Zadanie 1.

1. Zdefiniować trzy zadania do wykonania przez ExecutorService (jako stała pula wątków roboczych) (newFixedThreadPool). Rozmiar puli 2.
2. Każde zadanie zdefiniować w inny sposób (definicja nazwanej i anonimowej klasy implementującej **`Runnable`**, wyrażenie lambda).
3. W ramach każdego zadania wypisać na ekran 2 komunikaty (jeden na początku drugi na końcu zadania).
4. Między komunikatami uśpić zadania na losowy czas [s] U(1, 6). (użyć do tego TimeUnit.SECONDS.sleep zamiast Thread.sleep).
5. Komunikat powinien mieć strukturę:

```
Czas od uruchomienia : nazwa_tasku : wej : nazwa_wątku
```

6. Zlecić wykonanie zadań wykonawcy.
7. Zamknąć ExecutorService.
8. Zsynchronizować ich zakończenie z wątkiem głównym tak, aby po ich zakończeniu wypisać komunikat „Koniec”.

Zadanie 2.

1. Zdefiniować dwa zadania do wykonania przez ExecutorService (jako stała pula wątków roboczych) (newScheduledThreadPool). Rozmiar puli 2.
2. W ramach każdego zadania wypisać na ekran 2 komunikaty (jeden na początku drugi na końcu zadania).
3. Między komunikatami uśpić zadania na losowy czas [s] U(1, 6). (użyć do tego TimeUnit.SECONDS.sleep zamiast Thread.sleep).
4. Komunikat powinien mieć strukturę:

```
Czas od uruchomienia : nazwa_tasku : wej : nazwa_wątku
```

5. Zlecić wykonanie zadań wykonawcy:
    a. Pierwsze zadanie: rozpocząć jego wykonywanie z opóźnieniem 3s.
    b. Drugie zadanie: powtarzać jego wykonywanie co 2s.
6. Zamknąć ExecutorService po 10s.
7. Zsynchronizować ich zakończenie z wątkiem głównym tak, aby po ich zakończeniu wypisać komunikat „Koniec”.
8. Sprawdzić dodatkowo zachowanie programu, gdy po zamknięciu ExecutorService zlecimy jeszcze raz wykonanie dowolnego zadania.

Zadanie 3.

1. Zdefiniować zadanie do wykonania zwracające wynik `Callable`. (Interfejs wyniku to `Future`).
2. W ramach zadania uśpić je na 2s i zwrócić nazwę wątku wykonawcy/roboczego.
3. Zlecić wykonanie zadania ExecutorService (jako jeden wątek roboczy) (newSingleThreadExecutor).
4. W wątku głównym sprawdzać status zakończenia zadania co 0,5s.
5. Po zakończeniu zadania wypisać jego wynik.
6. Zamknąć ExecutorService.
7. Wypisać komunikat „Koniec”.

Zadanie 4.

1. Zdefiniować zadanie do wykonania zwracające wynik `Callable`. (Interfejs wyniku to `Future`).
2. W ramach zadania uśpić je na losowy czas [s] U(2, 5) i zwrócić czas wykonywania zadania [milisekundy].
3. Zlecić wykonanie zadania ExecutorService (z jednym wątkiem roboczym) (newSingleThreadExecutor).
4. W wątku głównym sprawdzać w pętli status zakończenia zadania co 0,5s i gdy czas oczekiwania przekroczy 3,5s anulować zadanie (`Future.cancel`).
5. W wątku głównym po zakończeniu zadania wypisać jego wynik, gdy było dokończone lub komunikat o anulowaniu.
6. Zamknąć ExecutorService.
7. Wypisać komunikat „Koniec”.

Zadanie 5.

1. Zdefiniować cztery zadania zwracające wynik do wykonania przez ExecutorService (jako stała pula wątków roboczych) (newFixedThreadPool). Rozmiar puli 4.
2. Każde zadanie zdefiniować na bazie interfejsu **`Callable`** .
3. W ramach każdego zadania uśpić je na losowy czas [s] U(2, 5) i zwrócić nazwę zadania.
4. Zlecić wykonanie zadań wykonawcy w taki sposób, aby odebrać wynik od pierwszego, które się zakończy.
5. Wypisać komunikat z wynikiem.
6. Zamknąć ExecutorService.
7. Wypisać komunikat „Koniec”.

Zadanie 6.

1. Zdefiniować cztery zadania zwracające wynik do wykonania przez ExecutorService (jako stała pula wątków roboczych) (newFixedThreadPool). Rozmiar puli 4.
2. Każde zadanie zdefiniować na bazie interfejsu **`Callable`** .
3. W ramach każdego zadania:
    a. uśpić je na losowy czas [s] U(2, 5),
    b. wypisać komunikat: `nazwa_tasku : nazwa_wątku : wynik`
    c. zwrócić wylosowaną wartość U(0, 10).
4. Zlecić wykonanie zadań wykonawcy w taki sposób, aby odebrać wyniki od każdego z nich w jednym wywołaniu.
5. Wypisać na ekran komunikat informujący o sumie wyników z zadań.
6. Zamknąć ExecutorService.
7. Zsynchronizować ich zakończenie z wątkiem głównym tak, aby po ich zakończeniu wypisać komunikat „Koniec”.

Zadanie 7.

1. Zdefiniować 2 zadania do wykonania przez ExecutorService (jako stała pula wątków roboczych) (newFixedThreadPool).
2. Pierwsze zadanie w nieskończonej pętli co 0,5s wypisuje komunikat: `nazwa_tasku : nazwa_wątku`
3. W ramach drugiego zadania po 4s następuje wywołanie anulujące wykonywanie pierwszego zadania (Future.cancel).
4. Zlecić wykonanie zadań wykonawcy.
5. Zamknąć ExecutorService.
6. Zsynchronizować ich zakończenie z wątkiem głównym.
7. Wypisać komunikat „Koniec”.