2. Po drugich zajęciach...

Zadanie 2.1 Funkcje na liczbach

Napisz funkcje:

1. int **sumaCyfr**(long n)
   Zwraca sumę wartości cyfr, z których składa się liczba n. Np. dla parametru 1023 wynikiem powinno być 6.
2. long **potega**(long podstawa, int wykladnik)
   Potęgowanie liczb całkowitych (wykonane za pomocą mnożenia w pętli).
   Ma działać dla wykladnik ≥ 0.
3. boolean **czyPierwsza**(long liczba)
   Sprawdza czy liczba jest pierwsza i zwraca **true** albo **false**.

Sprawdź czy funkcje działają prawidłowo pisząc program lub programy, w których są uruchamiane.

Zadanie 2.2 Funkcje na tablicach liczb

Każde z tych ćwiczeń należy zrealizować jako funkcję (metodę statyczną), która otrzymuje w parametrze tablicę. Przetestować możesz w jednym programie, w którym w main tworzysz kilka przykładowych tablic, a następnie uruchamiasz podane funkcje dla tych tablic i wypisujesz wyniki.

- void **wypiszPodzielne**(int[] tab, int x) – wypisuje na System.out wszystkie te liczby z tablicy tab, które są podzielne przez x (warunek do sprawdzenia: element % x == 0)
- Integer **pierwszaPodzielna**(int[] tab, int x) – zwraca (return) pierwszą znalezioną w tab liczbę podzielną przez x; zwraca null, jeśli takiej liczby tam nie ma.
- int **sumaPodzielnych**(int[] tab, int x) – liczy sumę tych elementów tablicy, które są podzielne przez x.
- int **ilePodzielnych**(int[] tab, int x) – liczy ile elementów tablicy tab jest podzielnych przez x.
- double **sredniaPodzielnych**(int[] tab, int x) – liczy średnią arytmetyczną tych elementów tablicy, które są podzielne przez x.
- Integer **max**(int[] tab) – zwraca najmniejszą wartość z tablicy
  - W przypadku pustej tablicy można zwrócić null
- Integer **min**(int[] tab) – zwraca najmniejszą wartość z tablicy
- int **roznicaMinMax**(int[] tab) – różnica pomiędzy największą a najmniejszą liczbą w tablicy; 0 jeśli tablica jest pusta.
- Integer **znajdzWspolny**(int[] t1, int[] t2) – zwraca element (liczbę), który występuje zarówno w tablicy t1, jak i t2; zwraca null, jeśli takiego elementu nie ma.
- List<Integer> **wszystkieWspolne**(int[] t1, int[] t2) – zwraca listę wszystkich wspólnych elementów z tablic t1 i t2. Jeśli takiego elementu nie ma, należy zwrócić pustą listę. (dla tych, którzy znają listy lub chcą poszukać jak się ich używa).

3. Klasy

Zadanie 3.1 Klasy dla ogłoszeń

Zaproponuj klasy, za pomocą których będzie się zapisywać ogłoszenia (takie jak w serwisie internetowym z ogłoszeniami). Użyj właściwych typów dla poszczególnych pól.

Najlepiej, aby klasa **`Ogloszenie`** opisywała rzeczy, które posiada każde ogłoszenie, m.in. tytuł, opis, cenę, dane kontaktowe sprzedawcy, a dodatkowo stwórz klasy, które rozszerzają tę klasę („podklasy") opisujące konkretne rodzaje ogłoszeń.

`OgloszenieSamochodowe` – dziedziczy z **`Ogloszenie`** i dodatkowo określa cechy sprzedawanego samochodu jak model, markę, rok produkcji, stan licznika, pojemność, moc i rodzaj paliwa (spróbuj użyć **`enum`**).

`OgloszenieMieszkaniowe` – też dziedziczy z **`Ogloszenie`**, a dodaje cechy sprzedawanego mieszkania / domu: miejscowość, metraż, liczba pokoi. Ewentualnie można utworzyć kolejne poziomy podklas: osobno dla nieruchomości różnego typu (mieszkanie / dom).

Własne pomysły mile widziane.

W używanym IDE wygeneruj standardowe konstruktory i metody. Dodaj inne metody według uznania. Napisz program, który tworzy i wypisuje kilka takich obiektów.

Zadanie 3.2 Klasa dla ułamków

Stwórz klasę, której obiekty reprezentują liczby wymierne w postaci ułamkowej. Klasa powinna implementować operacje na ułamkach.

Z kilku możliwych podejść najbardziej rekomenduję utworzenie klasy niemutowalnej, zgodnie z wzorcem „value object”: pola final, brak setterów, operacje matematyczne zawsze zwracają nowy obiekt w wyniku, a nie modyfikują bieżącego; metody porównujące biorą pod uwagę wartość, a nie tożsamość obiektu. Przykładami takich klas są BigDecimal oraz LocalDate.

Nie dopuszczaj do sytuacji, aby w mianowniku znalazło się zero – wyrzucaj wyjątek IllegalArgumentException. Zalecam taką „normalizację” tworzonych ułamków, aby mianownik był zawsze dodatni, a tylko licznik mógł być ujemny. Operacje arytmetyczne powinny także zwracać wynik w postaci maksymalnie skróconej.

Proponowane publiczne API klasy (tym razem piszę po angielsku):

- static Fraction of(long nom, long denom)
- static Fraction of(long number) // na podstawie liczby całkowitej, mianownik = 1
- static final Fraction ZERO, ONE, HALF, ... – kilka stałych, podobnie jak w BigInteger
- long getNominator(), long getDenominator()
- String toString()
- equals + hashCode – w oparciu o wartości licznika i mianownika,

  imo 3/4 i 6/8 powinny być uznane za różne, podobnie jak jest to w klasie BigDecimal, gdy wartość jest równa, ale skala (precyzja) się różni...
- int compareTo(Fraction other) – klasa powinna implementować interfejs Comparable
- Fraction shortened() – zwraca ułamek o tej samej wartości, skrócony w miarę możliwości (o największy wspólny dzielnik licznika i mianownika)
- Fraction reciprocal() – zwraca odwrotność ułamka
- Fraction neg() – zwraca liczbę ze zmienionym znakiem (plus/minus)
- Fraction add(Fraction) – suma
- Fraction sub(Fraction) – różnica
- Fraction mul(Fraction) – iloczyn
- Fraction div(Fraction) – iloraz
- double getAsDouble() – zwraca przybliżoną wartość tego ułamka jako double

Zadanie 3.3 Hulajnoga

Stwórz klasę Hulajnoga, której obiekty reprezentują hulajnogi elektryczne o różnym zasięgu.

Wpisuję tak, żeby pokazać, jak ma wyglądać użycie tej klasy. Można z tego zrobić program testujący (taki z main) albo zestaw testów jednostkowych.

```
// Tworzę hulajnogę o zasięgu 30 km
Hulajnoga1 h1 = new Hulajnoga1(30);
System.out.println(h1);
int wynik;
// Etap 1 - rozładowywanie baterii gdy jeździ się hulajnogą
// Przejeżdżam hulajnogą 20 km
// Metoda ma zwracać (return) informację o ilości przejechanych kilometrów.
// Wynikiem powinno być 20, bo tyle udało się przejechać ale "bateria powinna się częściowo rozładować".
wynik = h1.przejedz(20);
System.out.println("przejdz(20), wynik = " + wynik); // 20

// Jeśli teraz ponownie spróbuję przejechać 20 km, to wynikiem będzie już tylko 10,
// bo hulajnodze udało się przejechać tylko 10 - tylko na tyle starczyło baterii.
wynik = h1.przejedz(20);
System.out.println("przejdz(20), wynik = " + wynik); // 10
// Jeśli teraz spróbuję przejechać cokolwiek, wynikiem powinno być 0, bo bateria jest zupełnie rozładowana.
wynik = h1.przejedz(15);
System.out.println("przejdz(15), wynik = " + wynik); // 0

// Etap 2 - doładowywanie
// Metoda naladuj (bez parametrów!) powinna naładować baterię do pełna, czyli do tylu, ile było podane na początku.
System.out.println("ładowanie h1");
h1.naladuj();
// Teraz przejedz powinno pozwolić na przejechanie 30 km
wynik = h1.przejedz(100);
System.out.println("przejdz(100), wynik = " + wynik); // 30
```

// Ale każdy obiekt hulajnoga może mieć podany inny zasięg:

Hulajnoga1 h2 = new Hulajnoga1(50);

Zadanie 3.4 Aplikacje okienkowe (Swing)

Stwórz jedną lub więcej aplikacji okienkowych w technologii Swing. Wygląd interfejsu przygotuj w edytorze wizualnym, np. w NetBeans (New JFrameForm), ewentualnie w Eclipse z doinstalowaną wtyczką Window Builder (New WindowBuilder > Swing Designer > Application Window) lub w IntelliJ.

Można stworzyć aplikację według własnego pomysłu (mile widziane), albo wybrać coś poniższych propozycji.

BMI (tylko, jeśli ktoś potrzebuje bardzo prostego zadania)

Człowiek podaje swój wzrost i wagę, a otrzymuje wyliczony współczynnik BMI i informację tekstową czy jest w normie, czy ma się odchudzać, czy raczej przytyć.

Niektóre źródła na temat BMI rozróżniają normy ze względu na wiek lub płeć. Opcjonalnie możesz w swoim programie uwzględniać także te informacje.

Konwerter jednostek

Napisz program, który służy do przeliczania wartości między różnymi systemami miar. Minimum to program, który przelicza jedną parę, np. mile na kilometry. Można spróbować zrobić bardziej rozbudowaną aplikację, np. po jednej stronie ekranu umieść pola na dane w jednostkach metrycznych (centymetry, metry, kilometry, kilogramy, Celcjusze), a z drugiej brytyjskich (cale, stopy, mile, funty, Fahrenheity), a za pomocą przycisków można przeliczać w jedną lub w drugą stronę. Własne pomysły na układ okna i sposób działania mile widziane.

Automat na monety

Zrealizuj przykład z automatem parkingowym jako aplikację okienkową. Użytkownik podaje liczbę godzin, za które płaci, automat wylicza opłatę, następnie „wrzuca się monety” (np. przyciski dla różnych monet), a automat odejmuje wrzucone monety od kwoty do zapłaty, na końcu „wydaje resztę”.

Zamiast automatu parkingowego można wymyślić np. automat biletowy (z biletami normalnymi i ulgowymi), z kawą, z biletami do ZOO (normalne, ulgowe, rodzinne – z listy do wyboru) itp.

Kółko i krzyżyk

Program, który pozwala przeprowadzić rozgrywkę w kółko i krzyżyk. Można np. w oknie rozmieścić 9 przycisków, w których będą odpowiednie symbole. Gdy użytkownik kliknie w przycisk, jest to traktowane jako ruch i symbol się zmienia. Program prawidłowe kliknięcia traktuje na przemian jako ruchy jednego lub drugiego gracza. Program sam powinien rozpoznać kiedy dochodzi do wygranej lub kiedy wszystkie pola zostają zajęte i gra kończy się remisem.

To zadania można spróbować zrobić bez edytora graficznego, pisząc wszystko od zera i używając GridLayout do rozmieszczenia 9 guzików.

Wilk, koza, kapusta

Starożytna łamigłówka: Po jednej stronie rzeki znajdują się **wilk**, **koza** i **kapusta** oraz przewoźnik, który ma łodzią przewieźć je na drugą stronę rzeki. Problem polega na tym, że w łodzi zmieści się tylko jedno zwierzę/rzecz na raz – przewoźnik musi więc przewozić po jednej rzeczy i zostawiać je na brzegu. Jeśli jednak bez opieki pozostaną wilk i koza – wilk zje kozę; gdy zostaną zaś koza i kapusta – koza zje kapustę. Łamigłówka polega na tym, aby ustalić jak bezpiecznie przewieźć wszystkie trzy elementy na drugą stronę rzeki.

Zgodnie z opisanymi zasadami napisz prostą grę jako program w technologii Swing. W prostszej wersji możesz użyć wyłącznie przycisków i etykiet tekstowych. W miarę możliwości i dostępnego czasu możesz spróbować wprowadzić obrazki (poczytaj np. o ImageIcon i dodawaniu ich do JLabel/JButton).