OOP projekt RandomToDo Tegijad: Carolin Lüübek

Maarika Markus

Eesmärk: Luua programm, mis võimaldaks luua To-Do listi ja mis valiks sealt ära tegemiseks suvalise tegevuse, et eemaldada vajadus valida tegemiseks esimene tegevus.

Üldise töö selgitus: Programm võimaldab luua To-Do listi tegevustega, eemaldada tegevust, märkida tegevus tehtuks, kuvada list, muuta tegevuse nime ja lasta programmil valida suvaline tegevus listist.

Kasutusjuhis: Programmiga saab suhelda teksti põhiselt. Juhtnöörid:

* tegevuse lisamiseks sisesta 1 ja vajuta ENTER, siis avaneb võimalus uute tegevuste lisamiseks;
* tegevuste lisamisest väljumiseks vajuta ENTER;
* tegevuse kustutamiseks sisesta 2 ja vajuta ENTER, siis avaneb võimalus valida tegevus, mida soovitakse kustutada;
* To-Do listi kuvamiseks sisesta 3 ja vajuta ENTER;
* tegevuse tehtuks/tegemata tegevuseks märkimiseks sisesta 4 ja vajuta ENTER, siis avaneb võimalus valida tegevus, mis soovitakse tehtuks või tegemata tegevuseks märkida;
* laskmaks programmil valida suvaline tegevus tegemiseks, sisesta 5 ja vajuta ENTER;
* mingi tegevuse nime muutmiseks sisesta 6 ja vajuta ENTER, siis avaneb võimalus valida, millise tegevuse nime soovitakse muuta;

Klass Tegevus

Eesmärk: uue objekti tüübi Tegevus kirjeldus.

Olulisemad meetodid: selles klassis on kõik meetodid olulised, et ülejäänud programm saaks oma tööd teha.

Klass ToDoList

Eesmärk: programmi sisu (ToDoList-iga tehtavate tegevuste) haldamine.

Olulisemad meetodid: sisendiAnalüüs(String), mis vastavalt kasutajalt saadud sisendile kutsub välja kõiki ülejäänud meetodeid selles klassis.

Klass Peaklass

Eesmärk: Peaklassis on võimalik kogu programmi jooksutada ning seal toimub kasutajaga suhtlemine.

Olulisemad meetodid: kuvaJuhtnöörid(), mis kuvab ekraanile programmi tööle panevad käsud, ning peameetod, kus toimub kasutajaga suhtlemine ja temalt sisendi küsimine.

Protsessi kirjeldus:

Esimeseks etapiks oli välja mõelda funktsioonid, mida tahtsime oma programmile anda, ning luua klass Tegevus, et siis edasist struktuuri arendada.

Teiseks etapiks oli Peaklassi ja ToDoList klassi loomine, et programmi sisuline pool paika panna.

Edasi tegelesime meetodite loomisega ja programmi töö struktuuri paika panemisega. Pärast seda täienduste ja loogika parandamise ning ekstreemjuhtude kontrollimisega (nt kui kasutaja tahab muuta mingi tegevuse nime, aga listis pole ühtegi tegevust, saab ta vastavasisulise teate).

Carolini panus (klassid/meetodid + orienteeruv ajakulu):

Klassid: ToDoList ning Peaklassi struktuur

Meetodid: klassis ToDoList kuvaSissekanded, lisaToDoListi (põhi), kustutaTegevus, kuvaList, tegevusTehtuks, sisendiAnalüüs; Peaklassis main meetod ning meetod kuvaJuhtnöörid.

Orienteeruv ajakulu: 6h

Maarika panus (klassid/meetodid + orienteeruv ajakulu):

Klassid: Tegevus; Peaklassi skelett ja täiendamine.

Meetodid: sisendiAnalüüs (alguses skelett ja hiljem täiendused vastavalt meetodite lisamisele), muudaTegevuseNime, suvalineTegevus, lisaToDoListi, tegevusiListis, kuvaSissekanded (tegin Carolini kirjutatud koodijupist abimeetodi, et vältida koodikordusi), ülejäänud meetodite puhul tegin siin-seal täiendusi ja loogika parandusi.

Orienteeruv ajakulu kokku: 9h

Mured: Raskusi valmistas alguses struktuuri väljamõtlemine ja paikapanemine, milline lähenemine on meie idee jaoks mõistlik ja asjakohane.

Hinnang lõpptulemusele:

Carolin: programmi lõpptulemus on isegi parem kui ette kujutasin. Näen potentsiaali edasiseks arenguks. Koostöö oli meeldiv ning sujus, leidsime probleemidele lahendused ning leidsime mõnusa tööjaotuse.

Maarika: olen väga rahul programmi tööga, töötab nii nagu ette kujutasin. Oli tore seda kirjutada! Koostöö sujus ka hästi. Plaanime programmile edasiarendust kasutajaliidese näol.

Testimise selgitus:

Esmalt oli tähtis paika saada tegevuste listiga toimetamine. Lõpuks saime aru, et programmi heaks tööks on tarvis, et tegevuste list oleks staatiline isendimuutuja, ning kui sai ka vaikimisi konstruktoriga loodud klassi ToDoList isend, oli võimalik hakata peaklassis kõiki loodud meetodeid testima.

Teine etapp testimisel oli aru saada, kuidas teha programm kasutajale võimalikult mugavaks ning arusaadavaks. Oli vaja mõelda nii programmi visuaalsele kui ka struktuursele poolele. Näiteks, millal oleks kõige loogilisem küsida kasutajalt, kas ta soovib uuesti juhtnööre näha, või uurida, kas tekst on muudest IDE kirjetest selgesti eristatav.

Samuti kontrollisime n-ö ebaloogilisi programmi kasutamisi, et veenduda, et kasutaja oleks sunnitud programmi õigesti kasutama.