Proiect - Etapa 2 - Santa Claus is coming to ACS students

- **Data publicarii:** 03.01.2022
- **Data ultimei modificari:** 06.01.2022
- **Deadline hard:** 23.01.2022
- **Responsabili:** Ionela Nicuță, Hermina-Maria Matei, Maria Leoveanu, Mihail Ungureanu, Narcis-Florin Căroi, Andreea Oltean, Ioana-Alina Tudorache, Miruna Banu, Radu Pavel, Alexandra-Andreea Imbrișcă, Ștefan-Alin Pahonțu, Lucian-Florin Grigore, Andreea Tătulescu, Yasemin Menan
- **Actualizări**:
  - 07.01.2022, 13:11 - update niceScore - enunț, teste si checker
  - 07.01.2022, 18:20 - little fix in checker (modificare refs)
  - 07.01.2022, 19:43 - update schelet & refs
  - 08.01.2022, 00:49 - update ref23
  - 10.01.2022, 12:08 - update tests(15, 28, 29, 30) and refs(28, 29)
  - 10.01.2022, 12:30 - solve checkstyle errors

Obiective

- dezvoltarea unor abilități de bază de organizare și design orientat-obiect
- scrierea de cod pornind de la un alt cod existent, prin adăugarea de noi funcționalități
- crearea unui design decuplat și ușor citibil
- respectarea unui stil de codare și de comentare

Scenariu

Cu siguranță ai făcut o treabă minunată la prima etapă, iar Crăciunul e aproape salvat.

Fiindcă Moșul e bătrân, a uitat de niște informații importante. De aceea, el te roagă să îl ajuți din nou, punându-ți la dispoziție și elfii lui.

Context

Din cauza pandemiei, Moșul a avut mai puține resurse ca în anii trecuți. Prin urmare, are un număr limitat de cadouri. Uitându-se prin arhivele de la Polul Nord, a reușit să estimeze câte cadouri de fiecare fel are la dispoziție. Astfel, pentru fiecare cadou se va regăsi și câmpul **quantity** în json-ul de input. De fiecare dată când Moșul asignează un cadou unui copil, cantitatea acelui cadou scade.
Daca un anumit cadou are cantitate 0, acesta nu va mai putea fi asignat niciunui copil.

Pe ultima sută de metri, Moșul s-a gândit și la câteva detalii care i-ar fi de folos în noaptea de Crăciun.

Bonusul de cumințenie - niceScoreBonus

În json-ul de input, fiecare copil va avea un scor bonus de cumințenie opțional, care va influența calculul scorului **average**. După ce se calculează scorul average cu metoda specificată la prima etapa, la acest scor average se va adăuga bonusul de cumințenie.
averageScore-ul se va calcula folosind formula

```
averageScore += averageScore * niceScoreBonus / 100
```

- Bonusul de cumințenie se primește la fiecare copil și nu se poate modifica în schimbările anuale.
- **Se adaugă** conform formulei date mai sus **pentru fiecare rundă din simulare**
- Dacă averageScore-ul depășește valoarea 10 **dupa adăugarea bonusului de cumințenie**, atunci el **se va trunchia la 10**.
- Acesta poate avea valoare 0 (copilul respectiv nu are un scor bonus de cumințenie) și NU poate fi **null**.

Elfii Mosului - elf

Fiecare copil va avea propriul elf care se va ocupa de cadourile sale din acest an. În funcție de tipul de elf, scorul de cumințenie sau bugetul asignat unui copil poate să crească, să scadă sau să nu sufere nicio modificare, după cum urmează:

- elful **YELLOW** - le dă un cadou copiilor care nu au primit niciun alt cadou din sacul moșului astfel:
  - alege pe cel mai ieftin de la categoria preferată de copil
  - dacă nu există cadouri la categoria preferată sau dacă cel mai ieftin cadou s-a epuizat - cantiate = 0, atunci nu va asigna niciun cadou
- elful **BLACK** - scade bugetul asignat cu 30%

```
buget = buget - buget * 30 / 100
```

- elful **PINK** - crește bugetul asignat cu 30%

```
buget = buget + buget * 30 / 100
```

- elful **WHITE** - e un elf bătrân care după atâta timp nu mai are putere să contribuie la magia Crăciunului, însă nu vrea nici să devină un Grinch așa că

decide să nu modifice în niciun fel scorul de cumințenie al copilului sau bugetul acestuia

Elful asignat unui copil se poate schimba de la an la an.

Modalități de asignare a cadourilor - strategy

Fiecare schimbare anuală va conține o strategie de asignare a cadourilor:

- **În ordinea id-urilor (id)**
- **În ordinea scorurilor de cumintenie (niceScore)**
  - Moșul va asigna cadouri mai întâi copiilor mai cuminți (cu scorul average mai mare)
  - **! ATENȚIE**: Dacă 2 copii **au același scor de cumințenie** (nu contează din ce oraș sunt, poate fi același) Moșul îi ia **în ordinea crescătoare a id-urilor**.
- **În ordinea scorurilor de cumintenie a oraselor (niceScoreCity)**
  - Moșul va calcula scorul average pentru fiecare oraș în parte ca fiind media aritmetică a scorurilor average a copiilor care locuiesc în acel oraș
  - **! ATENȚIE**: Când se calculează media aritmetică pentru un oraș, copiii din acel oraș sunt sortați în funcție de id
  - Moșul va vizita prima dată cel mai cuminte oraș (orașul cu scorul average cel mai mare) și va parcurge copiii în ordinea id-urilor
  - **! ATENȚIE**: Când 2 orașe au același scor average, se vor sorta în ordine crescătoare lexicografic.

În primul an (**runda 0**) se va folosi strategia bazată pe id, deși acest lucru nu se afla in json-ul de input.

Flow-ul simulării

1. Se calculează score-ul average pentru fiecare copil și la acesta se adaugă bonusul de cumințenie
2. Se calculează bugetul pentru fiecare copil
3. Se aplică modificările pentru copiii care au elfi de tip **BLACK** și **PINK**
4. Se aplică strategia de asignare a cadourilor și copiii primesc cadouri
5. Se aplică modificările oferite de elful **YELLOW**

Format output

Formatul outputului rămâne același ca cel de la etapa 1. În cadrul outputului, copiii vor fi sortați în funcție de id pentru fiecare an al simulării.

Indicații

- Separați conceptele de sine stătătoare în clase separate, nu le îmbinați - clasele ar trebui să aibă un sigur rol
- Adaptați agregarea și moștenirea la situație, grupați pe cât posibil informația și acțiunile comune în clase generale
- Nu vă apucați să scrieți direct; alocați timp modelării și abstractizării, pentru că altfel vă puteți trezi cu o temă muncitorească, cu mult cod din care să nu înțelegeți prea multe și pe care să-l extindeți greu
- Vă recomandăm să porniți implementarea de la codul scris în prima etapă a proiectului, la care să adăugați noile funcționalități, dar se poate trimite și o implementare scrisă de la zero, care să nu păstreze codul primei părți, dar care să conțină funcționalitatea primei părți
- **Etapa a doua se poate trimite fără să fi trimis prima etapă a proiectului**, însă va fi nevoie să se implementeze și funcționalitățile primei etape pentru a putea primi punctajul total pe teste; în acest caz se va primi punctaj doar pe a doua etapă

Testarea soluției

Pentru testarea soluției, rulați funcția **main** a clasei **Test**. Aceasta va rula atât testele, cât și checkstyle-ul. Pentru rularea checkerului, aveți nevoie ca proiectul vostru să aibă încărcate bibliotecile pentru citirea fișierelor JSON. Mai multe detalii aici.

Evaluare

Punctajul constă din:

- 60p implementare - trecerea testelor
- 10p coding style (vezi checkstyle)
- 15p design și organizare (folosire design patterns)
- 10p README clar, concis, explicații axate pe design (flow, interacțiuni)
- 5p utilizare Git (minimum 3 commit-uri personale, relevante pentru flow-ul proiectului)
- Folosirea git pentru versionare va fi verificată din folderul .git pe care trebuie să îl includeți în arhiva temei.

- Punctajul se va acorda dacă ați făcut minim 3 commit-uri relevante și cu mesaj sugestiv.
- **NU** este permis să aveți repository-urile de git publice până la deadline-ul hard.

Pentru a se oferi punctaj maxim pentru implementare este necesară folosirea a **minim 4 design pattern-uri diferite** in total pentru cele 2 etape.

Pe pagina Indicații pentru teme găsiți indicații despre scrierea README-ului și depunctările generale pentru teme.

Depunctările pentru **designul și organizarea codului** se vor scădea din punctajul testelor. Dacă vor apărea depunctări specifice temei în momentul evaluării, nemenționate pe pagina cu depunctări generale, ele se vor încadra în limitele de maxim 15 pentru design, 10p pentru README. Dacă tema nu respectă cerințele, sau nu are design OOP, atunci pot apărea depunctări suplimentare.
Bonusuri: La evaluare, putem oferi bonusuri pentru design foarte bun, cod bine documentat, dar și pentru diverse elemente suplimentare alese de voi.
Se oferă punctaj bonus pentru implementarea scorului bonus de cumințenie utilizând design pattern-ul Builder.

Temele vor fi testate împotriva plagiatului. Orice tentativă de copiere va duce la **anularea punctajului** de pe parcursul semestrului şi **repetarea materiei** atât pentru sursă(e) cât şi pentru destinație(ii), fără excepție.

Checkstyle

Unul din obiectivele temei este învățarea respectării code-style-ului limbajului pe care îl folosiți. Aceste convenții (de exemplu cum numiți fișierele, clasele, variabilele, cum indentați) sunt verificate pentru temă de către tool-ul checkstyle.
Pe pagina de Recomandări cod găsiți câteva exemple de coding style.
Dacă numărul de erori depistate de checkstyle depășește 30, atunci punctele pentru coding-style nu vor fi acordate. Dacă punctajul este negativ, *acesta se trunchiază la 0*.
Exemple:

- `punctaj_total = 125` și `nr_erori = 200` ⇒ `nota_finala = 115`
- `punctaj_total = 125` și `nr_erori = 29` ⇒ `nota_finala = 125`
- `punctaj total = 80` și `nr erori = 30` ⇒ `nota finala = 80`
- `punctaj_total = 80` și `nr_erori = 31` ⇒ `nota_finala = 70`

Upload temă

Arhiva pe care o veţi urca pe VMChecker va trebui să conţină în directorul rădăcină:

- fișierul `README`
- folder-ul src cu pachetele și cu fișierele .java

Resurse și linkuri utile

- Schelet de cod

- Indicații pentru teme
- Recomandări coding style & javadoc