
Link for repository: https://github.com/karlowich/BDSA-Assignment3.

C#

Extension methods

- To flatten the numbers in *xs* one can simply use the line:

```
xs.SelectMany(x => x);
```

- To select numbers in *ys* which are divisible by 7 and greater than 42, the following line can be used:

```
ys.Where(x => x % 7  == 0 && x > 42);
```

- Leap years in *ys* can be selected with this line:

```
ys.Where(x => x > 1582 &&
            (x % 400 == 0  ||  !(x % 100 == 0)) &&
            x % 4 == 0);
```

Delegates / Anonymous functions

- A method which takes a string and prints the content in reverse order (by character)

  We will be using System.Linq.

```
Action<string> reverse = str =>
    Console.WriteLine(new String(str.Reverse().ToArray()));
```

- A method which takes two decimals and returns the product

```
Func<decimal, decimal, decimal> Product =
    (n1, n2) => n1 * n2;
```

- A method which takes a whole number and a string and returns true if they are numerically equal. Note that the string " 0042" should return true if the number is 42

```
Func <string, int, bool> equals = (str, n1) =>
    {
        int result;
        return int.TryParse(str.Trim(), out result)
        ? result == n1
        : false;
    };
```

Software Engineering

Exercise 1. *Research what a Kanban board is. Possibly starting from here, acquire sufficient application domain knowledge to write two as-is scenarios that represent the main usage of a physical Kanban board within a software engineering team. Note: make sure to cite your references in the submission document.*

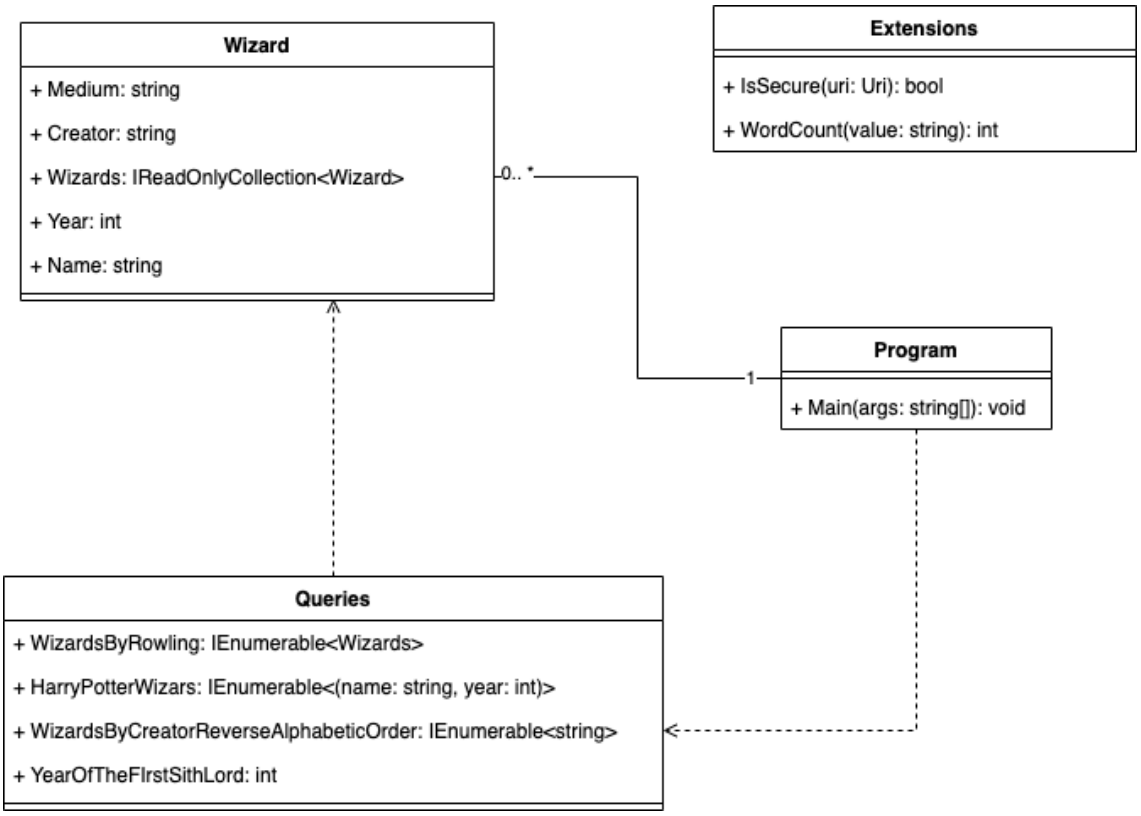
- Add work item [1]
- Move work item from 'in progress' to 'done' [2]

Scenario	Add work item
Participating actor instances	bob: Developer
Flow of events	• Bob Gets an idea to a new feature that he wants his team to start develop. • Bob begins to write down his idea on a small piece of paper. • When Bob is done describing his idea, he then adds the small piece of paper to the board in the column 'To do'.

Scenariro	Move work item from 'in progress' to 'done'
Participating actor instances	Bob: Developer
Flow of events	• Bob is done with implementing a feature from the 'in progress' column of the board and to make space for new work items he walks to the board, and moves the piece of paper representing the completed work item from the 'in progress' column to the 'done' column.

[1] https://www.atlassian.com/agile/kanban/boards
[2] https://www.atlassian.com/agile/kanban/boards

Exercise 2. *As a design activity (i.e., before writing the code): draw a class diagram representing your design of the entities for the C# part. The purpose of the diagram should be to sketch the main relationships between the entities and their multiplicity. To challenge yourselves, consider that often teams use the boards to also highlight task allocation (i.e., who is responsible for/will complete/has taken a card) and that nowadays pair programming or other group practices are used to implement code. How would you model this feature?*



Exercise 3.

- *The acronym FURPS+ stands for: F_____; U_____; R_____; P_____; S_____; +_____.*
- *The requirements engineering process is an iterative process that includes requirements _____, _____, and _____.*
- *Software engineering is a collection of _____, _____, and _____ that help with the production of a _____ software system developed _____ before a _____ while change occurs.*
- *Requirements need to be complete, _____, _____, and _____.*
- *Important properties of requirements are realism, _____, and _____.*
- *The output of the _____ activity is the _____, which include both the non-functional requirements and the functional model.*
- The accronym FURPS+ stands for
  - F: Functional
  - U: Usability
  - R: Reliability
  - P: Performance
  - S: Supportability
  - +: Other
- The requirements engineering process is an iterative process that includes requirements elicitation, specification, and analysis modeling.
- Requirements need to be complete, consistent, clear, and correct.
- Important properties of requirements are realism, verifiable, and traceable.
- The output of the requirements engineering activity is the RAD, which include both the non-functional requirements and the functional model.

Exercise 4.

- *What level of details should UML models have?*
- *What is the difference between structure diagrams and behavior diagrams in UML? Provide two examples per category.*

- Never more detail than what can be understood by a single person, but enough to not be ambiguous.
- Structure diagrams describe the structure of something, like a class diagram or package diagram. An example of a structure diagram could be an overview of inheritance between classes or which classes implement which interfaces. Where as a behavior diagram describe a sequence of events or a sort of state machine of something. Examples include a sequence diagram, flowchart or state machine.

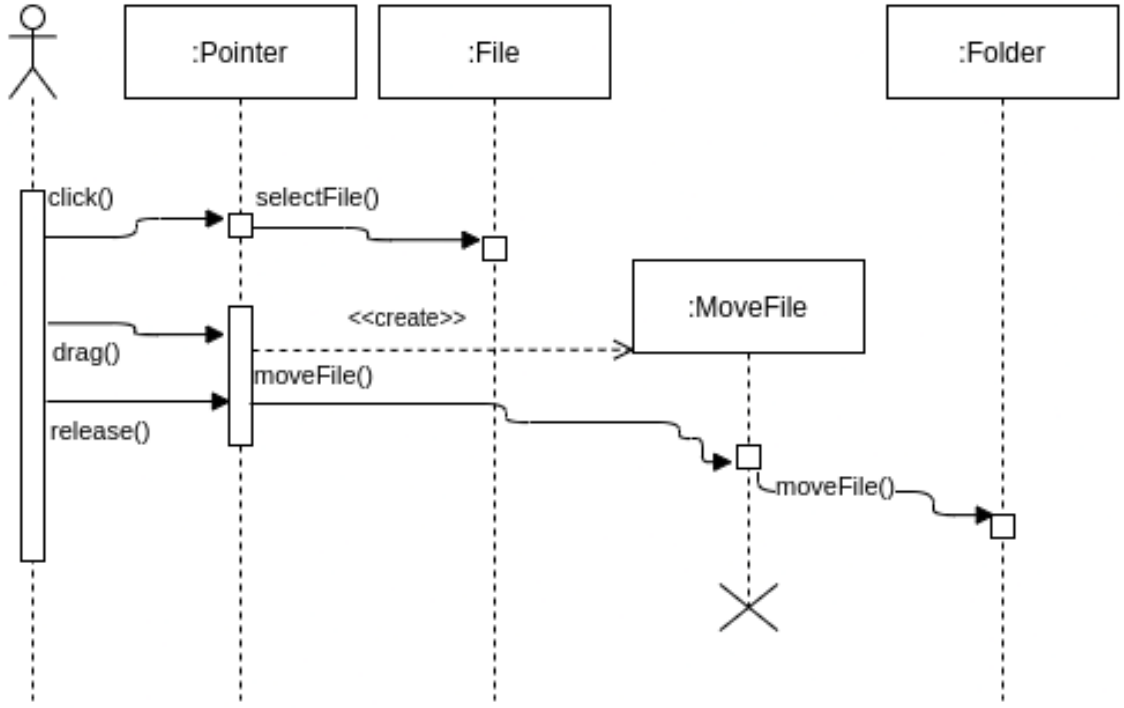
Exercise 5. *Consider a file system with a graphical user interface, such as Macintosh's Finder, Microsoft's Windows Explorer, or Linux's KDE. The following objects were identified from a use case describing how to copy a file from a floppy disk to a hard disk: File, Icon, TrashCan, Folder, Disk, Pointer. Specify which are entity objects, which are boundary objects, and which are control (interactor) objects.*

- Entity:
  - File
  - Icon
  - TrashCan
  - Folder
  - Disk
- Interactor:
  - CopyFile
- Boundary:
  - Pointer

Exercise 6. *Assuming the same file system as before, consider a scenario consisting of selecting a file on a floppy, dragging it to Folder and releasing the mouse. Identify and define one control (interactor) object associated with this scenario.*

The Interactor in this case can be defined as MoveFile. It handles copying the file (or the pointer to the file) to the new location and deleting it from the old one.

Exercise 7. *Arrange the objects listed in Exercises SE.5-6 horizontally on a sequence diagram, the boundary objects to the left, then the control (interactor) object you identified, and finally, the entity objects. Draw the sequence of interactions resulting from dropping the file into a folder. For now, ignore the exceptional cases.*



Exercise 8. *From the sequence diagram Figure 2-34, draw the corresponding class diagram. Hint: Start with the participating objects in the sequence diagram.*

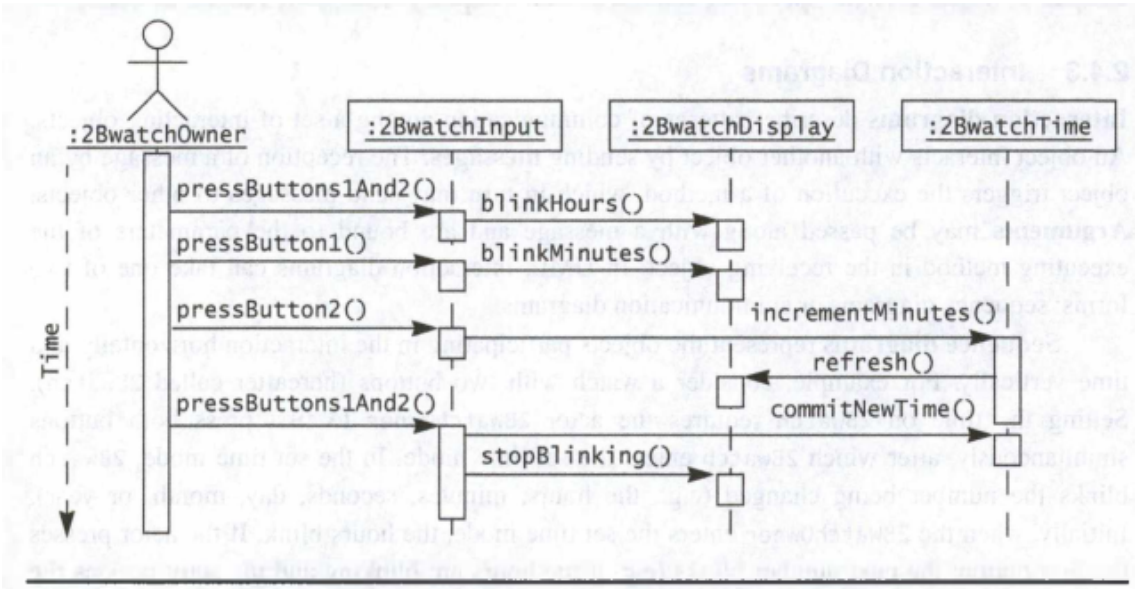
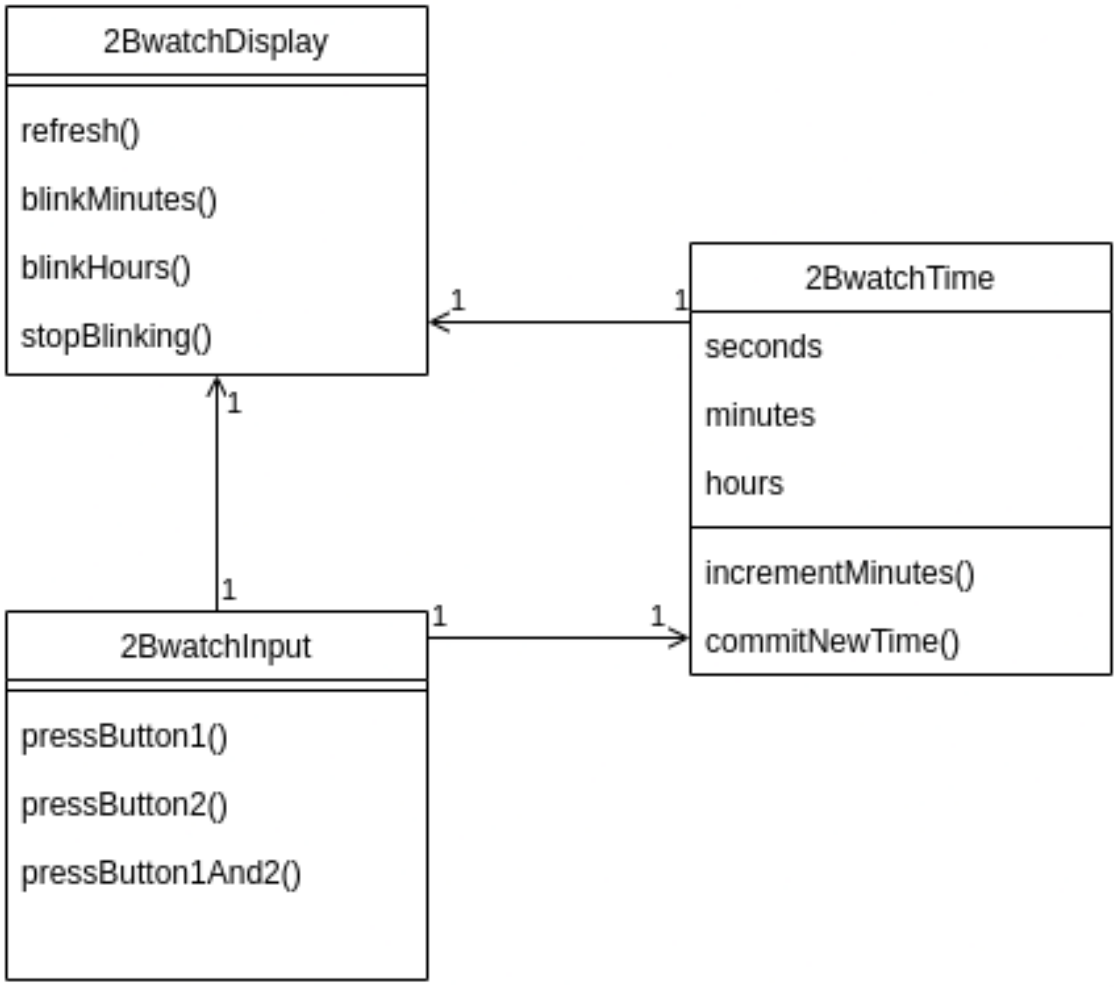


Figure 2-34 Example of a sequence diagram: setting the time on 2Bwatch.


2BwatchDisplay
refresh()
blinkMinutes()
blinkHours()
stopBlinking()
2BwatchTime
seconds
minutes
hours
incrementMinutes()
commitNewTime()
2BwatchInput
pressButton1()
pressButton2()
pressButton1And2()
1
1
1
1
1
1