~~Empty day~~

~~Mark work day start and end time~~

~~Pass day in constructor and timespans in start/end methods~~

~~Add tomato~~

~~Save tomato start time~~

~~Throw exception if tomato start time is earlier than the day start time~~

~~Add break~~

~~Throw exception if break start time is earlier than the day start time~~

~~Create one start method for all time slots~~

~~Add setup method~~

~~Multiple tomatoes and breaks~~

~~Throw exception when two consecutive elements of the same type are added~~

~~Check start time of new item compared to current~~

~~Notice item is in progress~~

~~Starting a new item ends the one in progress~~

~~Ending day ends the item in progress~~

~~Notice overtime~~

~~Add time provider~~

~~Create application setting~~

~~Add setting for regular hours~~

~~Add setting for tomato duration~~

~~Add setting for tomato warning time~~

~~Add setting for break warning time~~

~~Add setting for break duration~~

~~Add timer~~

~~Warn when break approaches due time~~

~~Warn when tomato reaches due time~~

~~Cleanup events~~

~~Start break when tomato reaches due time~~

~~Start a new tomato while in break~~

~~Stop running timers when starting new tomato~~

~~Start timers again when starting a new tomato~~

~~Stop running timers when ending day~~

~~User can end day~~

~~Add setting for long break duration~~

~~Add long break after 4 tomatoes~~

~~User can pause day~~

~~When a break is finished day is automatically paused~~

~~User can start break earlier~~

~~Cannot resume day with a break~~

~~Implement timer functionality~~

~~Implement cancel timer functionality~~

Add stopwatch to timer class to calculate elapsed time

Reset stopwatch when new timeslot begins

Expose time elapsed of running timeslot

Save current day activity to temporary file

Add day to archive files when finished

Loading a past day will show work details

Loading a past day with no activity will show an empty day

Validation for app settings

Handle date and time unexpected changes:

* Given that date changes in future then warn user, end current date and start new one
* Given that date changes in warn and overwrite previous day if exits
* Give time changes in future end current item normally and start new item with new time
* Given time changes in past warn and try to adjust timeslots if possible and if not throw exception to the user