KTH H16P02
Network Programming with Java
HW1 notes - due 18.11.16

Communication stuff

We're using BufferedReader and BufferedWriter.

Streams (byte streams) -> reader/writer (character "streams") -> BufferedReader/BufferedWriter (read chunks at a time, e.g. work with whole lines)

ObjectInputStream/ObjectOutputStream … sends heila Java objects (í raun byte stream) milli JVM. Þeir eru marshallaðir og unmarshallaðir sjálfkrafa.

Við erum að serialize-a og deserialize-a í JSON sjálfir, og vinna með text based protocol þannig, þó það sé óþarfi. Hefðum getað verið með object stream og sent hlutina beint á milli, þar sem að þetta er sama java forritið bæði í client og server og þau deila öllum klösunum sem eru sendir þarna á milli (shared mappan).

http://stackoverflow.com/questions/4602060/how-can-i-get-mime-type-of-an-inputstream-of-a-file-that-is-being-uploaded
"If you are uploading the file through a socket it will be your responsibility to specify the MIME type as part of your protocol as you will not inherit the http headers."

- Varðandi það að lesa file af mismunandi gerðum í gegnum socket

Hvernig við lesum inn heil messages

Við erum að lenda í vanda þegar við erum að reyna að nota bufferedreader og writer (semsagt að vinna með strengi en ekki bytes) þegar við erum að senda á milli client og server.
Hann hættir aldrei, hangir á readLine(), alveg þangað til socket er closed (það er ekkert EOF (end of file) í þessum socket samskiptum).
http://stackoverflow.com/questions/15538509/dealing-with-end-of-file-using-bufferedreader-read
Leysum þetta með því að vinna með byte í staðinn
NOPE
Ef við erum að vinna með bytes þá lendum við í því að vera með eitthvað byte[4096] eða eithvað rugl. Þurfum að setja limit á hversu mörg bytes við getum tekið við.
Við getum leyst þetta með því að vinna með BufferedReader og segja sem svo að client sé búinn að senda okkur þegar við fáum bara newline. Fáum semsagt tóma línu. Þá breakum við út úr while lúppunni.

```
while ((incomingLine = in.readLine()) != null && incomingLine.length() > 0)
```

KTH H16P02
Network Programming with Java
HW1 notes - due 18.11.16

Json vs sending objects

Við getum sleppt því að nota Json og sent objects á milli. Notað samt sömu request og response objectana og við erum að nota í Jsoninu. Það er algjör óþarfi að vera að nota Json þegar server og client er partur af sama kóðabasanum og skrifað í sama málinu, við erum meira að segja að deila request og response objectunum á milli server og client. En maður myndi samt alltaf nota Json eða eitthvað álíka (samskiptamál óháð forritunarmálum) í alvöru kerfi. Spurning hvort við tölum ekki bara um þetta í kynningunni.

Við notuðum gson libraryið til að vinna með json.

GameService.getRandomWord()

Við erum að scanna í gegnum allan fælinn til að finna random word.
Við hefðum getið tekið annað approach með því að nota RandomAccessFile og fara randomly einhversstaðar inn í fælinn, og skanna svo til hægri og vinstri til að finna næsta newline til vinstri og til hægri, og taka svo orðið á milli þeirra. Þyrftum þá að passa upp á að fara ekki of langt til vinstri (framhjá byrjuninni á fælnum) og of langt til hægri (framhjá endanum á fælnum) og þyrftum líka að passa upp á ef að fyrsti staðurinn sem við dettum inná randomly er newline. Þetta var of mikið vesen þannig að við beiluðum á þessu.

Read words.txt

Mega vesen að lesa words.txt þegar það er búið að pakka applicationinu í JAR file. Endaði á því að lesa fælinn sem stream, pumpa honum út í temporary file, og nota það. Þá virkar þetta í JAR file.

Sjá http://stackoverflow.com/questions/941754/how-to-get-a-path-to-a-resource-in-a-java-jar-file

“This is deliberate. The contents of the "file" may not be available as a file. Remember you are dealing with classes and resources that may be part of a JAR file or other kind of resource. The classloader does not have to provide a file handle to the resource, ***for example the jar file may not have been expanded into individual files in the file system.***
Anything you can do by getting a java.io.File could be done by copying the stream out into a temporary file and doing the same, if a java.io.File is absolutely necessary.”

KTH H16P02
Network Programming with Java
HW1 notes - due 18.11.16

Synchronized

Við erum bara að díla við thread synchronization í addGame og addPlayer. Þessi tvö föll eru merkt synchronized. Þetta höldum við að sé nóg, hættan er að ef að menn eru að búa til player eða leik á sama tíma, þá gætu þeir endað með sama ID. Þetta leysir það, því að aðgangurinn að getNewId föllunum er bara í gegnum þessi föll og er því synchronized.

Synchronized merkt föll taka lock á objectinn sem þau eru í. Ef það er static fall sem er merkt synchronized þá nær lockinn yfir öll static föll í klasanum. Ef það er ekki static fall þá nær lásinn yfir öll instance methods í klasanum, skilst mér.

GUI client

JavaFX.
Service pælingarnar. Það er service klasi out of the box með JavaFX sem extenda Worker. Hendum inn CommandInterface í það (onSucceeded, onFailure) sem kallað er svo í út frá nafnlausum föllum sem eru skilgreind í setOnSucceeded() og setOnFailed() föllunum í Service.

Við notum þetta til að hafa UI-ið multithreaded. Blockar ekki UI-ið þegar við erum að taka kall í serverinn.

Öll köll í serverinn eru wröppuð í sitt eigið svona service.

Test client

Við erum með test client sem var notaður við þróunina á servernum. Bara command line interface. Hann keyrir test scriptur á serverinn, t.d. búa til notanda og senda nokkur guesses.

Database pælingar

Beiluðum á honum. Hefðum getað notað objectstreams + filestreams til að serialize-a hluti í file og vista þá þannig. Deserializea svo með ObjectInputStream til að ná þeim út úr fælnum aftur.

KTH H16P02
Network Programming with Java
HW1 notes - due 18.11.16

JSON calls

Responses:

Only two types of responses from server: ResGameState and ResInvalidRequest.

Game state (response to all valid requests from client):

```
{
   "playerId":"0",
   "playerScore":"0",
   "gameId":"0",
   "numberOfGuessesLeft":19,
   "gameStateString":"_____e_a__e",
   "gameState":"InProgress",
   "guessedCharacters":[
      "e",
      "a"
   ],
   "responseType":"GameState"
}
```

Invalid request:

```
{
   "responseType":"InvalidRequest"
}
```

Requests:

Create player and start game:

```
{
   "requestType":"CreatePlayerAndStartGame"
}
```

KTH H16P02
Network Programming with Java
HW1 notes - due 18.11.16

Start game (with known player):

```
{
   "playerId":"1",
   "requestType":"StartGame"
}
```

Add a guess to a game:

```
{
   "guess":"l",
   "gameId":"1",
   "requestType":"Guess"
}
```

End a game:

```
{
   "gameId":"1",
   "requestType":"EndGame"
}
```