# Assignment 1

- create MergeSort
  - e.g. for an array of `int`s
- create parallel MergeSort
  - use threads
  - the number of threads is set as a program parameter

# Assignment 2

- create the class RoundBuffer
  - implements a round buffer with a given capacity (set in the constructor), to which Strings can be stored
  - has at least methods
    - void put(String msg)
    - String get()
  - the buffer can be used by several threads concurrently
  - if there is no space in the buffer resp. no item, then put() resp. get() are blocked

  - create a program in which several threads read and write from/to buffer
    - threads sleep randomly between reading/writing
      - for random values see java.util.Random)

Tests...

# Test 1

- What is printed out

```
public class Test01 {
  public static synchronized void main(String[] a) {
    Thread t = new Thread() {
      public void run() {
        pong();
      }
    };
    t.start();
    System.out.println("Ping");
  }
  static synchronized void pong() {
    System.out.println("Pong");
  }
}
```

A Ping Pong
B Pong Ping
C nothing – result in a deadlock
D throws an exception
E order of Ping and Pong can be different every time the program is run

The example based on code from *J. Bloch, N. Gafter: Java Puzzlers*

# Test 2

- What is printed out

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

    Thread.currentThread().interrupt();

    if (Thread.interrupted()) {
      System.out.println("Interrupted: " +
                                          Thread.interrupted());
    } else {
      System.out.println("Not interrupted: " +
                                          Thread.interrupted());
    }
  }
}
```

A Interrupted: true
B Not interrupted: false
C something else