VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS

Objektinio programavimo Java kurso projektinė užduotis

Teksto redaktorius

Projekto aprašas

Tomas Giedraitis

Informatika, 4 kursas 3 grupė

Vilnius
2021

Turinys

Įžanga

Projektinė užduotis – sukurti teksto redaktorių Java programavimo kalba – pasidalijo į dvi dalis – redaktorius, veikiantis terminale, ir redaktorius su grafine sąsaja (Java Swing pagrindu). Pagrindinis tikslas buvo apjungti abi šias realizacijas taip, kad grafinė programa būtų silpnos sankibos (low coupling) su pagrindine programos logika, su kuria įgyvendintas terminale veikiantis redaktorius, ir tuomet vartotojas komandinės eilutės argumentu galėtų pasirinkti, kuriuo režimu jam dirbti patogiau. Tačiau prastas projektavimo sprendimas atidėti grafinės sąsajos realizavimą į projekto pabaigą nulėmė, kad užduotis apjungti parašytą programos logiką su grafine sąsaja pasidarė per sudėtinga, palyginus su grafinės sąsajos įgyvendinimu Java Swing duotomis priemonėmis kaip atskiro projekto, kurio pagrindinė dalis – klasė, savyje turinti `JFrame` objektą ir vidines klases vartotojo veiksmams (įvykiams) apdoroti. Taigi šis projektas pasidalino į dvi dalis, bet programinis kodas yra apjungtas į vieną bendrą paketą `txedt`, ir vartotojas, kaip ir numatyta, gali pasirinkti, kuriuo režimu dirbti. Todėl šio aprašo pirmi trys punktai yra bendri abiems redaktoriams, o 4-8 punktai, susiję su konkrečia realizacija, yra atitinkamai aprašyti atskirai kiekvieno redaktoriaus kontekste.

1. Paskirtis

Suteikti vartotojui pagrindines bazines teksto redagavimo ir stilizavimo funkcijas, su galimybe saugoti tekstą į failą ir skaityti iš failo. Teksto redagavimas vyksta arba terminale, arba grafinės vartotojo sąsajos pagalba.

2. Paleidimas

```
java txedt.Main {--terminal} {failoPavadinimas}
```

`--terminal:` redaktorius paleidžiamas terminale. Kitu atveju paleidžiama grafinė sąsaja. Neprivalomas argumentas.

`failoPavadinimas:` failo, kurį norima redaguoti, pavadinimas. Neprivalomas argumentas.

3. Funkcionalumas

Teksto redaktorius vartotojui suteikia šias funkcijas:

- teksto įvedimas iš klaviatūros
- teksto iškirpimas, kopijavimas, įklijavimas
- teksto stilizavimas:
    - šrifto pasirinkimas
    - teksto dydžio pasirinkimas
    - teksto lygiavimo nustatymai (kairėje, viduryje, dešinėje, užpildyti tekstu iki eilučių pabaigos)
    - paryškintas tekstas

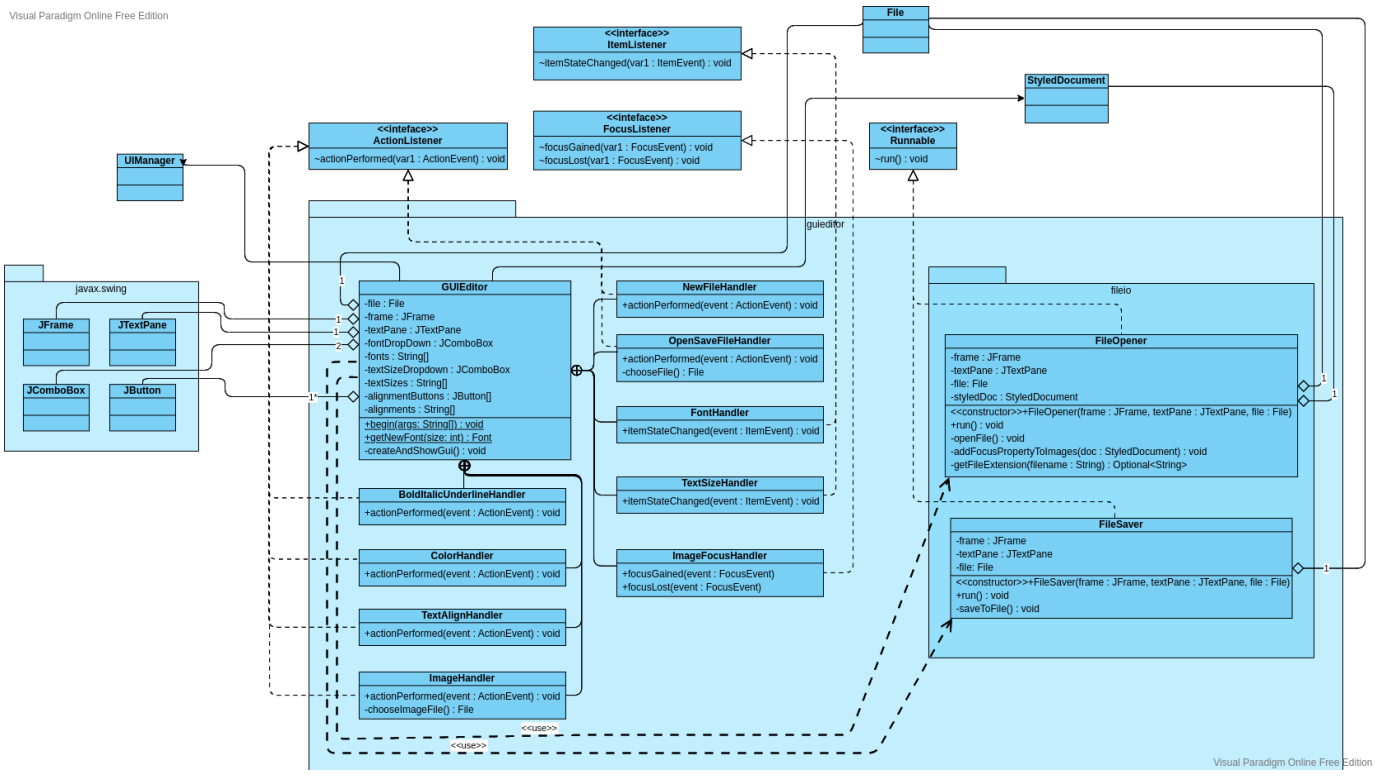
    - tekstas kursyvu
    - pabrauktas tekstas
    - teksto spalvos pasirinkimas
- paveiksliukų įkėlimas
- standartiniai Windows, Linux, Mac OS ir kitų operacinių sistemų greitieji klavišai naviguojant tekste ir redaguojant jį: *Ctrl (+ Shift) + Rodyklė į dešinę*, *Ctrl (+ Shift) + Rodyklė į kairę*, *Shift + Home*, *Shift + End*, *Ctrl + x*, *Ctrl + c*, *Ctrl + v*, ir kiti.
- mnemonikos, t.y. greitieji klavišai, prasidedantys *Alt*, meniu navigacijai.
- dokumento atidarymas iš binarinio failo.
- išsaugojimas į failą (binariniu pavidalu).

4. Redaktoriaus su grafine sąsaja realizacija

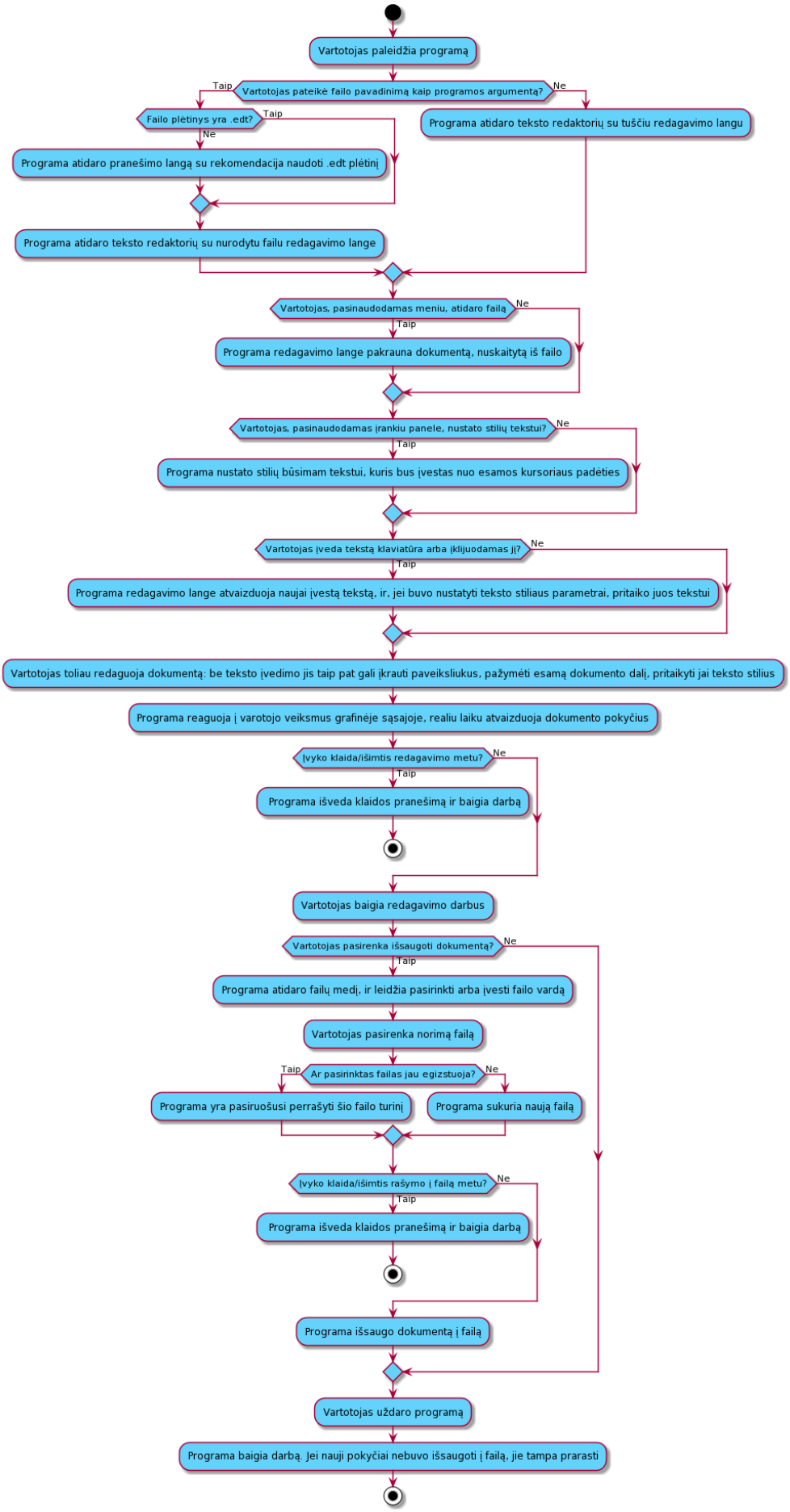
4.1. Pagrindinės klasės

Programos paleidimas vyksta iš `txedt.Main` klasės – programos pradžios taško, o visos kitos klasės yra pakete `txedt.guieditor`. Pagrindinė klasė, atsakinga už grafinės vartotojo sąsajos įgyvendinimą – klasė `GUIEditor`, kuri turi `JFrame` klasės objektą savyje bei vidines klases vartotojo veiksmams apdoroti: teksto stilizavimui, naujų paveiksliukų pridėjimui, paveiksliukų fokusavimui, taip pat naujo ar jau egzistuojančio failo atidarymui ir išsaugojimui į failą. Failų skaitymo ir rašymo operacijas atskiroje gijoje atlieka `fileio.FileOpener` ir `fileio.FileSaver` klasės.

4.2. Klasių diagrama



4.3. Veiklų diagrama



4.4. Programos klasių plėtimo galimybės

Yra galimybė toliau plėsti `GUIEditor` klasę pridedant naujas vartotojo veiksmus (įvykius) apdorojančias vidines klases. Tokiu būdu galima nesunkiai įgyvendinti *Undo*/*Redo* operacijų funkcionalumą, ir leisti atlikti *Undo*/*Redo* veiksmus tiek standartiniais *Ctrl* + *z*, *Ctrl* + *y* klavišais, tiek per meniu.

Sudėtingesnė užduotis būtų realizuoti ženklinimą (*bullet points*) ir numeravimą, nes tada atsiranda įvairios situacijos, kurias reikia tinkamai apdoroti, pvz. naujos eilutės įvedimas *Enter* klavišu ir numeracijos pratęsimas ar pernumeravimas, arba pvz. ženklo (*character*) ištrynimas *Backspace* klavišu, kuomet kursorius yra iškart po *bullet* ženklo ar numerio – tuomet reikia ištrinti ir atitinkamą ženklinimo ženklą. Šiai užduočiai atlikti reikėtų jau pasigilinti į darbą su interfeisais `javax.swing.text.StyledDocument` ir `javax.swing.text.Element`.

Taip pat, galima ir toliau plėsti teksto redaktorių, įgyvendinant ir kitas *MS Word* / *LibreOffice* / *OpenOffice* programų elementarias funkcijas. Tuomet jau vertėtų skaidyti projektą į daugiau klasių, ir galimai panaudoti MVC architektūrinį sprendimą.

4.5. Panaudoti projektavimo šablonai

Programos dalys, kuriose panaudoti žinomi projektavimo šablonai (design patterns):

- `GUIEditor` klasė skirtingose vietose naudoja kelis skirtingus grafinių elementų išdėstymus (`FlowLayout`, `BoxLayout`, `BorderLayout`), kurie įgyvendina `LayoutManager` interfeisą. Kiekviena jo realizacija turi savo algoritmus, strategiją, kaip išdėstyti elementus, ir kiekvienas iš šių išdėstymų, t.y. objektų, gali būti pakeistas kitu tą patį interfeisą realizuojančiu objektu. Taigi, tokiu būdu panaudotas *Strategy* šablonas.

- `FileOpener` ir `FileSaver` naudojami įgyvendinant *Command* šabloną, kai klasė `GUIEditor` duoda komandą išsaugoti ar atidaryti failą, ir realizacija perduodama klasėms FileOpener ir FileSaver, atskiriant nuo grafinės sąsajos realizavimo. Tačiau pilnam šio šablono pritaikymui dar reikėtų, kad `FileSaver` ir `FileOpener` klasės įgyvendintų tą patį interfeisą, pvz. `interface Command`, ir per šį interfeisą redaktorius galėtų vieningai perduoti ir kitas komandas, ne tik failo išsaugojimo ir atidarymo.

- `GUIEditor` klasėje naudojamas *Observer* šablonas, kai su Java Swing suteiktomis priemonėmis yra sekami vartotojo veiksmai, ir jiems įvykus sugeneruojamas tam tikro tipo įvykio objektas, kuris perduodamas reikiamo tipo „klausytojams“, kurie realizuoti vidinėmis `GUIEditor` klasėmis.

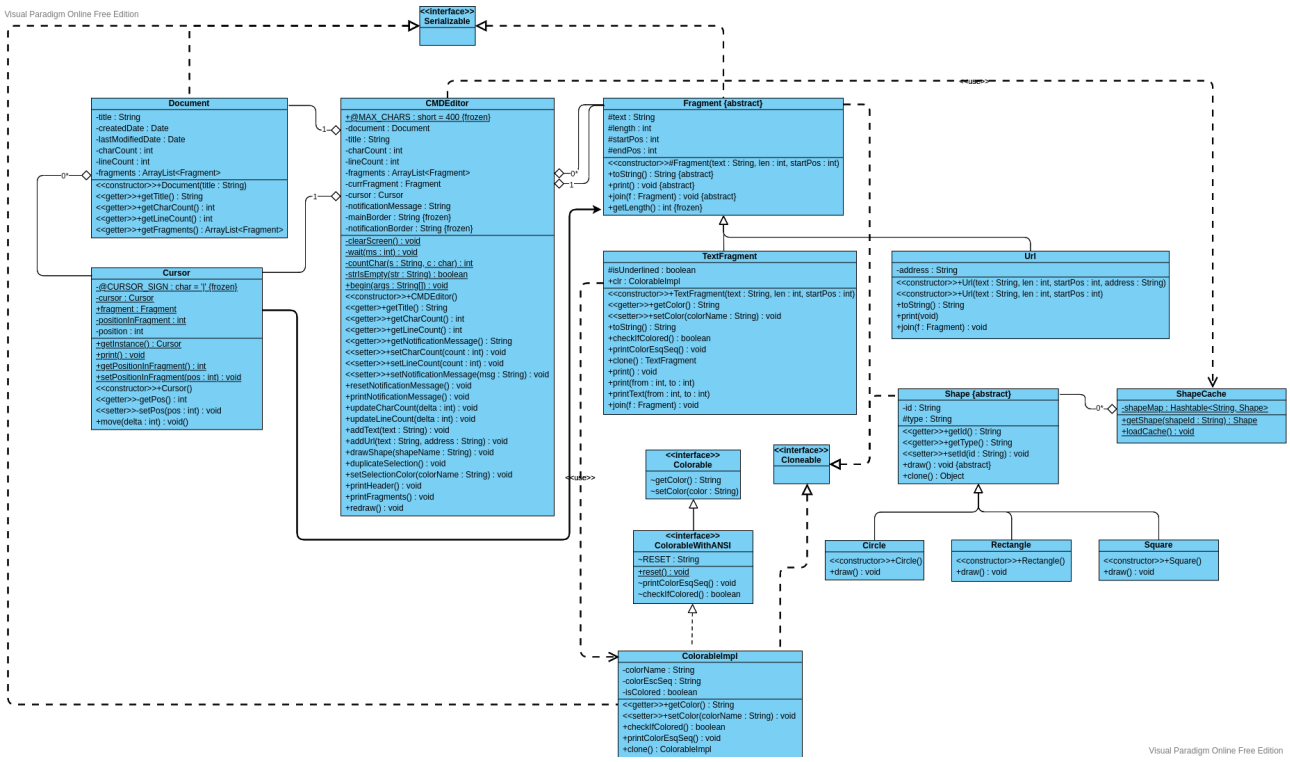
- Klasės `FileOpener` metodas `addFocusPropertyToImages` naudoja *Iterator* šabloną, perrinkdamas visus `Element` tipo objektus naujai atidarytame dokumente, paslepiant tai, kaip su šiais objektais yra dirbama `StyledDocument` klasėje.

5. Redaktoriaus terminale realizacija

5.1. Pagrindinės klasės

Programos paleidimas vyksta iš `txedt.Main` klasės – programos pradžios taško, o visos kitos klasės yra pakete `txedt.cmdeditor`. Pagrindinė klasė – `CMDEditor`. Einamojoje projekto versijoje šioje klasėse metodų kvietimais simuliuojamas teksto redagavimas, vartotojas gali pasirinkti metodus ir jų kvietimo tvarką programiniame kode. Redagavimo objektas – klasės `document.Document` egzempliorius, saugantis visą informaciją apie redaguojamą dokumentą, ir šis objektas yra skaitomas iš failo ir įrašomas į failą (binariniu pavidalu). Failų skaitymo ir rašymo operacijas atskiroje gijoje atlieka `fileio.FileOpener` ir `fileio.FileSaver` klasės. Redaguojamas dokumentas susideda iš fragmentų - abstrakčios klasės `fragment.Fragment` realizacijų - teksto fragmento, nuorodos fragmento, ir kitų. Klasė `cursor.Cursor` realizuoja vienbuvį kursoriaus objektą. Kitos projekto klasės atsakingos už teksto stilizavimą, taip pat – atskiras klasių paketas `exceptions` yra skirtas klaidoms, atsirandančiomis programos vykdymo eigoje, apdoroti.

5.2. Klasių diagrama



Pastaba: Siekiant, kad ši diagrama būtų lengviau skaitoma, joje neatvaizduotos `fileio.FileSaver` ir `fileio.FileOpener` klasės, kurios atrodo kaip grafinės sąsajos projekto diagramoje 4.2., tik neturi grafinių elementų savyje (`JFrame`, `JTextPane`). Taip pat neatvaizduotos programos logikai lengviau išreikšti sukurtos išimčių klasės – tai trivialiai įgyvendinta `exceptions.TextEditorException` bazinė išimties klasė su tuščiu kūnu, ir iš jos paveldinčios konkretesnės išimtys.

5.3. Veiklų diagrama

Terminale veikiančio redaktoriaus veiklų diagrama yra panaši į 4.3. punkte esančią grafinio redaktoriaus veiklų diagramą, tik su ribotu funkcionalumu šiuo metu. Vėlesnėse versijose redaktoriaus funkcionalumas turėtų atitikti 4.3. veiklų diagramą, tik vietoj grafinio lango būtų terminalo langas, vietoj meniu ir įrankių panelės – greitieji klavišai ir galimai vartotojo įvestos tekstinės komandos.

5.4. Programos klasių plėtimo galimybės

- Atskira klasė vartotojo įvesčiai įgyvendinti, kuri galėtų būti ir atsakinga už klavišų apdorojimą.
- Daugiau klasės `Fragment` subklasių: klasė, piešianti lentelę, klasė, leidžianti įkelti paveiksliuką, ir kt.
- Dokumentų šablonai – t.y. `Document` objektas su tam tikra būsena. Galėtų būti įgyvendintas panaudojant prototipo projektavimo šabloną.
- Pridėjimas daugiau įvairių formų piešimui, t.y. klasės `Shape` subklasių.
- Pageidautinas apjungimas su aukščiau aprašytu redaktoriumi, turinčiu grafinę sąsają.

5.5. Panaudoti projektavimo šablonai

Programos dalys, kuriose panaudoti žinomi projektavimo šablonai (design patterns):

- Klasė `Cursor` įgyvendina *Singleton* projektavimo šabloną - vienbuvis kursoriaus objektas užtikrinimas privačiu konstruktoriumi ir viešu `getInstance()` metodu, kuris leidžia sukurti ir turėti tik vieną šios klasės objektą visoje programos veikimo eigoje.

  Klasė `ShapeCache`, savyje turėdama `Shape` abstrakčios klasės tipo nuorodas, įgyvendina du projektavimo šablonus:

- *Prototype* šabloną, kuris leidžia vietoj naujų objektų kūrimo klonuoti jau egzistuojančius objektus.

- *Factory Method* šabloną – suteikiamas metodas `getShape`, į kurį kreipiantis galima gauti skirtingą `Shape` objektą (trikampį, stačiakampį, kvadratą) priklausomai nuo perduoto argumento, ir to objekto sukūrimu jau leidžiama pasirūpinti tą objektą realizuojančiai subklasei.

Priedas Nr. 1

Projekto aplankas `TextEditor.zip` faile, kurio struktūra yra:

`GUIEditor_ClassDiagram.png` – 4.2. punke esanti UML klasių diagrama.

`GUIEditor_ActivityDiagram.png` – 4.3. punkte esanti UML veiklų diagrama.

`CMDEditor_ClassDiagram.png` – 5.2. punke esanti UML klasių diagrama.

`src/docs` – *javadoc* įrankiu sugeneruota projekto programinio kodo dokumentacija.

`src/resources` – papildomi projekto failai (paveiksliukai).

`src/txedt` – projekto programinis kodas.