[GUI] Ćwiczenia VII:

```
import java.util.concurrent.*;

class ThreadTest {

    // Wątek main
    public static void main(String[] args) throws InterruptedException {

        // Jak uruchomić wątek:
        Zegar z = new Zegar("A", 5);
        z.start();                  // nigdy z.run()
        z.join();                   // czekamy, aż wątek z zakończy swoją pracę,
        System.out.println();

        /*************************************************************/

        Zegar2 z2 = new Zegar2();
        Thread watek = new Thread(z2);
        watek.start();

        Thread.sleep(5000);
        // Jak przerwać wątek
        watek.interrupt();          // nigdy watek.stop(), watek.suspend()

        // interrupt() TYLKO ustala status wątku jako przerwany
        // System.out.println(watek.isInterrupted());
        watek.join();

        System.out.println();

        /*************************************************************/

        // Jeszcze jeden wątek
        new Thread(() -> {System.out.println("Wątek: lambda\n");}).start();

        /*************************************************************/

        // import java.util.concurrent.*;
        ExecutorService wykonawca = Executors.newCachedThreadPool();
        wykonawca.execute(z2);
        wykonawca.awaitTermination(5, TimeUnit.SECONDS);
        wykonawca.shutdownNow();
    }
}
```

```
class Zegar extends Thread {

    private int czas;
```

```
    public Zegar(String nazwa, int czas) {
        super(nazwa);
        this.czas = czas;
    }

    public void run() {
        System.out.println("Zegar1");

        int i = 0;

        while (i < czas) {
            try {
                sleep(1000);    // Thread.sleep(...)
            }
            catch (InterruptedException e) {
                return;
            }
            System.out.println(++i);
        }
        System.out.println("Wątek " + this.getName() + " zakończył pracę.");
    }
}
```

```
class Zegar2 implements Runnable {

    public void run() {
        System.out.println("Zegar2");

        int czas = 0;

        while (true) {

            try {
                    //losowe opóznienie czasowe
                czas = (int)(Math.random()*2000);
                Thread.sleep(czas);
            } catch (InterruptedException e) {
                System.out.println("Przerwany wątek.");
                return; // ważne w momencie przerwania działania wątku (przez in
ny wątek)
            }
                // inna mozliwość przerwania pracy wątku
//          if (Thread.currentThread().isInterrupted())
//              return;

            System.out.println(czas + " ms");
        }
    }
}
```

```
class Buffer {

    private int[] arr;
        // konstruktor
    public Buffer(int size) {
        //...
    }
    //...

  public synchronized void put(int n){
        //...
        /*
        (pętla) dopóki bufor jest pełny
            wait();
        notify();
        */
        //...
    }
/*
        // albo inaczej, za pomocą bloku synchronizowanego
    public void put(int n){
        synchronized (this) {
            //...
            /*
            (pętla) dopóki bufor jest pełny
                wait();
            notify();
            */
            //...
        }
      }
*/
    public synchronized int get(){
        //...
        /*
        (pętla) dopóki bufor jest pusty
            wait();

        notify();
        *
        //...
    }
    // ...
}
```

```
class Producer implements Runnable {

    private Buffer buff;

        // konstruktor
```

```
    public Producer(Buffer b) {
        // ...
    }

    @Override
    public void run() {
        while (true){
            // ...
            // ...buff.put(...)...
            // usypianie wątku
            // ...
        }
    }
    // ...
}
```

```
class Consumer implements Runnable {

    private Buffer buff;
        // konstruktor
    public Consumer(Buffer b) {
        // ...
    }

    @Override
    public void run() {
        while (true){
            // ...
            // ... buff.get() ...
            // usypianie wątku
            // ...
        }
    }
    // ...
}
```

```
public class Test {
    public static void main(String[] args) {

        // ...
        Buffer buffer = new Buffer(10);
        Producer producer = new Producer(buffer);
        Consumer consumer = new Consumer(buffer);
            // utworzenie i uruchomienie wątków producenta i konsumenta
      // ...
        try {
                Thread.sleep(15000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            } finally {
```

```
            // przerywanie pracy wątków producenta i konsumenta
            // ...
        }
        // ...
    }
}
```