To Do List - Design Document

Adding a new task

To add a new task I added a menu strip object along the top of the form and gave the first column the name “File” and added a tool with text “Add New Task”, where I put the code to add and set the details of a new task. The way this worked was that clicking “Add New Task” would create a new “Edit Task” object, unique to the new task called “ET1”, “ET2” etc. and would display it where the user could type in the task’s title and description and set the deadline date as well. Then when the user clicked the “Done” button, all the details that they entered would update the relevant task in the “To Do List” form. I also added 3 other tools “Edit”, “View” and “Help” to make it more closely resemble a menu strip, although only the “Help” tool as any functionality. Clicking the “Help” tool displays a help forms with useful directions on how to use the program.

Editing a task

I created an "Edit” button for each task, which allows the user to edit an existing task. To edit a task the user clicks on the “Edit” button for the chosen task and its unique edit form is displayed, containing the task’s current contents. The user can now make their desired changes and when they are finished, click on “Done” to close the form and the program will update the changes.

Limiting character length

I limited the maximum length of the title labels and description text boxes in each task by setting the maximum length of the text boxes in the edit form to 16 and 256 respectively, by using textBox_TaskTitle.MaxLength = 16; and textBox_TaskDescription.MaxLength = 256; so if those textbox can’t exceed 16 and 256 characters then the task’s descriptions can’t either. To prevent the user from altering the contents of the task description boxes direct from the “To Do List” form, I used textBox_TaskDescription1.ReadOnly = true; so the only way the user could alter the contents of the textboxes is to use the edit form.

Deadline dates, indicators/reminders and task completion

Along with the title and description the user can set a deadline for the task, which will be shown on in the task’s group box, so the user can always see when the task is due in, I also set a “DateTime” called “Today” and set it to equal the current day, using “DateTime Today = DateTime.Now.Date;”. I created a bit of code that tests if the deadline set by the user is the same as “Today”s date which, if it were, would set the colour of all the text in the group box (apart from the description) to red and a message box is displayed, informing the user that the task is due in today, otherwise it is set to blue, which signifies that the task is on-going. If the “complete” checkbox for the task is checked, then the text is set to green, signifying that the task is complete and when the task’s deadline is equal to “Today”s date, the text will not be set to red and the message box will not be shown.

Originally I used strings for setting the deadline date for each task, but this made it impossible to calculate the time left value for the task because you can’t subtract a string from another string. I originally used “string Task1Date = dateTimePicker1.Value.ToString("dd-mm-yyyy");” but I later altered my approach to “DeadlineDate1 = dateTimePicker1.Value.Date;” and declaring “DateTime DeadlineDate1;” at the top along with all my other variables, so that they could be used in other methods. Which this now allowed me to calculate the time between the “Today”s date and the task’s “Deadline”, by declaring a “TimeSpan” for each task called “TimeLeft1”, “TimeLeft2” etc.

To calculate the time left value I used “TimeLeft1 = DeadlineDate1.Date.Subtract(Today.Date);”, also to be able to show the time left value as a number of days I created 5 ints, called “TimeLeftDays1”, “TimeLeftDays2” etc. To show the user the time left I put the latter code and “MessageBox.Show("Time Remaining: " + TimeLeftDays1 + " Days");” in a method corresponding to the appropriate button. I simply repeated everything for each task, so each task had a deadline and “time left” value.

To stop the user from selecting a deadline date that is in the past I use the code “ET1.dateTimePicker1.MinDate = Today;”, which blanks out any dates that are before the current day’s date, forcing the user to choose either today’s date or a date in the future.

Object display methods

I created 5 sets of task objects (labels, textboxes, buttons etc.) so that the user can have up to 5 concurrent tasks setand I used group boxes to hold the details for each task; its title, description, deadline, “completed” checkbox and its edit and delete buttons. I set each group box’s text as its appropriate task number e.g. “Task 1”, “Task 2” etc. This made it easy to manage the visibility of all the objects related to each task, because I only had to set the visibility for the group boxes and all the objects inside would be set accordingly, using the code “groupBox_Task1.Visible = true;” etc. to set them visible or “false” to set them invisible.

Group boxes also make it very easy to position all objects for each task in a uniform manner, for example: the title label in group box 1 will have the exact same location as the title label in group box 2, because the location is relative to the group box and not the entire form. If group boxes weren’t used then in this case the horizontal value for all the title labels will be the same, but the vertical value will change depending on how far down the next task title label was and this will have to be calculated each time. Also, using group boxes allows all the objects in it to be moved at the same time, with a single click as appose to selecting all the objects individually. This doesn’t benefit the user, because they have no control over the objects in terms of positioning, however it does benefit the creator of the program, as it did for me.

I didn’t want the user to be able to resize the forms because it would leave empty space or cover up objects, so to stop this from happening, I changed the form’s “FormBorderStyle” to “Fixed Single” and I disabled the functionality of the maximise button, so it is impossible to alter the size of the forms.

Deleting and task

I created a “Delete” button for each task, which allows the task to be deleted by setting the latest task’s group box visibility to false and transferring the details from the next task in the list to the deleted task’s group box. For example if there are 3 tasks set and I delete the 2nd task, all the details from the 3rd task are transferred to the 2nd task’s group box and the 3rd task’s group box’s visibility is set to false, along with its details being set to the task’s defaults. If there was a 4th task then the details from that task would be transferred to the 3rd task and the 4th task’s group box’s visibility would be set to false and its details will be defaulted.

Edit Form

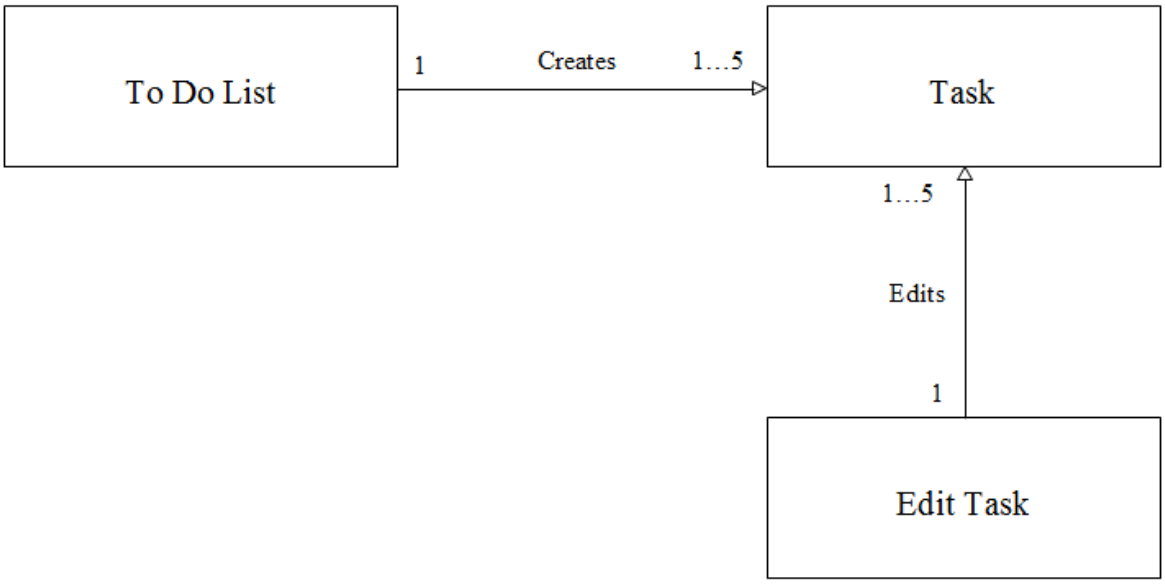
I created three buttons on the “Edit Task” form, that allowed the user to clear the textboxes with a single click, opposed to clearing them manually. Two of the buttons clear just the title or description, the last one clears both textboxes, allowing the user to decide what they want to clear, instead of being forced to either clear the whole form because of only one button or to do it manually.

Class and use – case diagrams

Task
String Task Title String Task Description String Task Deadline Bool Task Complete Bool Task Visible
Get Task Title() Set Task Title(newTitle) Get Task Description() Set Task Description(newDescription) Get Task Deadline() Set Task Deadline(newDeadline) Get Task Complete() Set Task Complete(newComplete) Get Task Visible() Set Task Visible(newVisible)

To Do List
Task Complete
Get Task Complete() Set Task Complete(newComplete) Set Task Visible(newVisible) Delete Task()

Edit Task
“Task Title” “Task Description” “Task Deadline”
Get Task Title() Set Task Title(newTitle) Get Task Description() Set Task Description(newDescription) Get Task Deadline() Set Task Deadline(newDeadline)



Collaboration diagrams