```
/* The following code example is taken from the book
 * "The C++ Standard Library - A Tutorial and Reference, 2nd Edition"
 * by Nicolai M. Josuttis, Addison-Wesley, 2012
 *
 * (C) Copyright Nicolai M. Josuttis 2012.
 * Permission to copy, use, modify, sell and distribute this software
 * is granted provided this copyright notice appears in all copies.
 * This software is provided "as is" without express or implied
 * warranty, and with no claim as to its suitability for any purpose.
 */
#include <thread>
#include <chrono>
#include <random>
#include <iostream>
#include <exception>
using namespace std;

void doSomething (int num, char c)
{
    try {
        // random-number generator (use c as seed to get different sequences)
        default_random_engine dre(42*c);
        uniform_int_distribution<int> id(10,1000);
        for (int i=0; i<num; ++i) {
            this_thread::sleep_for(chrono::milliseconds(id(dre)));
            cout.put(c).flush();
            //...
        }
    }
    // make sure no exception leaves the thread and terminates the program
    catch (const exception& e) {
        cerr << "THREAD-EXCEPTION (thread "
             << this_thread::get_id() << "): " << e.what() << endl;
    }
    catch (...) {
        cerr << "THREAD-EXCEPTION (thread "
             << this_thread::get_id() << ")" << endl;
    }
}

int main()
{
    try {
      thread t1(doSomething,5,'.');  // print five dots in separate thread
      cout << "- started fg thread " << t1.get_id() << endl;

      // print other characters in other background threads
      for (int i=0; i<5; ++i) {
          thread t(doSomething,10,'a'+i); // print 10 chars in separate thread
          cout << "- detach started bg thread " << t.get_id() << endl;
          t.detach();  // detach thread into the background
      }

      cin.get();  // wait for any input (return)

      cout << "- join fg thread " << t1.get_id() << endl;
```

```
    t1.join();  // wait for t1 to finish
  }
  catch (const exception& e) {
    cerr << "EXCEPTION: " << e.what() << endl;
  }
}
```