```cpp
#include <pthread.h>
using namespace std;

//! \class  speculator
//! \brief  implements an abstract class that is the basis for instantiable child
//!         classes such as loop_speculator and critical_section_speculator.
//!         It simply encapsulates the speculator state (active or inactive) and the
//!         reference of the shared data, operations over those elements have to be
//!         implemented in the instantiable classes.
//!
//! Limitations:
//!
class _speculator
{
        protected:

                //bool value to set if the speculator is running, and mutex for a synchronized access.
                bool _is_running;
                pthread_mutex_t _is_running_mutex;

                //shared data between the sections of any speculator
                void*& _shared_data;

                //auxiliary data to reset _shared_data and possibly prevent segmentation faults.
                int _null_data;

                //scheduling option for pthreads.
                int _sched_option;

                _speculator():_shared_data((void*&)_null_data){//necessary because the compiler doesn't allow uninitialized shared data...
                        _is_running=false;
                        _null_data=-1;
                        _sched_option=SCHED_RR;
                        pthread_mutex_init (&_is_running_mutex, NULL);
                }

        public:
//              virtual void*& _get_shared_data() {};
};
```