Java Oefeningen – Reeks 2

Vraag 1

In deze oefening gaan we gebruik maken van twee bestaande klassen: **ArrayList** en **Random**. Maak de klasse **Randomlijsten**, en los de maak de twee volgende methodes aan:

1. `ArrayList<Integer> throw_dice(int len).` Gegeven een lengte **len** als invoer, moet een lijst worden teruggegeven die bestaat uit 'len' willekeurige integers tussen 1 en 6.
2. `ArrayList<Integer> permute_list(int len).` Gegeven een lengte **len** als invoer, moet een lijst worden teruggegeven van alle getallen tussen 1 en **len** in een willekeurige volgorde. Elk getal komt dus precies éénmaal voor.

Vraag 2

Voor deze opgave gaan we een Juliafractaal tekenen. Hiervoor gaan we gebruik maken van de meegegeven klassen **ComplexNumber** (stelt een complex getal voor) en **SimpleImage** (voor het tekenen van afbeeldingen). Maak zelf een derde klasse **JuliaMain** aan en schrijf het volgende:

1. Maak een methode `double[] linspace(double start, double end, int nr_points)` aan. Zoals in Python, moet deze methode een array teuggeven bestaande uit **nr_points** elementen die uniform het bereik tussen **start** en **end** beslaat.
2. Teken de Julia-fractaal binnen het eenheidsvierkant in het complexe vlak ($x \in \mathbb{C}$, $|\mathrm{Re}(x)| \leq 1, |\mathrm{Im}(x)| \leq 1$). De waarde van de fractaal wordt voor elke x gegeven als:
$$n(x) = \underset{n}{\mathrm{argmin}}(|z_n| > M) \text{ met } z_{i+1}(x) = z_i(x)^2 + c,\ z_0(x) = x$$
m.a.w., vind vanaf welke n het element z_n de limiet M overschrijdt. stop met itereren wanneer $n > 255$; kies dan 255 als maximale waarde voor de afbeelding van de Julia-fractaal. Kies voor de constanten $c = -0.70176 - 0.3842i$ en $M = 5$

Vraag 3

Voor vraag 3 gaan we een klasse maken die een breuk voorstelt. Maak hiervoor twee nieuwe klassen aan: **Breuk** en **Test**. Gebruik de bestaande code op de website.

- Zorg dat de bestaande constructor een foutmelding geeft wanneer de noemer op nul gezet wordt. Zet die vervolgens dan op 1.
- Maak een nieuwe constructor aan die één **int** gebruikt. De breuk wordt dan gelijk aan die **int**.
- Schrijf een methode ***vermenigvuldig(…)*** die een breuk kan vermenigvuldigen met een ander.
- Maak vervolgens een **static** methode ***vermenigvuldig(…)*** aan binnen **Breuk** die twee **Breuk**en als input meekrijgt, en een nieuwe **Breuk** teruggeeft als returnwaarde.

- Maak een methode die de breuk omzet in een **double**.
- Tenslotte willen we ervoor zorgen dat de breuk automatisch steeds genormaliseerd blijft. Maak een **private** methode ***normalize()*** aan die dat realiseert door zowel teller als noemer door hun grootste gemeenschappelijk deler te delen. Gebruik ***normalize()*** in de andere methodes waar nodig. Vergeet niet om je methodes uit te testen.

Vraag 4

Voor deze opgave maken we een programma schrijven om een sociaal netwerk te beheren, genaamd "SocialNet". Maak hiervoor een nieuwe package, en een hoofdklasse **SocialNet** met een main().

1. Maak eerst een klasse **Gebruiker** aan; deze klasse heeft als attributen een **achternaam**, **voornaam**, **geboortejaar** en **relatiestatus.** Een relatiestatus wordt door een ***Enum*** voorgesteld en kan 4 waarden aannemen: **SINGLE**, **RELATIONSHIP**, **MARRIED** en **COMPLICATED**. Maak ook een constructor voor de klasse **Gebruiker** die een achternaam, voornaam en geboortejaar meekrijgt als inputvariabelen. Zet de initiële relatiestatus op "SINGLE".
2. Maak de methode **kennismaking()** in **Gebruiker**. Deze print het volgende uit: "Hallo, ik ben <voornaam> <naam> en ik ben geboren in <geboortejaar>."
3. Maak drie verschillende **Gebruiker**s aan in de **main()**. Gebruik deze om de verschillende geïmplementeerde methodes uit te testen.
4. We gaan een klasse **Group** aanmaken. Dit is een webpagina waar dingen gepost worden rond een bepaald thema. Een **Group** heeft als attributen een titel, beschrijving, tags (een lijst van Strings) en een **Set** van likers (**Gebruiker**s). Voorzie de volgende methodes binnen **Group**:
    a. Een constructor die een titel en een beschrijving meekrijgt.
    b. Een methode **like(Gebruiker)** die een Gebruiker toevoegt aan de "likers" Set.
    c. **count_likers()** geeft een **int** terug met het aantal likes dat het gekregen heeft.
    d. Een methode **add_tag(String)** die een tag toevoegt aan de tag-lijst.
5. Maak binnen de klasse **SocialNet** een lijst aan van **Group**s. Schrijf binnen **SocialNet** een methode **search_groups(String)** die een Lijst teruggeeft met alle titels van de **Group**s die een tag hebben die gelijk is aan de input-String van de methode.
6. **Gebruiker**s kunnen vrienden hebben in het sociale netwerk. Voorzie binnen de klasse **Gebruiker** het volgende:
    a. Twee **Set**s van **Gebruiker**s: één die Gebruikers bijhoudt van wie je een vriendschapsverzoek gekregen hebt, en één die bijhoudt wie je vrienden zijn.
    b. Een methode **verzoek_vriend** met als input een andere **Gebruiker.** De methode zorgt ervoor dat de verzochte Gebruiker die als input werd meegegeven de verzoekende Gebruiker toegevoegd krijgt aan zijn **Set** van verzoeken. Print het volgende uit: "<A> vraagt aan <B> om zijn/haar vriend te worden."

c. Maak een methode **reply_verzoek** met als input een **Gebruiker** en een **boolean** die aanduidt of het verzoek al dan niet aanvaard werd. Controleer eerst of de Gebruiker wel degelijk in de Set van verzoekers staat. Print het bericht uit dat van toepassing is:

   i. Error: <A> heeft geen vriendschapsverzoek van <B>

   ii. <A> heeft het vriendschapsverzoek van <B> aanvaardt!

   iii. <A> heeft het vriendschapsverzoek van <B> geweigerd.

   Verwijder steeds de verzoeker uit de Set van verzoekers (indien van toepassing). Als het vriendschapsverzoek aanvaard werd, voeg de Gebruikers aan elkaars Set van vrienden toe.

d. Maak tenslotte een methode **is_vriend** met als input een **Gebruiker** die een boolean teruggeeft die meegeeft of die persoon al dan niet een vriend is.

7. Test alle gemaakte methodes uit in de main() binnen **SocialNet**.