W ramach zadania przyjmujemy, że wszystkie kolekcje zdefiniowane w treści przechowują wyłącznie wartości typu double. Twoje zadanie polega na zaprojektowaniu architektury obiektowej pozwalającej na łatwe przeglądanie elementów kolekcji.

ArrayList2D reprezentuje kolekcję elementów implementowaną na tablicy dwuwymiarowej. Przykładowa zawartość tablicy:

A11	A12	A13	A14
A21	A22	A23	A24
A31	A32	A33	A34

- W dalszej części przez strategie wiersz po wierszu (RC) należy rozumieć sposób przeglądania kolekcji, który zwraca kolejno elementy z wiersza. Po osiągnięciu końca wiersza, procedura powtarzana jest analogicznie dla następnego wiersza.
  Dla podanego wyżej przykładu: RC = (A11, A12, A13, A14, A21, A22, … A34)

- Przez strategie kolumna po kolumnie (CR) należy rozumieć sposób przeglądania kolekcji, który zwraca kolejno elementy z kolumny. Analogicznie do RC
  Dla podanego wyżej przykładu: CR = (A11, A21, A31, A12, A22, A32, … A34)

Kod zawiera metody statyczne inicjujące kolekcje, które używane będą do testów. W zadaniu wykorzystywane są trzy kolekcje:

- ArrayList2DRC – lista implementowana na tablicy 2D wraz z podejściem RC
- ArrayList2DCR – lista implementowana na tablicy 2D wraz z podejściem CR
- FibonacciLazyList - implementacja leniwej kolekcji zawierającej elementy ciągu Fibonacciego. Ponieważ ciąg ten jest nieograniczony na potrzeby implementacji wprowadzony jest limit zawartych wyrazów, w ramach zadania należy przyjąć, że limit ten określa koniec ciągu.

Inicjalizacja kolekcji wykorzystywanych do testów umieszczona jest w funkcji main(), w ramach rozwiązania nie można zmieniać kodu odpowiadającego za inicjalizację kolekcji testowych.

Twoje zadanie:

1. Przygotuj implementację metody get(int n), która zwracać będzie n-ty element ciągu odpowiadającą przeglądaniu RC i CR dla klas ArrayList2DRC oraz ArrayList2DCR
2. Zdefiniujemy modyfikację dla CR oraz RC, która wykonywać będzie agregacje ciągu poprzez sumowanie jego wcześniejszych elementów,
   np. dla CR:
   $$ACR(n)=\sum_{1}^{n} CR(n)$$
   $$ARC(n)=\sum_{1}^{n} RC(n)$$
   Pozwala to na określenie dwóch nowych ciągów:
   ACR = (ACR(1), ACR(2), …)
   ARC = (ARC(1), ARC(2), …)

3. Wyświetl ciąg elementów ACR oraz ARC. W ramach zadania możesz przyjąć, że ARC i ACR to domyślne strategie przeglądania kolekcji ArrayList2DRC oraz ArrayList2DCR

4. Wyświetl wszystkie elementy FibonacciLazyList
5. Wyświetl wszystkie dwuelementowe wariacje elementów zbioru składającego się z sumy zbiorów ACR, ARC oraz FibonacciLazyList
   Przykładowe wyjście:
   (...)
   3 2
   3 3
   3 5
   6 0
   6 1
   (…)
6. Wyświetl sumy powyższych elementów
   Przykładowe wyjście (dla przykładowych kombinacji z pktu 6-ego):
   (...)
   5
   6
   8
   6
   7
   (…)
7. Dla chętnych (nie podlega ocenie): FibonacciLazyList zawiera błąd w implementacji i w niektórych przypadkach zwraca niepoprawne wartości ciągu. Kiedy i jak to poprawić w możliwie najprostszy sposób?

Uwaga: w ramach rozwiązania przyjmujemy, że nigdy nie może dojść do sytuacji, że kolekcja jest modyfikowana w trakcie przeglądania.