

2. Funkcje, pętle, tablice...

Zadanie 2.1 Zgadnij liczbę z zakresu

```
Random r = new Random(); int x = r.nextInt(1000);
```

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 $2^{10} = 1024$).

Zadanie 2.2 Choinka

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

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

Podpowiedź: `System.out.print(...)` wypisuje i nie przechodzi do nowej linii, `System.out.println()` przechodzi do nowej linii.

Zadanie 2.3 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 wykładnik ≥ 0 .
3. `long silnia(int n)`
Zwraca silnię z liczby `n`, czyli iloczyn kolejnych liczb naturalnych od 1 do `n` włącznie.
Np. $\text{silnia}(5) = 1*2*3*4*5 = 120$
Przyjmuje się, że $\text{silnia}(0)$ i $\text{silnia}(1)$ są równe 1.
4. `long fib(int n)`
Zwraca `n`-tą liczbę Fibonacciego, gdzie liczby Fibonacciego są zdefiniowane następująco:
 $\text{fib}(0) = 0$
 $\text{fib}(1) = 1$
 $\text{fib}(n) = \text{fib}(n-2) + \text{fib}(n-1)$
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 (jeszcze mieści się w `long`).
Dla chętnych: wersja `BigInteger` – bez ograniczenia na wielkość wyniku.
5. `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 2.4 Funkcje na tablicach liczb

Każde z tych ćwiczeń należy zrealizować jako funkcję (metodę statyczną), która otrzymuje w parametrze tablicę (`int[]` dla liczb całkowitych). Niektóre z metod będą wymagały podania dodatkowych parametrów.

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

- 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 znaną 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 największą wartość z tablicy
 - W przypadku pustej tablicy najlepiej zwrócić null, a by to było możliwe, typen wyniku powinien być Integer zamiast int.
- 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).

Zadanie 2.5 Skarb

Napisz grę tekstową polegającą na poszukiwaniu skarbu na dwuwymiarowej planszy o rozmiarach 10 na 10. Program na początku losuje pozycję skarbu oraz pozycję gracza.

Następnie użytkownik może wprowadzać komendy zmieniające położenie postaci o jedną pozycję w górę/dół/lewo/prawo (np zgodnie z konwencją WSAD) – normalnie za pomocą Scannera.

Gdy gracz wejdzie na pole, na którym kryje się skarb – wygrywa.

Gdy wyjdzie poza planszę – przegrywa.

Po każdym ruchu użytkownik powinien otrzymywać informację o tym, czy zmierza w dobrym kierunku (zbliżasz się / oddalasz się). Po znalezieniu skarbu wypisz liczbę ruchów wykorzystanych przez użytkownika na dojście do celu.

Zadanie 2.6 Kalkulator konsolowy

Napisz program działający w konsoli, który na podstawie dwóch podanych liczb oraz znaku operacji obliczy wynik działania matematycznego. W przypadku podania nieprawidłowej operacji lub złego formatu liczb program ma wyświetlić komunikat o błędzie.

Obsłuż co najmniej cztery podstawowe działania matematyczne (+ - * /), dodatkowo możesz inne (sprawdź możliwości klasy Math).

Przykładowa sesja programu (między liczbami a znakiem mnożenia są spacje!) :

Podaj działanie: 12 * 3

Wynik: 36

Najlepiej, aby program działał w pętli i wielokrotnie pytał o działanie. Sami opracujcie sposób kończenia programu, jest kilka możliwości...

Zadanie 2.7 Range

Jedną z często używanych możliwości języka Python jest łatwe przechodzenie przez zakresy liczbowe za pomocą funkcji `range` oraz wyrażeń listowych. Jeśli masz dostęp do interpretera Pythona, spróbuj np. następujących wyrażeń:

```
list(range(10))      list(range(5, 10))      list(range(10, 20, 4)) list(range(20, 10, -3))
```

Zaimplementuj podobną funkcjonalność w Javie jako program, który przyjmuje parametry wiersza poleceń (`String[] args` w `main`) i wypisuje ciąg liczb rozdzielonych spacją na standardowe wyjście.

Program przyjmuje od 1 do 3 parametrów wiersza poleceń – liczb całkowitych. Ich ilość jest dostępna jako `args.length`, a kolejne parametry jako `args[0]`, `args[1]` i `args[2]`.

- Pierwszy parametr, o ile podano co najmniej dwa, oznacza początek zakresu. Jeśli podano tylko jeden parametr, to domyślnym początkiem zakresu jest 0.
- Drugi parametr, a jedyny w przypadku gdy podano tylko jeden, oznacza koniec zakresu, czyli liczbę, przed którą program ma zakończyć wypisywanie.
- Trzeci parametr oznacza wartość kroku, o jaki różnią się kolejne liczby ciągu. Domyślną wartością kroku jest 1. Krok może być ujemny.

Ten program możesz spróbować napisać w zwykłym edytorze (typu Notepad++, a nie Eclipse/IntelliJ). Program najłatwiej przetestować w konsoli. Przykładowe wywołania i ich wyniki (zakładają, że klasa nie należy do żadnego pakietu - jak nasze przykłady z pierwszego dnia):

```
javac Range.java
java Range 10
0 1 2 3 4 5 6 7 8 9

java Range 0

java Range 5 10
5 6 7 8 9

java Range 10 20 4
10 14 18

java Range 20 10 -3
20 17 14 11
```