Сводка программ для выполнения вычислительных экспериментов

```
class CurrentThreadDemo {
  public static void main(String args[]) {
    Thread t = Thread.currentThread();
    t.setName("My Thread");
    System.out. println("current thread: " + t);
    try {
      for (int n = 5; n > 0; n--) {
        System.out.println(" " + n);
      Thread.sleep(1000);
      }
    }
    catch (InterruptedException e) {
        System.out.println("interrupted");
    }
  }
}
```

```
class ThreadDemo implements Runnable {
  ThreadDemo() {
    Thread ct = Thread.currentThread();
    System.out.println("currentThread: " + ct);
    Thread t = new Thread(this, "Demo Thread");
    System.out.println("Thread created: " + t);
    t.start();
    try {
      Thread.sleep(3000);
    }
    catch (InterruptedException e) {
      System.out.println("interrupted");
    }
    System.out.println("exiting main thread");
  }

  public void run() {
    try {
       for (int i = 5; i > 0; i--) {
         System.out.println("" + i);
         Thread.sleep(1000);
       }
    }
```

```
    catch (InterruptedException e) {
         System.out.println("child interrupted");
    }
    System.out.println("exiting child thread");
   }

   public static void main(String args[]) {
     new ThreadDemo();
   }

}



class Clicker implements Runnable {
    int click = 0;
    private Thread t;
    private boolean running = true;
    public Clicker(int p) {
       t = new Thread(this);
       t.setPriority(p);
    }

    public void run() {
       while (running) {
         click++;
       }
    }

    public void stop() {
        running = false;
    }

    public void start() {
       t.start();
    }
}
```

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

Thread.currentThread().setPriority(Thread.MAX_PRIORITY)
;
      Clicker hi = new Clicker(Thread.NORM_PRIORITY +
2);
      Clicker lo = new Clicker(Thread.NORM_PRIORITY -
2);
      lo.start();
      hi.start();
      try {
          Thread.sleep(30000);
      }
      catch (Exception e) {
      }
      lo.stop();
      hi.stop();
      System.out.println("lo="+lo.click + " vs.  hi=" +
hi.click);
   }
}
```

```
class Callme {
      //synchronized void call(String msg) {
      void call(String msg) {
      System.out.println("[" + msg);

      try {
         Thread.sleep(3000);
      }
      catch(Exception e)
      {}

      System.out.println("]");
   }
}

class Caller implements Runnable {
   String msg;
```

```
   Callme target;
   public Caller(Callme t, String s) {
      target = t;
      msg = s;
      new Thread(this).start();
   }

   public void run() {
      target.call(msg);
   }
}



class Synch {
   public static void main(String args[]) {
     Callme target = new Callme();
     new Caller(target, "Hello.");
     new Caller(target, "Synchronized");
     new Caller(target, "World");
   }
}




class Q {
   int n;
   synchronized int get() {
      System.out.println("Got: " + n);
      return n;
   }

   synchronized void put(int n) {
      this.n = n;
      System.out. println("Put: " + n);
   }
}

class Producer implements Runnable {
   Q q;
   Producer(Q q) {
     this.q = q;
     new Thread(this,"Producer").start();
```

```
   }

   public void run() {
     int i = 0;
     int k=0;
     while ( (k++) < 20 ) { //true) {
        q.put(i++);
     }
  }
}

class Consumer implements Runnable {
   Q q;
   Consumer(Q q) {
     this.q = q;
     new Thread(this,"Consumer").start();
   }

   public void run() {
      int k=0;
      while ( (k++) < 20 ) {  //true) {
        q.get();
     }
   }
}

class PC {
   public static void main(String args[]) {
     Q q = new Q();
     new Producer(q);
     new Consumer(q);
   }
}




class Q {
   int n;
   boolean valueSet = false;
   synchronized int get() {
      if (!valueSet)
         try { wait();
```

```
         }
         catch(InterruptedException e) {};
       System.out.println("Got: " + n);
       valueSet = false;
       notify();
       return n;
   }

   synchronized void put(int n) {
     if (valueSet)
        try { wait();
        }
        catch(InterruptedException e){};
     this.n = n;
     valueSet = true;
     System.out.println("Put: " + n);
     notify();
   }
}

//class Q {
//   int n;
//   synchronized int get() {
//      System.out.println("Got: " + n);
//      return n;
//   }
//
//   synchronized void put(int n) {
//      this.n = n;
//      System.out. println("Put: " + n);
//   }
//}

class Producer implements Runnable {
   Q q;
   Producer(Q q) {
     this.q = q;
     new Thread(this,"Producer").start();
   }

   public void run() {
     int i = 0;
     int k=0;
     while ( (k++) < 20 ) { //true) {
        q.put(i++);
```

```
      }
    }
}

class Consumer implements Runnable {
   Q q;
   Consumer(Q q) {
     this.q = q;
     new Thread(this,"Consumer").start();
   }

   public void run() {
      int k=0;
      while ( (k++) < 20 ) {  //true) {
        q.get();
     }
   }
}

class PC1 {
   public static void main(String args[]) {
     Q q = new Q();
     new Producer(q);
     new Consumer(q);
   }
}
```