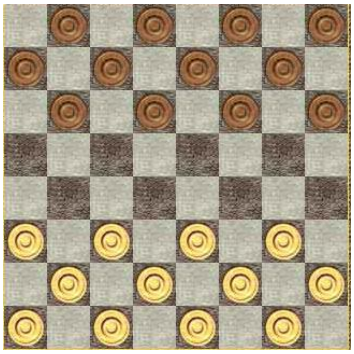
Домаћи задатак из предмета Експертски системи за јунски рок школске 2013/14. године

Основне информације

Дама (енг. *checkers,* фра. *jeu de dames*) је врста комибинаторне игре за два играча на табли од 64 или 100 црно-белих поља. Према неким изворима, ова игра је стара око 4.000 година. У свом развоју, она се играла на таблама с различитим бројем поља и различитим правилима. Познате су италијанска, руска, енглеска и турска дама (на шаховској табли од 64 поља) или канадска (табла од 144 поља).



Ваш задатак је да коришћењем правила игре дама, реализујете интелигентне играче за ову игру. Препоручује се да прочитате детаљнија правила игре из *Додатка 1*.

Циљ домаћег је да реализујете два играча. Прво ћете креирати основног „алфа-бета играча“, који ће користити алфа-бета одсецање, са итеративним продубљивањем. Након креирања алфа-бета играча, креираћете и „такмичарског играча“ надоградњом и побољшањем карактеристика вашег алфа-бета играча. Приликом креирања такмичарског играча, можете имплементирати бољи алгоритам претраживања, осмислити бољу статичку функцију процене, или користити друге технике према вашем избору (изузев коришћења нити, GPU програмирања и сличних ствари). Ваш такмичарски играч ће бити унет у класу турнир, која ће се користити за налажење најбољих играча у групи. Имплементација се ради у програмском језику Јава, а за развој се препоручује алат *Eclipse IDE*.

Фрејмворк дат у архиви (*Checkers.zip*) садржи највећи део потребног кода, тако да количина кода која је потребна да се напише није велика.

Фрејмворк

Фрејмворк је подељен на неколико Јавиних пакета:

- `etf.checkers`: Овај пакет обезбеђује основне функционалности фрејмворка и детаљно је описан у наставку.
- `etf.checkers.ui`: Овај пакет имплементира кориснички интерфејс овог фрејмворка. Овај кориснички интерфејс омогућава интерактивном играчу да одабере потез кроз `etf.checkers.ui.HumanPlayer`.

- `etf.checkers.demo`: Овај пакет демонстрира употребу фрејмворка са играчем који бира насумичне потезе (`etf.checkers.demo.RandomPlayer`).

Сваки студент треба да направи свој пакет `etf.checkers.<username>` где је `<username>` Ваше корисничко име за портал e-Студент (у формату *piGGBBBBd,* где су *pi* иницијали - презиме и име, ознака *GG* приказује последње две цифре године уписа факултета, а ознака *BBBB* приказује четвороцифрени број индекса, проширен водећим нулама). Студенти су обавезни да поставе у том пакету било који код који пишу или фајлове које су модификовали. У наставку текста неке класе ћемо називати само по имену, без комплетног назива ком пакету припадају.

Структуре података

Пре почетка израде игре, треба се упознати са следећим карактеристикама:

- **Side**: Постоје две стране, црвена и црна, које су представљене као целобројне променљиве 0 и 1, респективно. Поред тога, неке класе користе ниједну страну (`NEITHER`), која представља непознату или неодређену страну. У коду су дефинисана поља: `RED`, `BLK` и `NEITHER` у `CheckersConsts`.

Страна	Назив поља	Вредност (`int`)
Црвена страна	`RED`	0
Црна страна	`BLK`	1
Ниједна	`NEITHER`	2

- **Checkers Pieces**: Фигуре представљене су као целобројне променљиве, користећи поља `RED_PAWN`, `BLK_PAWN`, `RED_KING` и `BLK_KING` `CheckersConsts`. Ова структура укључује такође и празан квадрат (`BLANK`).

Фигуре	Назив поља	Вредност (`int`)
Црвени пион	`RED_PAWN`	0
Црни пион	`BLK_PAWN`	1
Празан квадрат	`BLANK`	2
Црвени краљ	`RED_KING`	4
Црни краљ	`BLK_KING`	5

  Треба нагласити да последња два бита у овим фигурама показују страну фигуре у чијем је власништву. На пример, власник `BLK_KING` ($5 = 0101_b$) је `BLK` ($1 = 0001_b$), или власник `BLANK` ($2 = 0010_b$) је `NEITHER` ($2 = 0010_b$).

- **Location**: Дводимензионална табла ове игре је индексирана по редовима. Ако гледамо из угла црвеног, квадрати табле су индексирани са лева удесно, па одозго надоле, почевши од 0, према следећој слици:

BLK							
	01		03		05		07
08		10		12		14	
	17		19		21		23
24		26		28		30	
	33		35		37		39
40		42		44		46	
	49		51		53		55
56		58		60		62	
RED							

- **Board State:** Стање табле је приказано као једнодимензионални низ `int[]`, низ чији индекси представљају 64 локације (квадрата) на табли. Елемент са индексом `i` је целобројна представа фигуре на локацији `i`; елемент је `BLANK`, ако је локација `i` празна.

- **Move:** Потез је представљен секвенцом локација, кроз које фигура путује. Непроменљиви потез, који не може бити модификован када се једном формира, приказан је класом `Move`. Класа `MutableMove` приказује потезе који се могу мењати. Заједно `Move` и `MutableMove` су потклассе генеричке колекције `java.util.List<Integer>`.

Имплементација фрејмворка

Садржај пакета: `etf.checkers`

Свака од класа у пакету `etf.checkers` може се сврстати у једну од две групе. Прва група садржи све инсталације неопходне за покретање игре, укључујући часовнике за контролу времена, петље догађаја за напредак у игри, објекте за контролу играчевих нити. Није потребно модификовати већину ових класа. Класе у овој првој групи су:

CheckersModel - одржава информације о стању игре, укључујући стање табле, ко је на потезу и врши временске прорачуне.

CheckersController - контролише напредовање игре делујући на инстанцу `CheckersModel`. Ова класа контролише прорачунске нити и спроводи контролу времена.

Checkers - парсира опције командне линије, иницијализује игру и покреће кориснички интерфејс. Ова класа садржи главни програм `main()`.

CheckersPlayer - бира потезе који ће бити одиграни на табли. Ваши играчи алфа-бета и такмичарски играч ће проширити ову класу. Пример како се проширује класа дата је у: `demo.RandomPlayer`.

CountdownClock - описује методе повезане са сатом који одбројава. Овај интерфејс је имплементиран од стране `DefaultCountdownClock`.

GameClock - описује методе играчког сата. Овај интерфејс је имплементиран од стране `Default-GameClock`.

Друга група класа садржи структуре описане у секцији изнад - „Структуре података“, а такође садржи помоћне методе које раде над тим структурама. Класе у овој групи се користе у целом фрејмворку:

CheckersConsts - садржи фигуре и репрезентацију стране. Потребно је да у пројекат статички увезете сва статичка поља ове класе, тако што ћете користити декларацију: `import static etf.checkers.CheckersConsts.*;`

Move - представља непроменљиви потез, као низ локација.

MutableMove - представља променљиви потез (потез који се може мењати) као низ локација.

Utils - обезбеђује статичке методе које користе структуре описане у секцији изнад - „Структуре података“. Ви ћете користити ову класу често, тако да треба прочитати пажљиво документацију за ту класу.

BoardState - обезбеђује објектно-оријентисану представу стања табле, која се иначе чува као низ целих бројева (`int[]`). Ако се одлучите да користите ову класу, такође треба прочитати пажљиво документацију за ту класу.

Evaluator - описује интерфејс за статичку функцију процене. Овај интерфејс се имплементиран је од стране `SimpleEvaluator`. Нисте у обавези да користите овај интерфејс. Треба имати на уму да метод `evaluate()` враћа резултат са тачке гледишта црвеног, тако да морате негирати функцију процене да добијете резултат са тачке гледишта црног.

Као пример како се користе ове класе, треба да погледате: `demo.RandomPlayer` и `demo.DemoPlayer`.

Структура фолдера

Овај фрејмворк користи следећу структуру:

- Изворни фолдер (**`src/`**) - садржи све изворне фајлове класа из пакета `etf.checkers`
- Изворни фолдер за кориснички интерфејс (**`src/etf/checkers/ui/`**) - садржи све изворне фајлове за пакет `etf.checkers.ui`
- Кориснички фолдер (**`src/etf/checkers/<username>/`**) - садржи све фајлове које Ви креирате и можете ручно направити овај фолдер.

Алати

За изградњу овог пројекта препоручује се окружење *Eclipse IDE (https://www.eclipse.org/)*. У њему је потребно направити нови, празан Јава пројекат, извршити увоз свих јавиних фајлова у изворни директоријум (након отпакивања). Занемарити упозорења. Ви можете програм покренути у *Eclipse*-у или можете да га извезете као JAP фајл. Детаљно упутство дато је у *Додатку 2*.

Извршавање

Након изградње фрејмворка, покренути га следећом командом:
`java -jar Checkers.jar <red_player> <blk_player> [options]`
где су `red_player` и `blk_player` скраћени називи за класе црвених и црних играча.

Скраћени назив класе може да се прошири до потпуног додавањем на почетку "`etf.checkers.`" и додавањем на крају "`Player`". На пример `etf.checkers.demo.RandomPlayer` скраћени назив класе је `demo.Random` и подешавање `red_player := demo.Random` поставља случајног играча на црвено.

Следећи параметри могу бити опциони у овој игри:

`--turntime <turnLimit>` – прецизира колико дуго, у милисекундама (подразумевано = 1000), играчи имају на располагању за свој потез. Ово се не примењује на интерактивне играче.

`--step` - захтева клик миша пре почетка сваког потеза. Подразумевано, потез почиње аутоматски са малим закашњењем.

`--verbose` - коментари штампани на излазу. Класе могу приступити овој опцији преко поља `Utils.verbose`

`--initbs <fileName>` - поставља стање табле дефинисано називом `fileName` као почетно стање табле. Овај параметар је користан за исправљање багова. Погледати очекивани формат фајла `fileName` у следећем: `Utils.parseBoardState`.

`--initside <side>` - означава која страна игра прво. Ова опција је корисна за исправљање багова (подразумевано = `RED`).

Пример команде:
`java -jar Checkers.jar ui.Human <username>.AlphaBeta --turntime 3000`

Ова команда ће покренути игру између интерактивног играча (RED) и Вашег алфа-бета играча (BLK), при чему алфа-бета играч има 3 секунде по потезу. Ако добијате `TimeoutException` приликом дебаговања, користите `--turntime -1`

Захтеви домаћег задатка

Ваш задатак је да направите два играча, који ће комуницирати са обезбеђеним фрејмворком. Прво треба направити алфа-бета играча, који користи алфа-бета одсецање са итеративним продубљивањем приликом претраживања стабла игре. Овај играч служи да покаже Ваша знања о основним методама у претраживању стабла игре. Након тога потребно је да побољшате особине алфа-бета играча користећи технике по Вашем избору. Овог играча ћемо називати такмичарским. Турнир ће бити покренут коришћењем такмичарских играча сваког студента, од којих ће најбољи добити додатне поене на домаћем задатку.
За максималан број поена, програм мора да испуни све услове пројектног задатка и да ради поуздано.

Захтев #1 - Алфа-бета играч

Први од два играча који треба имплементирати је алфа-бета играч, који приликом претраживања користи алфа-бета одсецање. Морате користити методу `getAllPossibleMoves()` да преузмете листу могућих потеза за сваки нетерминални чвор, и морате испитати ове потезе у редоследу у коме сте их добили. Користите SBE (*static board evaluator*) у `SimpleEvaluator` да процените терминалне чворове. Имплементација овог играча коришћењем итеративног продубљивања по дубини, који инкрементира дубину одсецања стабла игре за по један ниво у свакој итерацији. Ово инкрементирање дубине одсецања ће бити настављено до максималне дубине `depthLimit` или док време не истекне, односно шта се прво деси.

Поред тога, морате избројати број подстабала одсечених коришћењем алфа-бета претраживања. То значи да је потребно да пребројите колико пута се јавила операција одсецања, а не колики је укупан број чворова у одсеченом подстаблу. Дакле, ако чвор има петоро деце, али се одсецање јавило после обиласка прва два детета (и њихових придружених подстабала), онда треба повећати бројач одсецања за 3, односно за број чворова-деце који нису посећени. Такође, пратите коначне минимакс вредности израчунате на последњем одсеченом чвору (односно ако дати родитељски чвор одсече дете, треба сачувати минимакс вредност родитеља; враћајте вредност последњег одсеченог чвора током претраживања).

Алфа-бета играч је имплементиран у функцији `calculateMove()` као метод потклассе `CheckersPlayer`. Не стављајте одабрани потез у повратну вредност методе `calculateMove()`; уместо тога сачувајте одабрани потез помоћу `setMove()`. Контролер игре ће приступити одабраном потезу позивањем методе `getMove()`. Приметите да сваки корак у итеративном продубљивању може произвести другачији најбољи потез. Дакле, треба позвати методу `setMove()` након сваке итерације, тако да када контролер игре прекида низ израчунавања на којој је покренуто `calculateMove()`, метода `getMove()` враћа најбољи потез до тада пронађен. Да би могли да добијемо исправне потезе из датог стања, обезбеђена је и метода (погледати: `demo.DemoPlayer.java` и `demo.RandomPlayer.java`):

```
List<Move> possibleMoves = Utils.getAllPossibleMoves(boardState, side);
```

Ова статичка метода враћа све могуће исправне потезе за дато стање табле и дату страну. Докле год се одлучујете за потез са листе, Ви не морате да бринете о валидности потеза. Ако имамо један скок на располагању, на пример `possibleMoves` листа ће садржати скок и ништа друго.

Поставите алфа-бета играча у класу `etf.checkers.<username>.AlphaBetaPlayer` и именујте изворни фајл `src/<username>/AlphaBetaPlayer.java`

Скелетон:

Скелетон алфа-бета играча дат је у: `src/demo/AlphaBetaPlayer.java`
Овај скелетон је једноставан да се заврши: прво ископирајте фајл у `src/<username>/` и промените декларацију пакета на врху фајла:
`package etf.checkers.<username>;`

Све што треба имплементирати је метода `calculateMove`() која обавља алфа-бета одсецање. Овај код ће изгледати као псеудо-код за алфа-бета претраживање дат у слајдовима са предавања и вежби. Потребно је да имплементирате код за изградњу и пролазак кроз структуре података стабла претраживања. Један од начина да се то уради је да се користи класа `Utils.java`, која има методе `execute()` и `revert()`. Ви можете извршавати потезе и мењати таблу, а затим вратити таблу назад у оригинално стање претраживањем уназад. Дакле, за свако стање можете извршити потез који ће резултирати у чвору детету, рекурзивно позивати алфа-бета претраживање са дететом, а затим вратити дете назад на оригинално стање табле. Друга могућност је да имплементирате Вашу класу *Чвор*, која ће садржати информације, као што су родитељ, листа деце, стање табле, алфа вредност и бета вредност. Ваш алфа-бета играч треба да имплементира интерфејс `GradedCheckersPlayer`, који пружа методе `getPruneCount()` и `getLastPrunedNodeScore()`. Имплементацијом тих метода, добићете број одсечених подстабала и вредност последњег одсеченог чвора, током последње итерације итеративног-продубљивања. Можете претпоставити да ове методе неће бити позване, док Ваш играч врши израчунавање.

Захтев #2 - Такмичарски играч

Након креирање алфа-бета играча, потребно је направити другог играча, који треба да оптимизује перформансе. Можете укључити било које технике које желите (изузев нити, GPU програмирања или других сличних ствари) за реализацију овог такмичарског играча, који ће се користити за учешће на турниру против других играча. Почните од надоградње постојећег алфа-бета играча. Алфа-бета играч користи веома једноставну статичку функцију процене (SBE), користећи само број преосталих црвених и црних фигура на табли. Ваш задатак је да у овом играчу користите боље статичке функције процене, које користе више информација из стања табле. Поред креирања сопствене статичке функције процене, можете побољшати процедуру претраживања коришћењем алфа-бета метода.

Имплементирајте Вашег такмичарског играча у класи `<Username>Player` пакета `etf.checkers.<username>` и назовите изворни фајл `src/<username>/<Username>Player.java`. Напомена да `<Username>` представља `<username>` са првим великим словом.

Ваш такмичарски играч треба да имплементира најмање три карактеристике, и најмање једна од тих карактеристика треба да буде побољшана функција процене.

Поред тога, треба написати извештај, који описује Вашег такмичарског играча. Извештај треба да садржи:

- Опис функције процене, објашњавајући шта израчунава и зашто је добра
- Описе свих других функција или карактеристика, које такмичарски играч користи
- Коментаре о томе како Ваш такмичарски играч наступа против алфа-бета играча или људи, наводећи предности и слабости, које ће се вероватно испољити

Извештај приложити уз код, у фолдеру `/documentation` и именовати га као: `Prezime_Ime_ES_Domaci_2014.doc (docx).`

Напомене:

Пакети: Будите сигурни да се сви фајлови, које сте писали или модификовали, налазе у пакету `etf.checkers.<username>` и фолдеру `src/<username>/`. Сви други фајлови ће аутоматски бити обрисани пре оцењивања.

Синтакса: Оба Ваша играча, и алфа-бета и такмичарски играч треба да штампају извештаје о отклоњеним грешкама на `stdout`, када је флег `--verbose` укључен. Ви можете приступити статусу тог флега референцирањем поља `Utils.verbose`. Ваши играчи треба да минимално одштампају 3 ствари:

1. Најбољи потез и његову коначну вредност.
2. Број одсечених подстабала, на крају сваке итерације итеративног продубљивања.
3. Минимакс вредност од последњег одсеченог чвора на крају сваког корака процеса итеративног продубљивања. Насупрот томе, Ваш програм не треба да штампа ништа, ако је флег искључен.

Архива: Електронску верзију решења овог пројекта послати најмање 2 дана пре усмене одбране, као ZIP архиву (која садржи два фолдера: `/checkers` и `/documentation`). Фолдер `/checkers` треба да садржи модификовану верзију кода скелетона, заједно са пропратним кодом који сте писали. Упутство за слање решења пројекта биће објављено на предметној мејлинг листи/страници предмета крајем маја.

Пројекат из предмета Експертски системи се ради самостално и није обавезан за полагање испита (на испиту се може заменити са теоретским питањима из целокупног градива). Пројекат се може бранити само у јунском испитном року. Пројекат вреди максимално 20 поена (уз могућност додатних поена за најбоље радове).

На усменој обрани кандидат мора самостално да инсталира све потребне програме потребне за приложено решење (уколико не постоје у лабораторији). Кандидат мора да поседује потребан ниво знања о задатку, мора да буде свестан недостатака приложеног решења и могућности да те недостатке реши. Кандидат мора тачно да одговори и на одређен број питања која се баве тематиком пројекта.

Кандидати који имају питања треба да се јаве предметном асистенту на мејл.

ДОДАТАК 1 - Правила игре „Дама“

How To Play Checkers (Using Standard U.S. Rules)

Checkers, called Draughts in most countries, has been traced back to the 1300s, though it may indeed stretch further into history than that. These are the standard U.S. rules for Checkers.

Difficulty: Easy

Time Required: 15 minutes

Here's How:

1. Checkers is played by two players. Each player begins the game with 12 colored discs. (Typically, one set of pieces is black and the other red.)
2. The board consists of 64 squares, alternating between 32 dark and 32 light squares. It is positioned so that each player has a light square on the right side corner closest to him or her.
3. Each player places his or her pieces on the 12 dark squares closest to him or her.
4. Black moves first. Players then alternate moves.
5. Moves are allowed only on the dark squares, so pieces always move diagonally. Single pieces are always limited to forward moves (toward the opponent).
6. A piece making a non-capturing move (not involving a jump) may move only one square.
7. A piece making a capturing move (a jump) leaps over one of the opponent's pieces, landing in a straight diagonal line on the other side. Only one piece may be captured in a single jump; however, multiple jumps are allowed on a single turn.
8. When a piece is captured, it is removed from the board.
9. If a player is able to make a capture, there is no option -- the jump must be made. If more than one capture is available, the player is free to choose whichever he or she prefers.
10. When a piece reaches the furthest row from the player who controls that piece, it is crowned and becomes a king. One of the pieces which had been captured is placed on top of the king so that it is twice as high as a single piece.
11. Kings are limited to moving diagonally, but may move both forward and backward. (Remember that single pieces, i.e. non-kings, are always limited to forward moves.)
12. Kings may combine jumps in several directions -- forward and backward -- on the same turn. Single pieces may shift direction diagonally during a multiple capture turn, but must always jump forward (toward the opponent).
13. A player wins the game when the opponent cannot make a move. In most cases, this is because all of the opponent's pieces have been captured, but it could also be because all of his pieces are blocked in.

Tips:

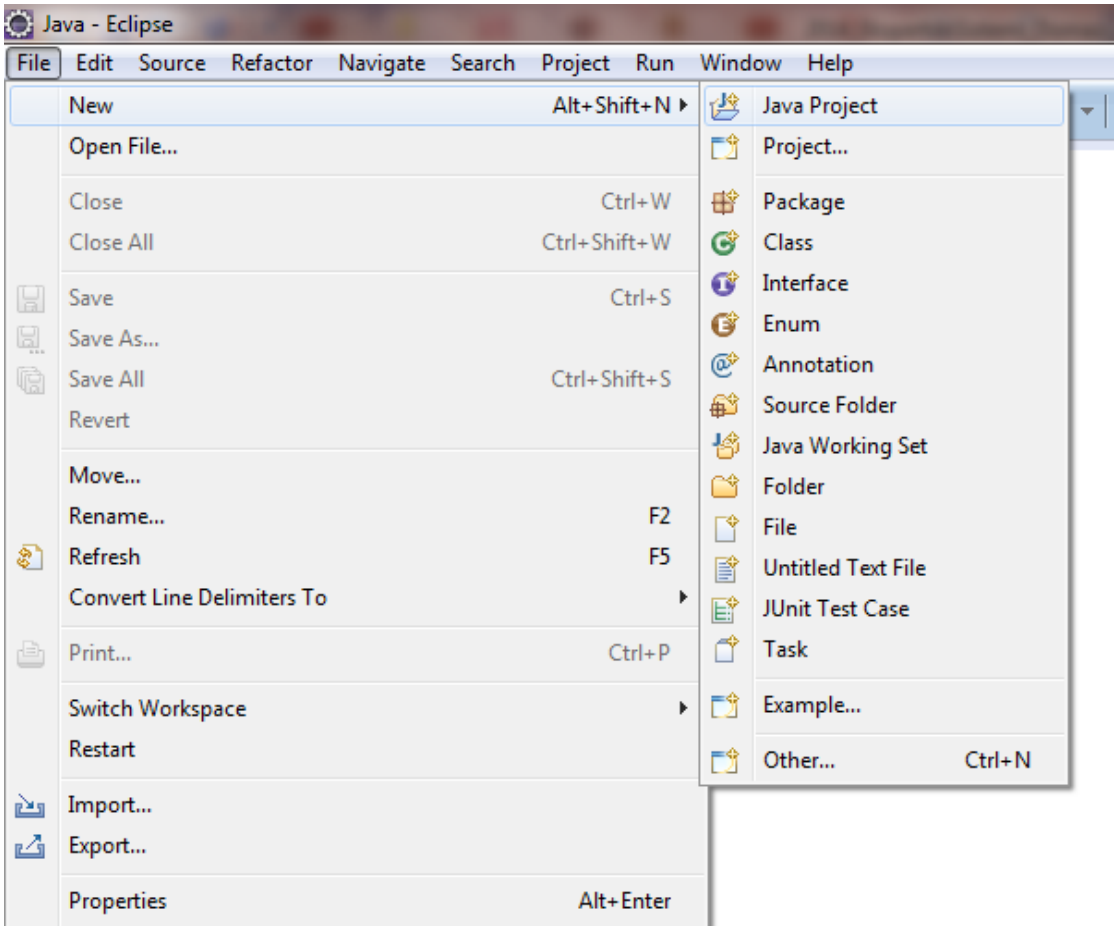
1. Checkers (using the U.S. rules) uses the same board as Chess. Many sets comes with the pieces needed to play both games.
2. Many different games can be played using the basic 8x8 Checkers board and pieces. In fact, it's not too hard to come up with your own variants. And here's a collection of free games for an 8x8 board that can be played with game pieces you probably already have.
3. 24 discs (12 of 2 colors)

Извор: http://boardgames.about.com/cs/checkersdraughts/ht/play_checkers.htm

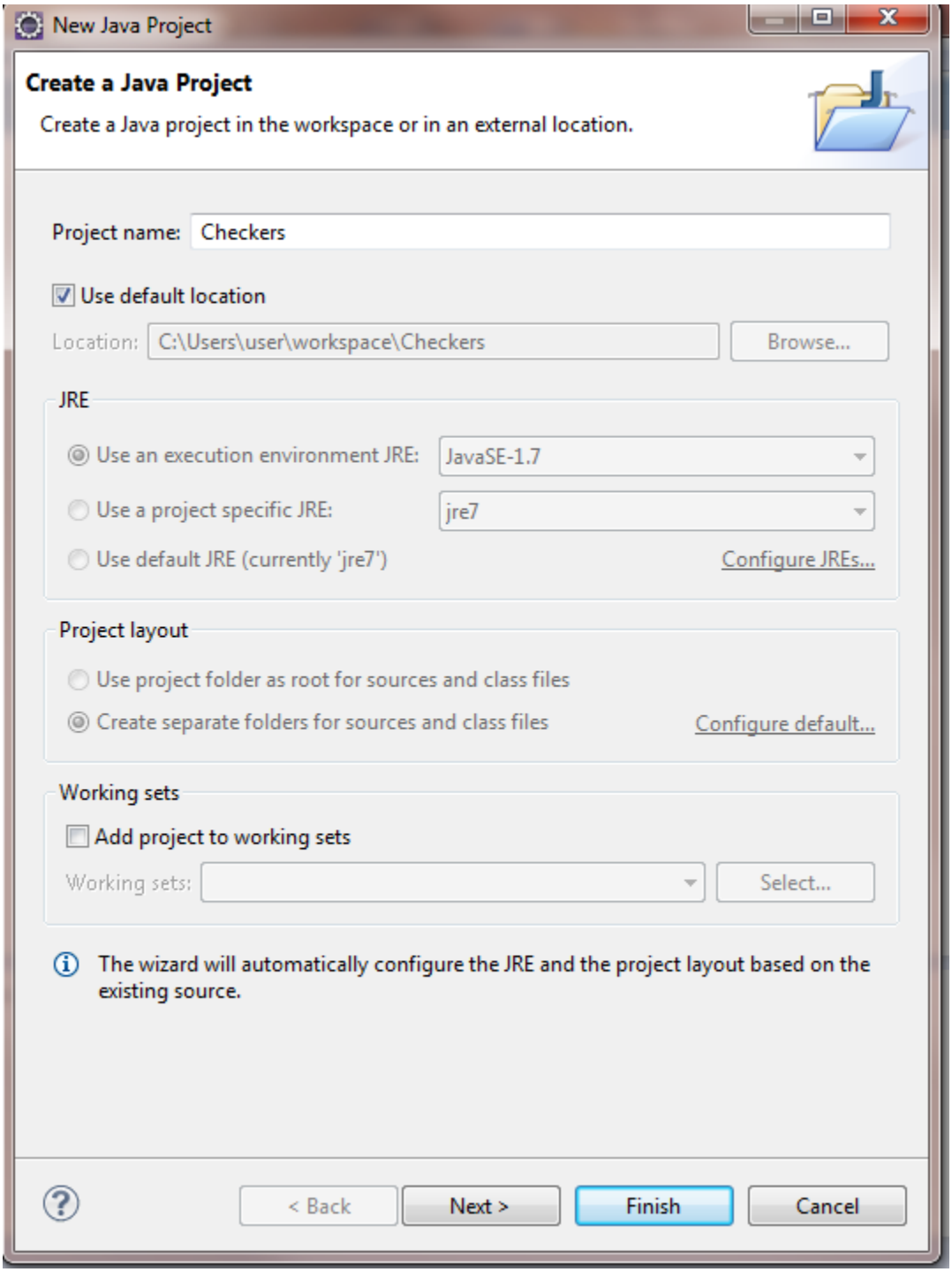
ДОДАТАК 2 - Упутство за коришћење фрејмворка у оквиру окружења *Eclipse IDE*

У радни фолдер окружења (најчешће *C:\users\User\workspace*) отпаковати архиву *Checkers.zip* и следити следеће кораке:

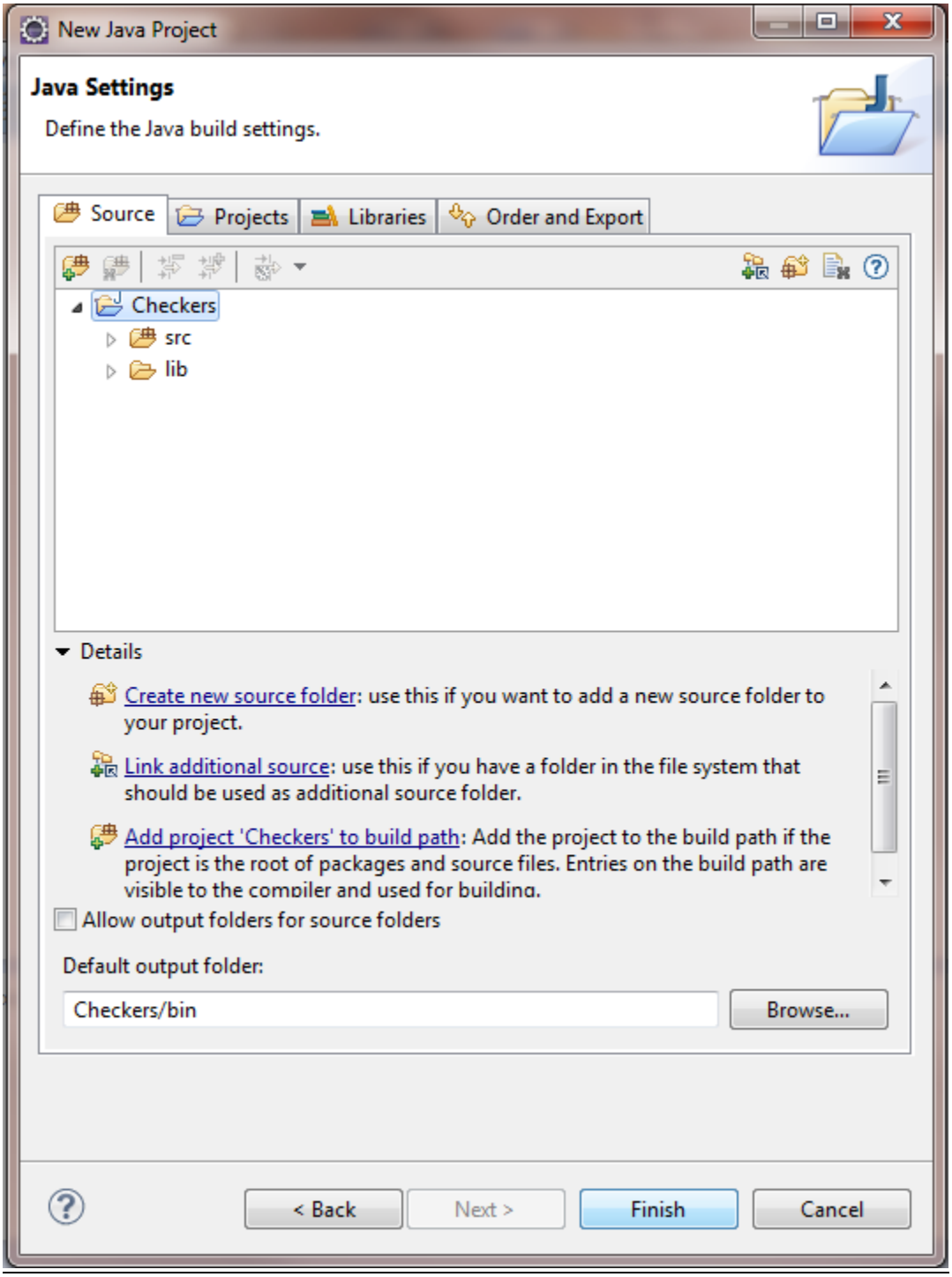
1. Отворити мени **File** и кликнути на **New**. Кликнути на **Java Project** како бисте покренули прозор **New Java Project**.



2. У оквиру прозора **New Java Project**, поставити у поље за име пројекта *Checkers*. Потребно је да буде чекирано и **Use default location**.



3. Кликнути на дугме **Next** да бисте прешли на следећи екран.



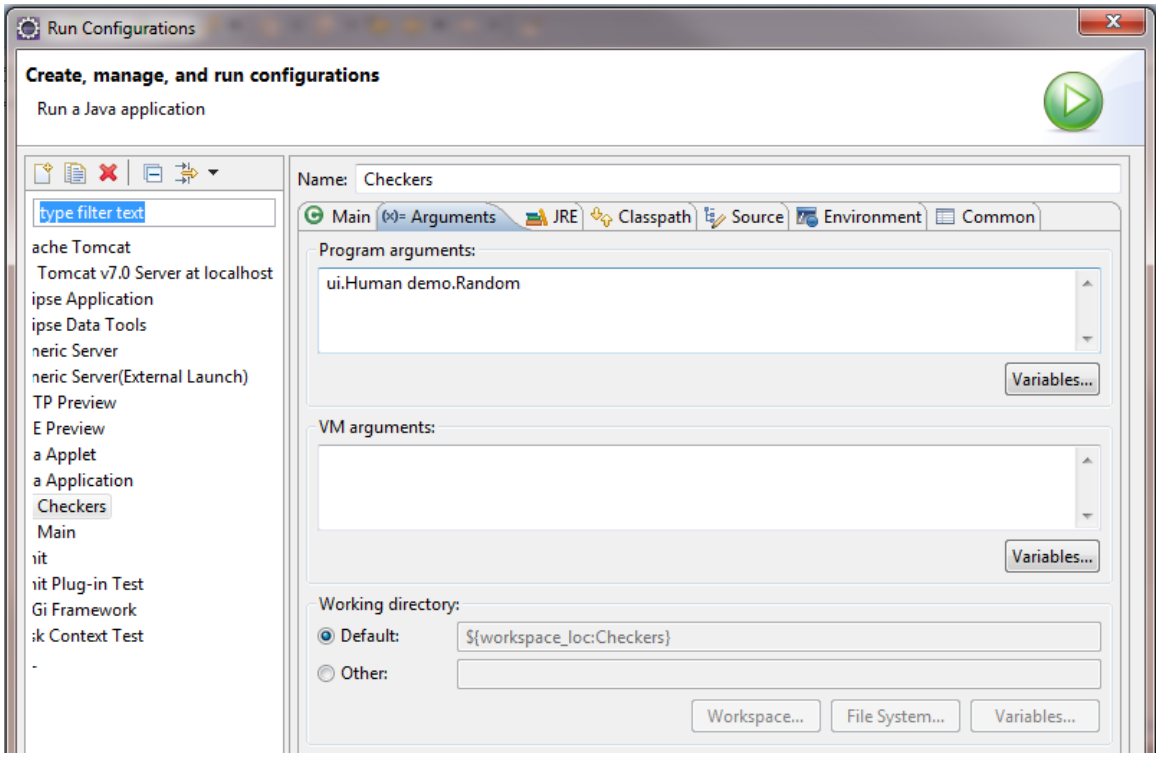
4. Кликнути на дугме **Finish** да бисте успешно направили пројекат.

5. Притиснути **Ctrl+F11** да би се отворио **Run As** дијалог. Одабрати **Java Application** и притиснути **OK**. Сет корисничких инструкција биће одштампан у табу **Console**, зато што ниједан аргумент није наведен за *main( )*.

6. Да бисте поставили неке аргументе приликом извршавања главног програма *main( )*, кликните десним дугметом миша на пројекат, одаберите **Run As** и кликните на **Run Configurations.** Треба да се отвори активни пројекат *"Checkers"*, који смо убацили у *Eclipse*. Отворити **Arguments** таб за ту конфигурацију, и поставити аргументе у пољу **Program Arguments**:

На пример, аргументи могу бити:

```
ui.Human demo.Random
demo.Random drasko.Drasko --turntime 2000
```



ДОДАТАК 3 - Корисни линкови

- WIKIPEDIA, Igra “DAMA”
  http://sr.wikipedia.org/wiki/%D0%94%D0%B0%D0%BC%D0%B0_(%D0%B8%D0%B3%D1%80%D0%B0)

- http://aitopics.org/topic/checkers

- Rules for playing Checkers
  http://boardgames.about.com/cs/checkersdraughts/ht/play_checkers.htm

- Play Chinook - Checkers program
  http://webdocs.cs.ualberta.ca/~chinook/