdf`
: Text práce ve formátu PDF.

`hive-project/`
: Kompletní zdrojové texty aplikace ve formátu projektu pro vývojové prostředí Eclipse.

`hive.jar`
: Spustitelný .jar soubor aplikace (viz podkapitola 4.1).

`readme.txt`
: Instrukce pro spuštění aplikace (viz podkapitola 4.1).

Reference

[1] INGERSOLL, Randy. *Play Hive Like a Champion: Strategy, Tactics and Commentary.* 2nd ed. 2012. 253 s. ISBN 148006095X.

[2] CAULY. *Hive pravidla základ.*
Dostupné z: https://www.zatrolene-hry.cz/spolecenska-hra/hive-191/k-stazeni/

[3] DRAENOGG. *Hive Game With Box.*
Dostupné z: https://www.instructables.com/Hive-game-with-box/

[4] *Using WebSocket to build an interactive web application.*
Dostupné z: https://www.spring.io/guides/gs/ messaging-stomp-websocket/