**General Information about the code:**

Coded with C# 4.5.2 on Visual Studio Community 2015

I did not know what it was wanted exactly so I made my own rules. The general guidelines I used are:

-Keep as much as possible of what was there already. In real life I respect legacy code;

-Respect SOLID principles, otherwise it is just drawing graffiti on a wall;

-Use MVP structure, it is clear, simple and the one I know best;

-Make tests in advance (TDD is the best when you want to make a project from scratch);

-Use patterns (I am ready to show off to get this job);

-Modularity;

About MVP:

**Presenter** **Model View**

ParentForm

IGauge

IParameterTools

ISelectorInbox

Functions:

void GaugeCalculator();

bool SetStopLoopPermission{get; set;}

bool SetPauseLoopPermission{get;set;}

decimal FirstQuantity { get; }

long counterToPrint { get; set; }

void ItemAdd(string item, decimal number);

float Getgaugespeedvalue { get; }

decimal getQuantity(Books item);

string getTitle(Books item);

string FirstTitle { get; }

void Setgaugespeedvalue(float speedvalue);

float GetcounterToPrint { get; }

void InsertItem(List<Books> List, Books item, int index);

void DeleteItem(List<Books> List, int index);

void ClearList();

void MoveDownChecker(int index, bool SelectItemBool);

void MoveUpChecker(int index, bool SelectItemBool);

IOutputMessages

Functions:

void PrintFinished(string item);

void PrintIsPaused();

void PrintIsStopped();

void QuantityNeeded();

StartPage

OutputMessages

Looger

Gauge

ParameterTools

SelectorInbox

About SOLID:

Previously all the logic was in Form1. I split the logic in single class with single focused functions (Exception; class Gauge).

Parameters are passed through constructors.

Forms are updated through functions and Events Handler.

Each class has an interface (Exception Logger).

I prepared the interface and constructor expecting to use ninject for dependency injection (you can see the DataBindingModule.cs). Later I found it was useless because the binding is already loose and there are no extension, still I am trying to complicate it more to give it a meaning.

About Tests:

They are Unit tests and aredone using Nunit3 .

I am used to TDD. I had not a clear idea at the beginning and time was limited so I made just a few. But they work well as a proof of how I work.

About Patterns:

Aside for the MVP structure I could not find a situation where a pattern was clearly needed. So at the end I implanted a classic singleton to use in the Logger class.

About Modularity.

I divided the previous Forms1.cs class into a Parent form which contains a top bar, a bottom bar and the close button only. The rest of the forms are on the StartPage which inherits from the ParentForm , the idea is that when we will add new pages we will inherit from a blank parent Page and just add the groupboxes we need.

The Classes are single focused and the functions simplified, as you can see adding new forms and functions is quite simple now.

USECASES

Usecase 1:

User wants to print a book.

User inserts title “AAAA” , Quantity 1.

Click ADD.

User choses a speed >0.

Click Start.

Result: Counter decrease. After it reaches 0 Message Box for Print “AAAA” finished appears.

Usecase 2:

User wants to print several copies of the same book.

User inserts title “AAAA” , Quantity 100.

Click ADD.

User choses a speed >0.

Click Start.

Result: Counter decrease. After it reaches 0 Message Box for Print “AAAA” finished appears.

Usecase 3:

User wants to print several books.

User inserts title “AAAA” , Quantity 100.

Click ADD.

User inserts title “BBBB” , Quantity 100.

Click ADD.

User inserts title “CCCC” , Quantity 100.

Click ADD.

User choses a speed >0.

Click Start.

Result: Counter decrease. After it reaches 0 Message Box for Print “AAAA” finished appears.

User Click Ok.

Result: Counter decrease. After it reaches 0 Message Box for Print “BBBB” finished appears.

Click Ok,

Result: Counter decrease. After it reaches 0 Message Box for Print “CCCC” finished appears.

Usecase 4:

User wants to change the order in the list of jobs:

User inserts title “AAAA” , Quantity 100.

Click ADD.

User inserts title “BBBB” , Quantity 100.

Click ADD.

User inserts title “CCCC” , Quantity 100.

Click ADD.

User scrolls Up/Down with the buttons till it reaches “BBBB” .

User clicks “Select”

“Select” button changes to “Row Selected”

User moves “BBBB” on top.

User choses a speed >0.

Click Start.

Result: Counter decrease. After it reaches 0 Message Box for Print “BBBB” finished appears.

Click Ok.

Result: Counter decrease. After it reaches 0 Message Box for Print “AAAA” finished appears.

Click Ok,

Result: Counter decrease. After it reaches 0 Message Box for Print “CCCC” finished appears.

Usecase 5:

User wants to delete one element in the list of jobs:

User inserts title “AAAA” , Quantity 100.

Click ADD.

User inserts title “BBBB” , Quantity 100.

Click ADD.

User inserts title “CCCC” , Quantity 100.

Click ADD.

User scrolls Up/Down with the buttons till it reaches “BBBB” .

User clicks “Select”.

“Select” button changes to “Row Selected”

User Clicks “Remove”

“BBBB” disappear from the list.

User choses a speed >0.

Click Start.

Result: Counter decrease. After it reaches 0 Message Box for Print “AAAA” finished appears.

Click Ok,

Result: Counter decrease. After it reaches 0 Message Box for Print “CCCC” finished appears.

Usecase 6:

User wants to Stop the process :

User inserts title “AAAA” , Quantity 100.

Click ADD.

User inserts title “BBBB” , Quantity 100.

Click ADD.

User inserts title “CCCC” , Quantity 100.

Click ADD.

User choses a speed >0.

Click Start.

Result: Counter decrease. After it reaches 0 Message Box for Print “AAAA” finished appears.

User Click Ok.

User Clicks “Stop”.

Result: Counter Stops. Button “Stop” is disabled and grays out.

Speed goes to 0.

User Clicks “Start”.

Result: process starts again.

Result: Counter decrease. After it reaches 0 Message Box for Print “BBBB” finished appears.

Click Ok,

Result: Counter decrease. After it reaches 0 Message Box for Print “CCCC” finished appears.

Usecase 7:

User wants to reset the process :

User inserts title “AAAA” , Quantity 100.

Click ADD.

User inserts title “BBBB” , Quantity 100.

Click ADD.

User inserts title “CCCC” , Quantity 100.

Click ADD.

User Clicks “Reset”.

Result: List of jobs is cleared, Processes are stopped. Speed is set to 0. Counter is set to 0.