Zadanie 1.9 Funkcje z pętlami

Napisz funkcje:

1. int **sumaNieparzystych**(int n)
   Oblicza sumę n kolejnych liczb nieparzystych zaczynając od 1. Np. dla parametru 5 wynikiem powinno być 1+3+5+7+9 = 25
2. int **ileCyfr**(long n)
   Zwraca informację z ilu cyfr składa się liczba n. Np. dla parametru 1023 wynikiem powinno być 4.
3. long **fib**(int n)
   Zwraca n-tą liczbę Fibonacciego, gdzie liczby Fibonacciego są zdefiniowane następująco:
   fib(0) = 0
   fib(1) = 1
   fin(n) = fib(n-2) + fib(n-1)
   Co oznacza, że początkowe liczby Fibonacciego to: 0 1 1 2 3 5 8 13 21 34
   Postaraj się do następnego spotkania ;) obliczyć osiemdziesiątą liczbę Fibonacciego (mieści się w long).
   Pomocniczo możesz zamiast funkcji zwracającej returnem napisać program, który wypisuje kolejne liczby na ekran. Potem wróć do napisania funkcji.
   Dla chętnych: wersja BigInteger – bez ograniczenia na wielkość wyniku.
4. 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 1.10 Operacje na tablicach liczb

Każde z tych ćwiczeń należy zrealizować jako funkcję (metodę statyczną), która otrzymuje w parametrze tablicę (int[] dla liczb całkowitych); jeśli ktoś zna albo pojawiły się już na zajęciach, można także użyć list (List<Integer>). Niektóre z metod będą wymagały podania dodatkowych parametrów.

Dla każdego z tych ćwiczeń minimum to jest implementacja samej metody („logiki") i uruchomienie dla przykładowych danych w metodzie main. **Dla chętnych** (być może tylko dla kilku wybranych zadań) można napisać program, który pobiera dane od użytkownika interaktywnie, na jeden z wcześniej poznanych sposobów, dodaje do tablicy bądź listy i wywołuje funkcję.

Pomijać te, które zrobiliśmy na zajęciach, ewentualnie jeszcze raz jako powtórzenie.

- int **suma**(int[] tab)
- double **srednia**(int[] tab) lub double srednia(double[] tab)
- int **min**(int[] tab) – zwraca najmniejszą wartość z tablicy
  - Wynikiem może być Integer i wtedy w przypadku pustej tablicy można zwrócić null
- int **roznicaMinMax**(int[] tab) – różnica pomiędzy największą a najmniejszą liczbą w tablicy; 0 jeśli tablica jest pusta.
- void **wypiszWieksze**(int[] tab, int x) – wypisuje na System.out wszystkie te liczby z tablicy tab, które są większe od x
- Integer **pierwszaWieksza**(int[] tab, int x) – zwraca (return) pierwszą znalezioną w tab liczbę większą od x; zwraca null, jeśli takiej liczby tam nie ma.
- int **sumaWiekszych**(int[] tab, int x) – zwraca (return) sumę liczb z tablicy tab, które są większe niż x.
- int **ileWiekszych**(int[] tab, int x) – liczy ile elementów tablicy tab jest większych od liczby x.
- 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 **ilePodzielnych**(int[] tab, int x) – liczy ile elementów tablicy tab jest podzielnych przez x.
- 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).

2. Aplikacje okienkowe (Swing)

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

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

Zadanie 2.1 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, Farenheity), a za pomocą przycisków można przeliczać w jedną lub w drugą stronę. Własne pomysły na układ okna i sposób działania mile widziane.

Zadanie 2.2 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.

Zadanie 2.3 BMI (jeśli ktoś potrzebuje bardzo prostego)

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.

Zadanie 2.4 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).