2. Rühmatöö

# Elektri ööpäeva tunnihindade analüüs

Aines:

Objektorienteeritud projekteerimine

(IT mitteinformaatikutele)

Jaak Ivask, Peeter Niidas

2020/21 kevad

# Projekti kirjeldus

2. projekti eesmärk on täiendada ning parandada 1. projekti käigus tehtud elektrihinna päringute rakendust. Üldisem eesmärk on endiselt säästva elektritarbimise propageerimine läbi elektri tunnihindade analüüsi. Antud rakendust on võimalik kasutada näiteks elektri tunnihindade muutuse demonstreerimiseks või huvi korral erinevate hindade päringute tegemiseks ning faili (csv formaadis) hilisemaks töötlemiseks salvestamiseks.

# Märkus programmi kasutamiseks

Antud rakendus toetub oma töös 3 library kasutamisele:

* json (json-simple-1.1.11.jar)
* junit (junit-4.101.jar)
* hamcrest (hamcrest-core-1.11.jar)

# Programmi klassid

## Klass EleringJsonApi

Pöördub Eleringi API poole ning teeb päringu vastavalt etteantud ajaperioodile ning tagastab JSON objekti. Kasutatavad meetodid on:

* Konstruktor (restEndPoint) - määrab vaikimisi Elering API endpointi.
* getEleringData() – kasutab väärtustatud isendivälju. Tagastab JSON objekti. Lisaparameetreid ei ole
* Private setEleringApiUrl() - Koostab vastavalt väärtustatud algus ja -lõppkuupäevale, baasurl ja API endpointile päringu URL. Kutsutakse välja meetodis getEleringData().
* Setter meetodid restEndPoint, alg ja lõpukuupäevale.

## Klass Elektrihind

Klass elektrihind sisaldab konstruktorit elektritarbimise aja ja hinnaga, antud parameetrite setter’eid ning implementeerib Comparable liidest sorteerimaks andmeid elektrihinna alusel.

## Klass ElektriHindPäev

Klass elektrihindPäev laiendab klassi elektrihind võimaldamaks koguda ning töödelda pikema perioodi andmeid. ElektriHindPäev klass käsitleb elektrihind klassi muutujat hind perioodi keskmise hinnana, lisab aga juurde perioodi minimaalse ja maksimaalse hinna parameetri. Klass sisaldab vastavat konstruktorit ja setter’erid.

## Klass KuvaElektriHind

Klass teostab Elering’i API kaudu saadud json formaadis andmete (eel)töötlust ning valmistab andmed ette kasutajale vajalikus formaadis väljastamiseks. Kasutatakse järgmisi meetodeid:

* loeJson
  + saab sisendina Json ojekti ja riigi tähise, loeb eeltoodud objektist timestamp’i, teisendab selle meile sobivasse formaati meetodiga tunnidTimestampist, loeb elektrihinna, arvutab selle ümber EUR/MWh -> senti/kWh ning kannab need ArrayList’i elektrihind klassi Elektrihind kasutades
* loeJsonKuu
  + saab sisendina Json ojekti ja riigi tähise, loeb eeltoodud objektist info, leiab iga päeva minimaalse maksimaalse elektrihinna, arvestab päeva keskmise, arvutab hinnad ümber EUR/MWh -> senti/kWh ning kannab need ArrayList’i päevaHinnad klassi ElektriHindPäev kasutades. Klassi Elektrihinnad kasutades lisab päeva maksimumid ArrayList’i maxHindadeList, päeva miinimumid ArrayList’i minHindadeList, kõik hinnad ArrayList’i perioodiHinnad.
* leiaHinnad, leiaMaxHind, leiaMinHind, leiaKeskmised
  + 24 h ettevaatava perioodi info töötlusse suunamise ja tulemuse väljastamise funktsioonid
* leiaPerioodiHinnad, leiaPerioodiMaxHind, leiaPerioodiMinHind, leiaPerioodiKeskmine
  + 1 kuuse perioodi info töötlusse suunamise ja tulemuse väljastamise funktsioonid
* topUp
  + saab sisendina numbrilise parameetri Top’i pikkuse jaoks, ArrayList’i võrreldavate andmetega ja ArrayList’i väljundandmete jaoks. Viimasesse salvestab andmed kahanevas järjekorras.
* topDown
  + saab sisendina numbrilise parameetri Top’i pikkuse jaoks, ArrayList’i võrreldavate andmetega ja ArrayList’i väljundandmete jaoks. Viimasesse salvestab andmed kasvavas järjekorras.
* leiaKeskmine
  + meetod, mis kalkuleerib etteantud ArrayList’i elektrihind objektidest keskmise elektrihinna ning tagastab selle ümardatuna
* tunnidTimestampist
  + saab sisendina timestamp’i ning teisendab selle inimloetavaks kuupäevaks ja kellaajaks
* kuupäevadTimestampist
  + saab sisendina timestamp’i ning teisendab selle inimloetavaks kuupäevaks
* NulliListid
  + kustutab elektrihinna arvestamise listidest eelmise tsükli jooksul kogutud andmed
* NulliPerioodiListid
  + kustutab perioodi andmete arvestuse listidest eelmise tsükli jooksul kogutud andmed
* teeEttevalmistus ja teeEttevalmistusPeriood
  + väljundmeetodite korduvate tegevuste esilekutsumiseks

## Klass Päevahind

Kontrolleri kaudu kasutajaliidesest saadud riigi ning käesoleva hetke (kuupäev, aeg) alusel koostab päringud klassile KuvaElektriHind. Kasutab järgmiseid meetodeid:

* konstruktor info saamiseks
* getPäevaHinnad
  + teeb päringu ning annab vastuseks elektrihind tüüpi listi järgneva 24 tunni elektrihindadega
* keskmineHind, maksimaalneHind, minimaalneHind
  + teevad päringu ning annavad vastuseks elektrihind tüüpi objekti järgneva 24 tunni elektrihindade hulgast (keskmine hind on double tüüpi)
* leiaAeg
  + kalkuleerib hetke kellaaja alusel päringu alguse ja lõpu kellaaja. Elering API eripärast tulenevalt tuleb LocalDateTime ja päringu kellaaega korrigeerida 2 tunni võrra ning kuna programmi koostamise soov oli, et näitaks infot ka antud tunni kohta, mis juba on alanud, siis korrigeerisime 3 tunni võrra
* teeEttevalmistus
  + väljundmeetodite korduvate tegevuste esilekutsumiseks

## Klass PerioodiHind

Kontrolleri kaudu kasutajaliidesest saadud riigi, kuu numbri ja aasta alusel koostab päringud klassile KuvaElektriHind. Kasutab järgmiseid meetodeid:

* konstruktor info saamiseks
* getPerioodiHinnad
  + teeb päringu ning annab vastuseks ElektriHindPäev tüüpi listi järgneva määratud kuu andmetega
* getPerioodMaksimaalneHind, getPerioodMinimaalneHind, getPerioodKeskmineHind
  + teevad päringu ning annavad vastuseks elektrihind tüüpi objekti päringukuu vastava parameetri kohta (keskmine hind on double tüüpi)
* teeEttevalmistus
  + väljundmeetodite korduvate tegevuste esilekutsumiseks

## Klass Main

Graafilise keskkonna initsialiseerimine.

## Klass Controller

SceneBuilder’is konstrueeritud graafilise keskkonna (Home.fxml) ja eelnevalt kirjeldatud klasside omavahelise suhestumise organiseerimise klass. Peamised meetodid:

* päevaHindKuva
  + vasakpoolses menüüs valiku ’Järgmised 24h’ tegevuse kontrollimine
* valikuHindKuva
  + vasakpoolses menüüs valiku Valikuhind’ tegevuse kontrollimine
* lopetaProgramm
  + vasakpoolses menüüs valiku Lõpeta’ tegevuse kontrollimine, valiku kinnituse hüpikakna loomine
* kaivitaParingUuteAndmetega
  + nupu ’Käivita’ alusel uute andmete lugemise käivitamine sõltuvalt päevaHindKuva ja valikuHindKuva olekust
* salvestaCSVFail ja writeCSV
  + päringutulemuste faili salvestamine
* showPäevaHind
  + andmete päringu tulemusel saadud info jagamine õigetele elementidele
* showValitudPeriood
  + andmete päringu tulemusel saadud info jagamine õigetele elementidele

# Projekti tegemise protsessi kirjeldus

Projekt sai kirjutadud kasutades GitHub keskkonda. Kirjutamise ajal suhtlesid autorid omavahel ka konstantselt Teams’i rakenduses. Viimane oli eriti vajalik tagamaks, et erinevad kirjutatud koodid repo’s ei hakkaks üksteisega konfliki minema või ei läheks update/push protsessides kaduma. Autorid üritasid tööd ühiselt teha, kuid jagatud olid klassid siiski põhiliselt järgnevalt:

|  |  |
| --- | --- |
| Jaak Ivask | Peeter Niidas |
| Graafiline pool / Controller klass | Andmete töötlus / päringute vastused |

Suures plaanis antud jaotuse järgi sai toimitud, kuid projekti koostajate vahel toimis pidevalt konsulteerimine ning infovahetus, tagamaks eesmärgi püstitusest sünkroniseeritud arusaam. Erinevate klasside ja graafilise kasutajaliidese arendamine ja muutmine toimus vaheldumisi, kuid ühtses järjekorras (24 tunni info enne, perioodiinfo hiljem) juba valmiskirjutatud lõike kontrollisime pidevalt läbi GitHubi repo. Mitmel pool sai kasutatud ka vahetulemuste kontrollimiseks erinevate andmete ekraanile väljastamist ning eraldi täiendavate klasside meetodite kirjutamisega.

Ajakulu projekti tegemisele hindavad mõlemad autorid orienteeruvalt 30-40 astronoomilise tunni piires. Lisatud indikaatorina ka tabel GitHub’i insight kokkuvõttest:



Projektiga antud staadiumis võivad autorid üldjuhul rahul olla. Kindlast oleks võimalik koodi lühemaks ja efektiivsemaks muuta ja/või kasutajaliidesele rohkem funktsioone lisada.

# Raskused projekti koostamisel

Üheks suureks murekohaks oli vähene kogemus/oskused GitHub’i kasutamisel. Esines olukordi, kus ühe osapoole üles laetud versiooni ja teise osapoole muudatuste lisamine ei lõppenud oodatud tulemusega. Võrreldes esimese projektiga koodi kadumist õnnestus vältida. Teiseks takistuseks oli java keele vähene tundmine - suur hulk aega kulus internetist erinevate õpetuste/juhiste otsimisele ning läbitöötamisele. Oli teada mida programm peab tegema, isegi mis sammud tuleb selleks teha, kuid kuidas seda javas välja kirjutada vajas proovimist, juhendmaterjalide ja näidiste alusel katsetamist.