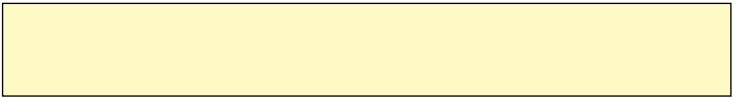

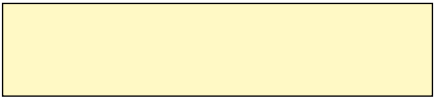

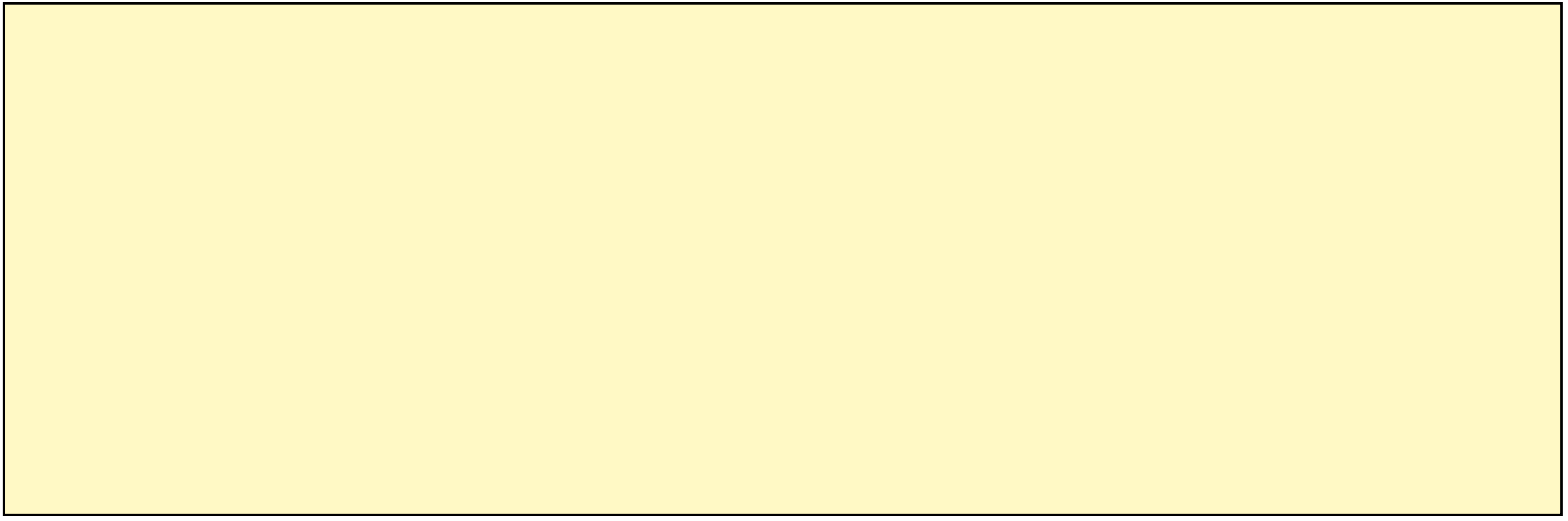

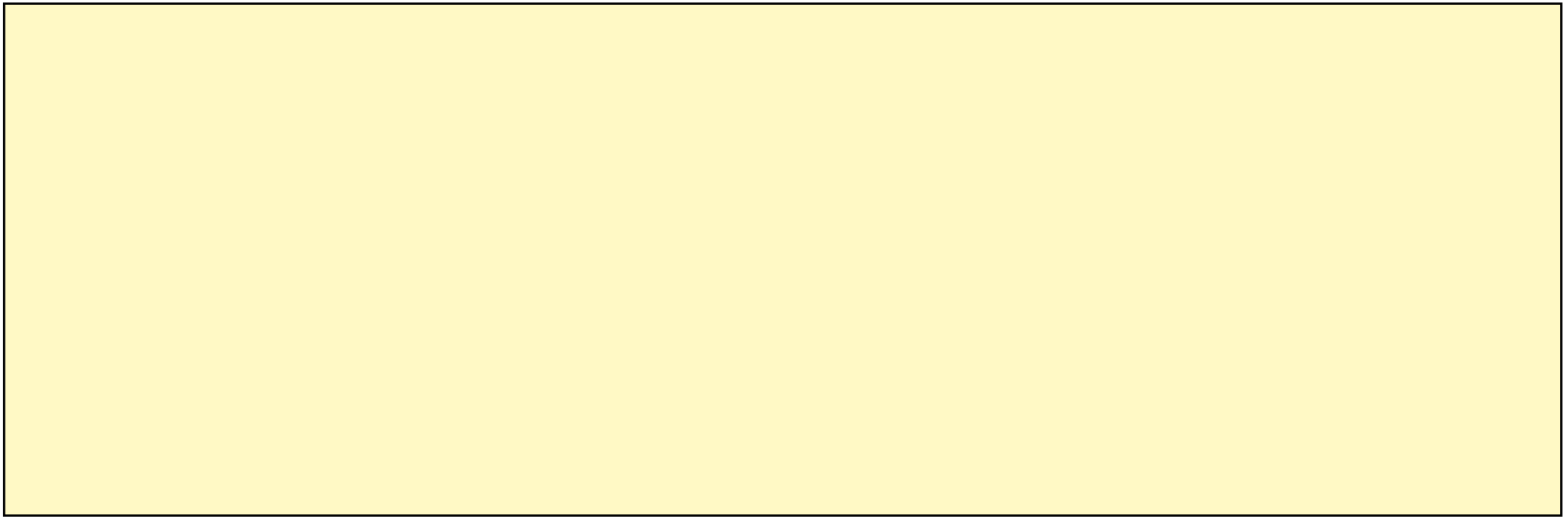

```
//==============================================================================
class PerformanceMeasurement
{
public:
    //==============================================================================
    /** Creates a PerformanceMeasurement object.

        @param counterName      the name used when printing out the statistics
        @param runsPerPrintout  the number of start/stop iterations before calling
                                printStatistics()
    */
    PerformanceMeasurement (const std::string& counterName,
                            int runsPerPrintout = 100,
                            bool printOnDestruction = true);

    /** Destructor. */
    ~PerformanceMeasurement();

    //==============================================================================
    /** Starts timing.
        @see stop
    */
    void start() noexcept;

    /** Stops timing and prints out the results.

        The number of iterations before doing a printout of the
        results is set in the constructor.

        @see start
    */
    bool stop();

    /** Dumps the current metrics to std::cout. */
    void printStatistics();


    /** Returns a copy of the current stats. */
    Statistics getStatistics() const;
```

```cpp
/** Holds the current statistics. */
struct Statistics
{
    Statistics() noexcept = default;

    void clear() noexcept;
    double getVarianceSeconds() const;
    double getVarianceCycles() const;
    std::string toString() const;

    void addResult (double secondsElapsed, uint64_t cyclesElapsed);

    std::string name;

    double meanSeconds       = 0.0;
    double m2Seconds         = 0.0;
    double maximumSeconds    = 0.0;
    double minimumSeconds    = 0.0;
    double totalSeconds      = 0.0;

    double meanCycles        = 0.0;
    double m2Cycles          = 0.0;
    uint64_t maximumCycles   = 0;
    uint64_t minimumCycles   = 0;
    uint64_t totalCycles     = 0;

    int64_t numRuns = 0;
};
```

25

```
//==============================================================================
class PerformanceMeasurement
{
public:
    //==============================================================================
    /** Creates a PerformanceMeasurement object.

        @param counterName      the name used when printing out the statistics
        @param runsPerPrintout  the number of start/stop iterations before calling
                                printStatistics()
    */
    PerformanceMeasurement (const std::string& counterName,
                            int runsPerPrintout = 100,
                            bool printOnDestruction = true);

    /** Destructor. */
    ~PerformanceMeasurement();

    //==============================================================================
    /** Starts timing.
        @see stop
    */
    void start() noexcept;

    /** Stops timing and prints out the results.

        The number of iterations before doing a printout of the
        results is set in the constructor.

        @see start
    */
    bool stop();

    /** Dumps the current metrics to std::cout. */
    void printStatistics();


    /** Returns a copy of the current stats. */
    Statistics getStatistics() const;
```

```
    /** Holds the current statistics. */
    struct Statistics
    {
        Statistics() noexcept = default;

        void clear() noexcept;
        double getVarianceSeconds() const;
        double getVarianceCycles() const;
        std::string toString() const;

        void addResult (double secondsElapsed, uint64_t cyclesElapsed);

        std::string name;

        double meanSeconds      = 0.0;
        double m2Seconds        = 0.0;
        double maximumSeconds   = 0.0;
        double minimumSeconds   = 0.0;
        double totalSeconds     = 0.0;

        double meanCycles       = 0.0;
        double m2Cycles         = 0.0;
        uint64_t maximumCycles  = 0;
        uint64_t minimumCycles  = 0;
        uint64_t totalCycles    = 0;

        int64_t numRuns = 0;
    };
```