Sprawozdanie nr 5

Data ćwiczeń: 23.04.2015

Data oddania sprawozdania: 14.05.2015

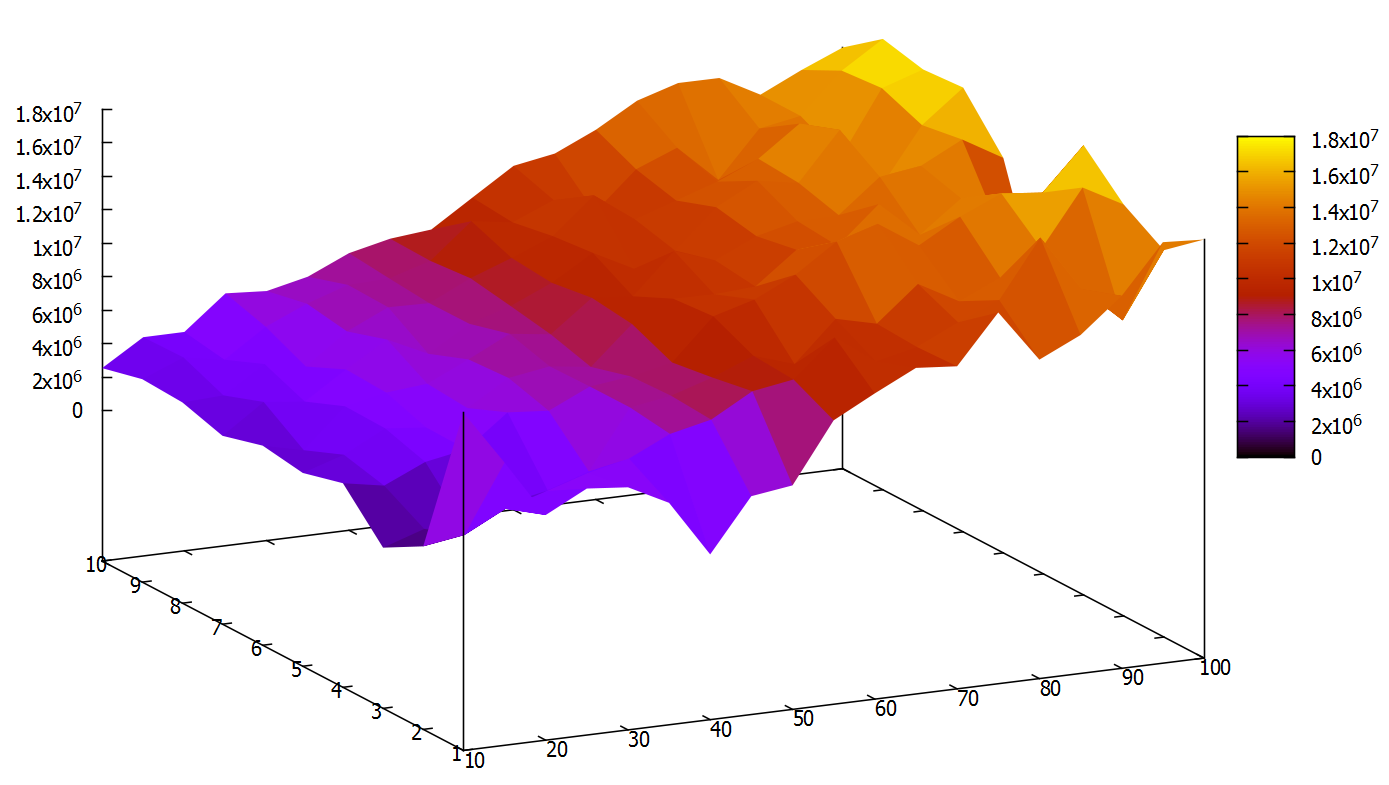
* Tematyka

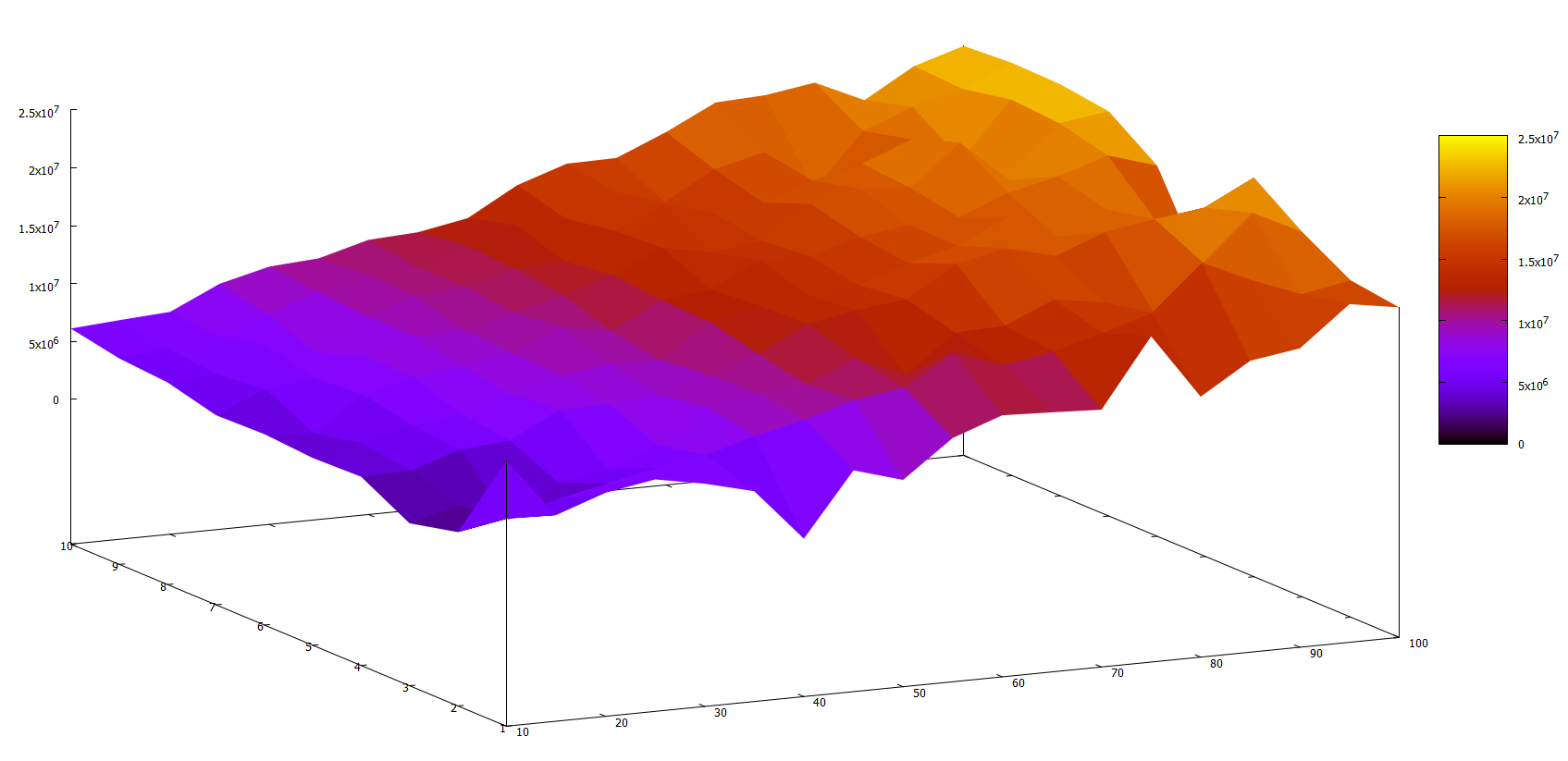
Na piątych zajęciach laboratoryjnych należało rozwiązać dwa zadania związane z synchronizacją wątków. Pojawiła się problematyka czytelników i pisarzy oraz blokowania drobnoziarnistego.

* Zadanie

Na zajęciach należało zaimplementować dwa zadania. Pierwszym z nich było rozwiązanie zadania czytelników i pisarzy oraz wykonać pomiary dla różnej ilości wątków oraz przedstawienie zależności od czasu na wykresach. Wyniki badań przedstawione zostały na wykresie Fig 2. oraz Fig 3. Drugim zadaniem było zapoznanie się z metodą blokowania drobnoziarnistego oraz zaimplentowanie klasy *List1*, która z tej metody korzystała. Następnie należało wydajność klasy *List* oraz klasy *List2*. Wyniki badań przedstawione zostały na wykresie Fig 1.

Fig 1. Wydajność listy blokowanej drobnoziarniście (List1) oraz globalnie (List2)

Fig 2. Średni czas oczekiwania jednego czytelnika (1-10 pisarze, 10-100 czytelnicy)

Fig 3. Średni czas oczekiwania jednego pisarza (1-10 pisarze, 10-100 czytelnicy)

* Kod zadania 1

*/\*\*  
 \* Created by Patryk on 2015-04-23.  
 \*/***public class** Main {  
 **private int readersAmount**;  
 **private int writersAmount**;  
 **public long readerWaitingTime** = 0L;  
 **public long writerWaitingTime** = 0L;  
  
 **public** Main(**int** readersAmount, **int** writersAmount) {  
 **this**.**readersAmount** = readersAmount;  
 **this**.**writersAmount** = writersAmount;  
 }  
  
 **public static void** main(String[] args) {  
 Main tmp;  
 **for** (**int** writersAmount = 1; writersAmount <= 10; ++writersAmount) {  
 **for** (**int** readersAmount = 10; readersAmount <= 100; readersAmount += 5) {  
 tmp = **new** Main(readersAmount, writersAmount);  
 tmp.run();  
 }  
 }  
 }  
  
 **public void** run() {  
 **int** i;  
 **final** Library library = **new** Library(**this**);  
 Reader[] reader = **new** Reader[**readersAmount**];  
 Writer[] writer = **new** Writer[**writersAmount**];  
 **for** (i = 0; i < **readersAmount**; ++i) {  
 reader[i] = **new** Reader(i, library);  
 reader[i].start();  
 }  
 **for** (i = 0; i < **writersAmount**; ++i) {  
 writer[i] = **new** Writer(i, library);  
 writer[i].start();  
 }  
 **for** (i = 0; i < **readersAmount**; ++i) {  
 **try** {  
 reader[i].join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 **for** (i = 0; i < **writersAmount**; ++i) {  
 **try** {  
 writer[i].join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 System.***out***.println(String.*format*(**"%d %d %d %d"**,  
 **readersAmount**, **writersAmount**, **readerWaitingTime**/**readersAmount**, **writerWaitingTime**/**writersAmount**));  
 }  
  
 **public long** getReaderWaitingTime() {  
 **return readerWaitingTime**;  
 }  
  
 **public void** setReaderWaitingTime(**long** readerWaitingTime) {  
 **this**.**readerWaitingTime** = readerWaitingTime;  
 }  
  
 **public long** getWriterWaitingTime() {  
 **return writerWaitingTime**;  
 }  
  
 **public void** setWriterWaitingTime(**long** writerWaitingTime) {  
 **this**.**writerWaitingTime** = writerWaitingTime;  
 }  
}

**public class** Library {  
 **private** Lock **libraryLock** = **new** ReentrantLock();  
 **private** Condition **readers** = **libraryLock**.newCondition();  
 **private** Condition **writers** = **libraryLock**.newCondition();  
 **private int isReading** = 0;  
 **private int isWriting** = 0;  
 **private int writerWaiting** = 0;  
 **private int readerWaiting** = 0;  
 **private** Main **program**;  
  
 **public** Library(Main program) {  
 **this**.**program** = program;  
 }  
  
 **public void** beginReading() {  
 **long** tmp = System.*nanoTime*();  
 **libraryLock**.lock();  
 **try** {  
 **while** (**writerWaiting** > 0 || **isWriting** > 0) {  
 ++**readerWaiting**;  
 **readers**.await();  
 }  
 **program**.setReaderWaitingTime(**program**.getReaderWaitingTime() + System.*nanoTime*() - tmp);  
 **isReading** += 1;  
 } **catch** (InterruptedException e) {  
 System.***out***.println(e.getMessage());  
 } **finally** {  
 **libraryLock**.unlock();  
 }  
 }  
  
 **public void** endReading() {  
 **long** tmp = System.*nanoTime*();  
 **libraryLock**.lock();  
 **program**.setReaderWaitingTime(**program**.getReaderWaitingTime() + System.*nanoTime*() - tmp);  
 **try** {  
 **isReading** -= 1;  
 **if** (**isReading** == 0) {  
 **if** (**writerWaiting** > 0) {  
 --**writerWaiting**;  
 }  
 **writers**.signal();  
 }  
 } **finally** {  
 **libraryLock**.unlock();  
 }  
 }  
  
 **public void** beginWriting() {  
 **long** tmp = System.*nanoTime*();  
 **libraryLock**.lock();  
 **try** {  
 **while** (**isReading** + **isWriting** > 0) {  
 ++**writerWaiting**;  
 **writers**.await();  
 }  
 **program**.setWriterWaitingTime(**program**.getWriterWaitingTime() + System.*nanoTime*() - tmp);  
 **isWriting** = 1;  
 } **catch** (InterruptedException e) {  
 System.***out***.println(e.getMessage());  
 } **finally** {  
 **libraryLock**.unlock();  
 }  
 }

**public void** endWriting() {  
 **long** tmp = System.*nanoTime*();  
 **libraryLock**.lock();  
 **program**.setWriterWaitingTime(**program**.getWriterWaitingTime() + System.*nanoTime*() - tmp);  
 **try** {  
 **isWriting** = 0;  
 **if** (**readerWaiting** == 0) {  
 **if** (**writerWaiting** > 0) {  
 --**writerWaiting**;  
 }  
 **writers**.signal();  
 } **else** {  
 **readerWaiting** = 0;  
 **readers**.signalAll();  
 }  
 } **finally** {  
 **libraryLock**.unlock();  
 }  
 }  
}

**class** Reader **extends** Thread {  
 **private int nr**;  
 **private** Library **library**;  
 **public** Reader(**int** nr, Library library) {  
 **super**();  
 **this**.**nr** = nr;  
 **this**.**library** = library;  
 }  
 @Override  
 **public void** run() {  
 **int** i = 0;  
 **while** (i++ < 1000) {  
 **library**.beginReading();  
 **library**.endReading();  
 }  
 }  
}

**class** Writer **extends** Thread {  
 **private int nr**;  
 **private** Library **library**;  
 **public** Writer(**int** nr, Library library) {  
 **super**();  
 **this**.**nr** = nr;  
 **this**.**library** = library;  
 }  
 @Override  
 **public void** run() {  
 **int** i = 0;  
 **while** (i++ < 1000) {  
 **library**.beginWriting();  
 **library**.endWriting();  
 }  
 }  
}

* Kod zadania 2

*/\*\*  
 \* Created by Patryk on 2015-05-13.  
 \*/***public class** Main **extends** Thread {  
 **private** Object[] **o**;  
 **private** List1 **list**;  
 **private** List2 **list2**;  
 **private static long** *sleepTime*;  
  
 **public** Main(Object[] o, List1 list) {  
 **super**();  
 **this**.**o** = o;  
 **this**.**list** = list;  
 }  
  
 **public** Main(Object[] o, List2 list2) {  
 **super**();  
 **this**.**o** = o;  
 **this**.**list2** = list2;  
 }  
  
 @Override  
 **public void** run() {  
*// for (int i = 0, n = o.length; i < 10; ++i) {  
// try {  
// list.add(o[i % n]);  
// list.contains(o[(i + 1) % n]);  
// list.remove(o[(i + 2) % n]);  
// list.contains(o[(i + 3) % n]);  
// list.add(o[(i + 4) % n]);  
// list.remove(o[(i + 5) % n]);  
// list.add(o[(i + 6) % n]);  
// list.remove(o[(i + 7) % n]);  
// list.contains(o[(i + 8) % n]);  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// }* **for** (**int** i = 0, n = **o**.**length**; i < 10; ++i) {  
 **try** {  
 **list2**.add(**o**[i % n]);  
 **list2**.contains(**o**[(i + 1) % n]);  
 **list2**.remove(**o**[(i + 2) % n]);  
 **list2**.contains(**o**[(i + 3) % n]);  
 **list2**.add(**o**[(i + 4) % n]);  
 **list2**.remove(**o**[(i + 5) % n]);  
 **list2**.add(**o**[(i + 6) % n]);  
 **list2**.remove(**o**[(i + 7) % n]);  
 **list2**.contains(**o**[(i + 8) % n]);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 **public static void** main(String[] args) {  
 Object[] o = { **"ala ma kota"**, 10L, 10.0, **"ola ma kota"**, 11L, 11.0,  
 **"ala ma kata"**, 11L, 11.0 };  
 List1 list = **new** List1(**"ala ma kota"**, **null**);  
 List2 list2 = **new** List2(**"ala ma kota"**, **null**);  
  
*// for (sleepTime = 0; sleepTime < 200; sleepTime += 20) {  
// list.setSleepTime(sleepTime);  
// long time = System.nanoTime();  
// Thread[] t = {new Main(o, list), new Main(o, list), new Main(o, list)};  
// for(int i = 0; i < t.length; ++i) {  
// t[i].start();  
// }  
// for(int i = 0; i < t.length; ++i) {  
// try {  
// t[i].join();  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }  
// }  
// time = System.nanoTime() - time;  
// System.out.println(sleepTime + " " + time);  
// }* **for** (*sleepTime* = 0; *sleepTime* < 200; *sleepTime* += 20) {  
 list2.*setSleepTime*(*sleepTime*);  
 **long** time = System.*nanoTime*();  
 Thread[] t = {**new** Main(o, list2), **new** Main(o, list2), **new** Main(o, list2)};  
 **for**(**int** i = 0; i < t.**length**; ++i) {  
 t[i].start();  
 }  
 **for**(**int** i = 0; i < t.**length**; ++i) {  
 **try** {  
 t[i].join();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 time = System.*nanoTime*() - time;  
 System.***out***.println(*sleepTime* + **" "** + time);  
 }  
 }  
}

**import** java.util.concurrent.locks.Lock;  
**import** java.util.concurrent.locks.ReentrantLock;  
  
**public class** List1 {  
 **private** Object **val**;  
 **private** List1 **next**;  
 **private** Lock **lock**;  
 **private static long** *sleepTime*;  
  
 **public** List1(Object val, List1 next) {  
 **this**.**val** = val;  
 **this**.**next** = next;  
 **lock** = **new** ReentrantLock();  
 }  
  
 **public boolean** contains(Object o) **throws** InterruptedException {  
 List1 prev = **null**, tmp = **this**;  
 **lock**.lock();  
 **try** {  
 **while** (tmp != **null**) {  
 **if** (**val** == o) {  
 Thread.*sleep*(*sleepTime* / 10);  
 **return true**;  
 }  
 prev = tmp;  
 tmp = tmp.**next**;  
 **try** {  
 **if** (tmp != **null**) {  
 tmp.**lock**.lock();  
 }  
 } **finally** {  
 prev.**lock**.unlock();  
 }  
 }  
 } **finally** {  
 **if** (tmp != **null**) {  
 tmp.**lock**.unlock();  
 }  
 }  
 **return false**;  
 }

**public boolean** remove(Object o) **throws** InterruptedException {  
 List1 prevprev = **null**, prev = **null**, tmp = **this**;  
 **lock**.lock();  
 **try** {  
 **while** (tmp != **null**) {  
 **if** (**val** == o) {  
 **if** (prev != **null**) {  
 prev.**next** = tmp.**next**;  
 tmp.**next** = **null**;  
 }  
 Thread.*sleep*(*sleepTime* / 3);  
 **return true**;  
 }  
 prevprev = prev;  
 prev = tmp;  
 tmp = tmp.**next**;  
 **try** {  
 **if** (tmp != **null**) {  
 tmp.**lock**.lock();  
 }  
 } **finally** {  
 **if** (prevprev != **null**) {  
 prevprev.**lock**.unlock();  
 }  
 }  
 }  
 } **finally** {  
 **if** (prev != prevprev) {  
 prev.**lock**.unlock();  
 }  
 **if** (tmp != **null**) {  
 tmp.**lock**.unlock();  
 }  
 }  
 **return false**;  
 }  
  
 **public boolean** add(Object o) **throws** InterruptedException {  
 **if** (o == **null**) {  
 **return false**;  
 }  
 List1 tmp = **this**, next = **this**.**next**;  
 **lock**.lock();  
 **try** {  
 **while** (next != **null**) {  
 **try** {  
 next.**lock**.lock();  
 } **finally** {  
 tmp.**lock**.unlock();  
 }  
 tmp = next;  
 next = next.**next**;  
 }  
 tmp.**next** = **new** List1(o, **null**);  
 Thread.*sleep*(*sleepTime*);  
 **return true**;  
 } **finally** {  
 tmp.**lock**.unlock();  
 **if** (next != tmp && next != **null**) {  
 next.**lock**.unlock();  
 }  
 }  
 }  
  
 **public static void** setSleepTime(**long** sleepTime) {  
 List1.*sleepTime* = sleepTime;  
 }  
}

**public class** List2 {  
 **private** Object **val**;  
 **private** List2 **next**;  
 **private static** Lock *lock* = **new** ReentrantLock();  
 **private static long** *sleepTime*;  
  
 **public** List2(Object val, List2 next) {  
 **this**.**val** = val;  
 **this**.**next** = next;  
 }  
  
 **public boolean** contains(Object o) **throws** InterruptedException {  
 List2 tmp = **this**;  
 *lock*.lock();  
 **try** {  
 **while** (tmp != **null**) {  
 **if** (**val** == o) {  
 Thread.*sleep*(*sleepTime* / 10);  
 **return true**;  
 }  
 tmp = tmp.**next**;  
 }  
 } **finally** {  
 *lock*.unlock();  
 }  
 **return false**;  
 }  
  
 **public boolean** remove(Object o) **throws** InterruptedException {  
 List2 prev = **null**, tmp = **this**;  
 *lock*.lock();  
 **try** {  
 **while** (tmp != **null**) {  
 **if** (**val** == o) {  
 **if** (prev != **null**) {  
 prev.**next** = tmp.**next**;  
 tmp.**next** = **null**;  
 }  
 Thread.*sleep*(*sleepTime* / 3);  
 **return true**;  
 }  
 prev = tmp;  
 tmp = tmp.**next**;  
 }  
 } **finally** {  
 *lock*.unlock();  
 }  
 **return false**;  
 }  
  
 **public boolean** add(Object o) **throws** InterruptedException {  
 **if** (o == **null**) {  
 **return false**;  
 }  
 List2 tmp = **this**, next = **this**.**next**;  
 *lock*.lock();  
 **try** {  
 **while** (next != **null**) {  
 tmp = next;  
 next = next.**next**;  
 }  
 tmp.**next** = **new** List2(o, **null**);  
 Thread.*sleep*(*sleepTime*);  
 **return true**;  
 } **finally** {  
 *lock*.unlock();  
 }  
 }  
  
 **public static void** setSleepTime(**long** sleepTime) {  
 List2.*sleepTime* = sleepTime;  
 }  
}