1. Pętle (bez tablic)

Zadanie 1.1 Choinka

Napisz program, który (np. za pomocą Scanner) wczytuje liczbę całkowitą, a następnie na konsolę wypisuje w tylu liniach „choinkę” ze znaków * . Np. dla parametru 3 powinno się wypisać:

```
    *
  * * *
* * * * *
```

Zadanie 1.2 Tabliczka mnożenia

Napisz program wypisujący na konsolę prostokątną tabliczkę mnożenia od 1 do 10*10 = 100.

Zadanie 1.3 Ulubiony język programowania

Za pomocą JOptionPane.showInputDialog pytaj użytkownika jaki jest jego ulubiony język programowania tak długo, aż wpisze odpowiedź Java.

Zadanie 1.4 Zagadka matematyczna

Napisz program sprawdzający znajomość tabliczki mnożenia.

Program losuje dwie liczby z zakresu od 1 do 10. Podaje te dwie liczby i pyta jaki jest ich iloczyn (nie podaje go). Użytkownik ma podać wynik. Program pyta o wynik wielokrotnie, tak długo, aż użytkownik poda prawidłowy wynik. Na końcu program podaje, w której próbie udało się znaleźć wynik.

Losowanie można zrealizować za pomocą klasy java.util.Random. Np. poniższy kod losuje dwie liczby z zakresu od 0 do 9 włącznie:

Random r = new Random(); int x = r.netxInt(10); int y = r.nextInt(10);

Zadanie 1.5 Zgadnij liczbę z zakresu

Program losuje liczbę z zakresu od 0 do 999 (jak wyżej). Użytkownik ma zgadnąć tę liczbę nie widząc jej. Kiedy użytkownik poda nieprawidłowy wynik, program podpowiada pisząc czy podana liczba była za duża, czy za mała. Gdy użytkownik poda właściwą liczbę, program wypisuje gratulacje jednocześnie informując, w której próbie udało się zgadnąć liczbę.

Nawiasem mówiąc technika wyszukiwania oparta o „podpowiedzi” *za dużo/za mało* nazywa się **bisekcją** i pełni w informatyce bardzo ważną rolę. Umiejętnie ją stosując powinno się te zagadki rozwiązywać w 9-10 próbach (bo $2^{10} = 1024$).

2. Pętle i tablice, czyli „pisanie algorytmów”

Każde z tych ćwiczeń należy zrealizować jako 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.

Zadanie 2.1 Operacje na tablicach liczb

(pomijając te, które zrobiliśmy na zajęciach, ewentualnie jeszcze raz jako powtórzenie)

- int **suma**(int[] tab)
- double **srednia**(int[] tab) lub double srednia(double[] tab)
- int **max**(int[] tab) – zwraca największą 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 **ileWiekszych**(int[] tab, int x) – liczy ile elementów tablicy tab jest większych od liczby 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).

3. Aplikacje okienkowe (Swing)

Stwórz jedną lub więcej aplikacji okienkowych w technologii Swing. Wygląd interfejsu przygotuj w edytorze wizualnym, np. w Netbeans (New JFrameForm), ewentualnie można spróbować w IntelliJ.

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

Zadanie 3.1 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 3.2 BMI

Człowiek podaje swój wzrost i wagę, a otrzymuje wyliczony współczynnik BMI i informację tekstową czy jest w normie, czy ma się odchudzać, czy raczej „lepiej się odżywiać" ;).

Zadanie 3.3 Automat parkingowy

Zrealizuj przykład z automatem parkingowym jako aplikację okienkową. Użytkownik podaje liczbę godzin, za które płaci, automat wylicza opłatę, następnie (jakoś wirtualnie) wrzuca się monety, 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) itd. :-)