Uncached:

```
Random uncached access write time:
         total 1915ms
         avg 38ms
         50 runs
Random uncached access read time:
         total 1885ms
         avg 37ms
         50 runs
Localized uncached access write time:
         total 10048ms
         avg 200ms
         50 runs
Localized uncached access read time:
         total 10023ms
         avg 200ms
         50 runs
Random uncached access write time:
         total 1090ms
         avg 21ms
         50 runs
Random uncached access read time:
         total 1091ms
         avg 21ms
         50 runs
Adversarial uncached access write time:
         total 1006ms
         avg 20ms
         50 runs
Adversarial uncached access read time:
         total 1011ms
         avg 20ms
         50 runs
```

Cached:

```
Random cached access write time:
         total 1538ms
         avg 30ms
         50 runs
Random cached access read time:
         total 1975ms
         avg 39ms
         50 runs
Localized cached access write time:
```

```
        total 0ms
        avg 0ms
        50 runs
Localized cached access read time:
        total 0ms
        avg 0ms
        50 runs
Random cached access write time:
        total 252ms
        avg 5ms
        50 runs
Random cached access read time:
        total 543ms
        avg 10ms
        50 runs
Adversarial cached access write time:
        total 832ms
        avg 16ms
        50 runs
Adversarial cached access read time:
        total 1157ms
        avg 23ms
        50 runs
```

The second chance finder attempts to look for a page that is still available/hasn't been touched yet. If this is not found, then the second chance algorithm according to the Notes page is done:
- an initial scan of the blocks is done, checking for totally untouched pages
- if there is nothing found, then the next step is to look for blocks not accessed recently, but modified (also, reference bit is reset for the next step if needed)
- if there is still nothing, repeat step 1 (the reference bit has been reset already, so there should be more opportunities to find a block)
- do step 2 again

Any access in Cache.java will enable the reference bit, and any modifications will enable the dirty bit, until a write to disk is done, at which point the dirty bit is reset to false.

Set the variable "DEBUG" to true in Cache.java for more detailed/verbose logging.

Writes look for a cache block and write it into the block first (and sets that page dirty) – changes are technically not "committed" yet for cached writes.

Reads look for a cached block and copy from the cached block first. If this is not found, then the page is read from disk block (disk.java, using rawread()) and copied to cache and returned to the buffer.

Caching seems to produce incredible efficiency improvements for localized access and slight improvements on mixed accesses. Random access looks like an improvement, but needs a bit more testing to make sure, as it’s not really deterministic – truly random testing and truly random access might just coincidentally act like localized access or it might be randomly picking the absolute worst possible cases where every single request is a cache miss.