Mogodeanu Claudiu 321CC

README TEMA POO

Arhiva incarcata va contine toate fisierele corespunzatoare claselor sau interfetelor, 2 fisiere de test, unul „neoficial” (Test_neoficial) si unul „oficial” (Test), 2 fisiere de tip JSON, cate unul per fisier de test, și acest README.

Fiecare clasa sau interfata, care nu este interna, a fost creata intr-un fisier separat, iar fisierele care se termina cu “Page” corespund implementarilor Swing. Am implementat fiecare clasa din cerintele 1 si 2, iar la 3 (bonus) am abordat primele 2 bullet-uri (vedem daca mai am timp pentru urmatorul, probabil nu). Exista si unele functionalitati extra, pe care le voi detalia ulterior.

Pentru rulare se da Run doar pe unul dintre fisierele de Test (preferabil pe cel intitulat “Test” si atat), celalalt va fi comentat, acolo lucrand cu testele facute de mine si testand putin diferit.

Timp de implementare: probabil undeva la 30h (lucrand cam 4 ore pe seara si contorizand totul)

Implementare + descriere functionalitati speciale:

-> **Catalog**: implementeaza sablonul Singleton cu constructor private. Vom considera ca un student are nota trecuta doar la una dintre rubrici (partial sau examen) daca la cealalta are 0.0. Notificarile din notifyObservers sunt personalizate in functie de cat de mult din nota are un student. Se verifica si existenta parintilor acestuia.

-> **User**: By default toate tipurile de date ce mostenesc User au ca toString toString-ul acestuia, exceptand Student, unde se afiseaza si parintii.

-> **Teacher si Assistant**: pe baza unui ScoreVisitor se apeleaza metoda visit din acesta

-> **Parent**: Am adaugat un ArrayList de Notificari in care dam add de fiecare data cand se primeste ceva nou. Se afiseaza si in consola in acel moment.

-> **Grade**: La calcularea totalului (metoda getTotal) am formatat nota astfel incat sa fie doar cu 2 zecimale, motivul fiind unele erori provocate de java de la adunarea elementelor de tip Double. Cum am mentionat anterior, o nota de 0.0 la o categorie inseamna ca nu s-a trecut nota inca in acea categorie. Pentru implementarea “clone” am urmat indicatiile scrise in metoda clone din Object: “By convention, the returned object should be obtained by calling super.clone”.

-> **Group**: in momentul in care nu este primit un comparator, acesta se va initializa cu null

-> **Course**:

- Daca, de exemplu, se creeaza o grupa noua cu un asistent care deja exista in cursul respectiv, acesta nu se va adauga iar in set-ul de asistenti, la fel si pentru Student

- Clasa interna Snapshot este continue un ArrayList de grades, un getter si un setter, care reseteaza ArrayListul si il umple cu clone

- makeBackup reseteaza campul Snapshot din clasa si umple ArrayListul salvat

- undo sterge toate notele din Course si le inlocuieste cu cele din Snapshot

- PartialCourse si FullCourse doar au implementata metoda abstracta de promovare

-> **ScoreVisitor**:

- fiecare metoda apare de 2 ori, o data pt teacher si o data pentru assistant

- am adaugat metodele: to_validate care ajuta la crearea unui array de note care trebuie validate de profesorul / asistentul respectiv si add_to_dict, care practic creeaza dictionarele aferente clasei

-> ***Interfete grafice***: cam toata partea grafica functioneaza dupa aceleasi principii:

- am impartit JFrame-ul dupa un GridLayout

- fiecare panel a fost creat intr-o metoda separata

- un panou putea avea o lista de tip JScrollPane si, uneori, ListSelectionListener

- sau putea fi blank, cu TextFielduri si Labeluri sau/si cate un Buton

-> In **CoursePage** campul Gr. este multifunctional, acesta putand insemna atat un grade cat si o grupa, realizandu-se o verificare dupa spatiere si punctuatie.