```
                    «interface»
                    IController

        Start ()
```

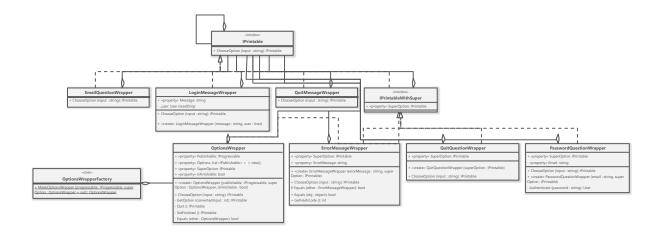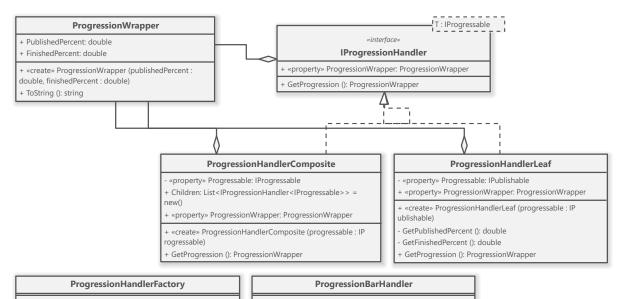```
                  ProgramController

    - «property» User: User
    - «property» ConsolePrinter: ConsolePrinter = = new();
    - _printable: IPrintable
    - «property» Reader: IReader

    + «create» ProgramController (reader : IReader)
    + Start ()
    - SaveUpdates ()
```

**IPrintable** «interface»
+ ChooseOption (input : string): IPrintable

**EmailQuestionWrapper**
+ ChooseOption (input : string): IPrintable

**LoginMessageWrapper**
+ «property» Message: string
- _user: User (readOnly)
+ ChooseOption (input : string): IPrintable
+ «create» LoginMessageWrapper (message : string, user : User)

**QuitMessageWrapper**
+ ChooseOption (input : string): IPrintable

**IPrintableWithSuper** «interface»
+ «property» SuperOption: IPrintable

**OptionsWrapper**
+ «property» Publishable: IProgressable
+ «property» Options: List<IPublishable> = = new();
+ «property» SuperOption: IPrintable
+ «property» IsFinishable: bool
+ «create» OptionsWrapper (publishable : IProgressable, superOption : OptionsWrapper, isFinishable : bool)
+ ChooseOption (input : string): IPrintable
- GetOption (convertedInput : int): IPrintable
- Quit (): IPrintable
- SetFinished (): IPrintable
- Equals (other : OptionsWrapper): bool

**ErrorMessageWrapper**
+ «property» SuperOption: IPrintable
+ «property» ErrorMessage: string
+ «create» ErrorMessageWrapper (errorMessage : string, superOption : IPrintable)
+ ChooseOption (input : string): IPrintable
# Equals (other : ErrorMessageWrapper): bool
+ Equals (obj : object): bool
+ GetHashCode (): int

**QuitQuestionWrapper**
+ «property» SuperOption: IPrintable
+ «create» QuitQuestionWrapper (superOption : IPrintable)
+ ChooseOption (input : string): IPrintable

**PasswordQuestionWrapper**
+ «property» SuperOption: IPrintable
- «property» Email: string
+ ChooseOption (input : string): IPrintable
+ «create» PasswordQuestionWrapper (email : string, superOption : IPrintable)
- Authenticate (password : string): User

**OptionsWrapperFactory** «static»
+ MakeOptionsWrapper (progressable : IProgressable, superOption : OptionsWrapper = null): OptionsWrapper

**ProgressionWrapper**

+ PublishedPercent: double
+ FinishedPercent: double

+ «create» ProgressionWrapper (publishedPercent : double, finishedPercent : double)
+ ToString (): string

---

T : IProgressable

«interface»
**IProgressionHandler**

+ «property» ProgressionWrapper: ProgressionWrapper

+ GetProgression (): ProgressionWrapper

---

**ProgressionHandlerComposite**

- «property» Progressable: IProgressable
+ Children: List<IProgressionHandler<IProgressable>> = new()
+ «property» ProgressionWrapper: ProgressionWrapper

+ «create» ProgressionHandlerComposite (progressable : IProgressable)
+ GetProgression (): ProgressionWrapper

---

**ProgressionHandlerLeaf**

- «property» Progressable: IPublishable
+ «property» ProgressionWrapper: ProgressionWrapper

+ «create» ProgressionHandlerLeaf (progressable : IPublishable)
- GetPublishedPercent (): double
- GetFinishedPercent (): double
+ GetProgression (): ProgressionWrapper

---

**ProgressionHandlerFactory**

+ MakeProgressionHandler (progressable : IProgressable): IProgressionHandler<IProgressable>

---

**ProgressionBarHandler**

+ GenerateProgressBar (progressionWrapper : ProgressionWrapper): String

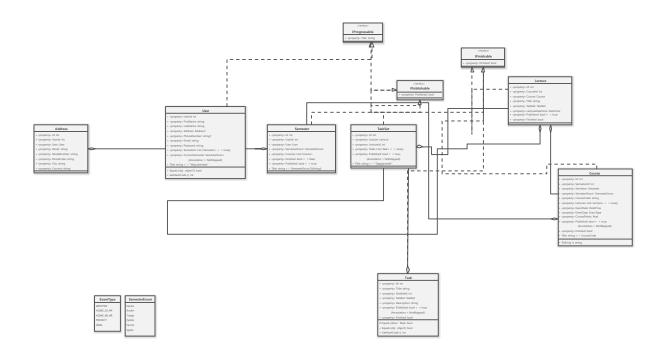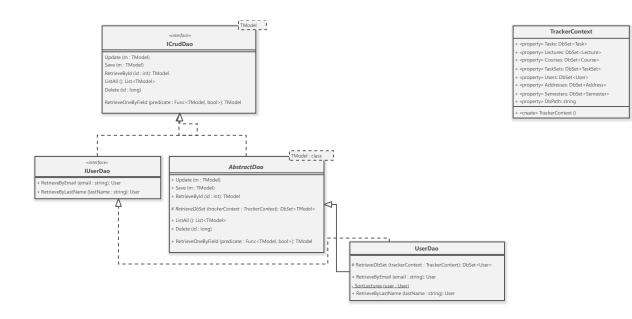- BarGenerator (c : char, amount : long, buffer : StringBuilder)

## migrationsfila
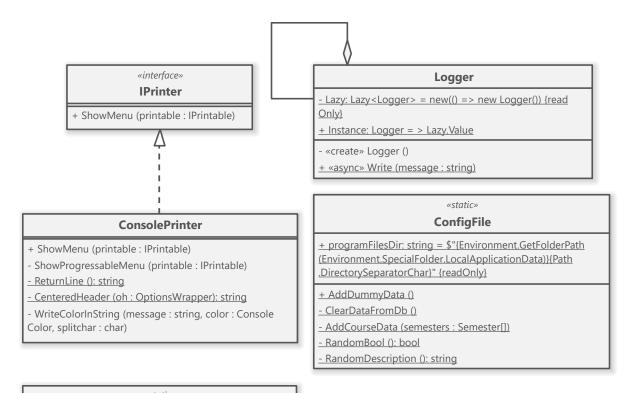
# BuildTargetModel (modelBuilder : ModelBuilder)

## migrationsfila

# Up (migrationBuilder : MigrationBuilder)
# Down (migrationBuilder : MigrationBuilder)

## IProgressable

«interface»
**IProgressable**

+ «property» Title: string

## IFinishable

«interface»
**IFinishable**

+ «property» Finished: bool

## IPublishable

«interface»
**IPublishable**

+ «property» Published: bool

## Lecture

**Lecture**

+ «property» Id: int
+ «property» CourseId: int
+ «property» Course: Course
+ «property» Title: string
+ «property» TaskSet: TaskSet
+ «property» LectureDateTime: DateTime
+ «property» Published: bool = = true;
+ «property» Finished: bool

## Address

**Address**

+ «property» Id: int
+ «property» UserId: int
+ «property» User: User
+ «property» Street: string
+ «property» StreetNumber: string
+ «property» PostalCode: string
+ «property» City: string
+ «property» Country: string

## User

**User**

+ «property» UserId: int
+ «property» FirstName: string
+ «property» LastName: string
+ «property» Address: Address?
+ «property» PhoneNumber: string?
+ «property» Email: string
+ «property» Password: string
+ «property» Semesters: List<Semester> = = new();
+ «property» CurrentSemester: SemesterEnum
    (Annotation = NotMapped)
- Title: string = = "Velg semester"

+ Equals (obj : object?): bool
+ GetHashCode (): int

## Semester

**Semester**

+ «property» Id: int
+ «property» UserId: int
+ «property» User: User
+ «property» SemesterEnum: SemesterEnum
+ «property» Courses: List<Course>
+ «property» Finished: bool = = false;
+ «property» Published: bool = = true;
+ Title: string = = SemesterEnum.ToString()

## TaskSet

**TaskSet**

+ «property» Id: int
+ «property» Lecture: Lecture
+ «property» LectureId: int
+ «property» Tasks: List<Task> = = new();
+ «property» Published: bool = = true;
    (Annotation = NotMapped)
Title: string = = "Oppgavesett"

## Course

**Course**

+ «property» Id: int
+ «property» SemesterId: int
+ «property» Semester: Semester
+ «property» SemesterEnum: SemesterEnum
+ «property» CourseCode: string
+ «property» Lectures: List<Lecture> = = new();
+ «property» ExamDate: DateTime
+ «property» ExamType: ExamType
+ «property» CoursePoints: float
+ «property» Published: bool = = true
    (Annotation = NotMapped)

+ «property» Finished: bool
Title: string = = CourseCode

ToString (): string

## Task

**Task**

+ «property» Id: int
+ «property» Title: string
+ «property» TaskSetId: int
+ «property» TaskSet: TaskSet
+ «property» Description: string
+ «property» Published: bool = = true;
    (Annotation = NotMapped)
+ «property» Finished: bool

# Equals (other : Task): bool
+ Equals (obj : object): bool
+ GetHashCode (): int

## ExamType

**ExamType**

WRITTEN
HOME_24_HR
HOME_48_HR
PROJECT
ORAL

## SemesterEnum

**SemesterEnum**

Første
Andre
Tredje
Fjerde
Femte
Sjette

**«interface»**
**ICrudDao** [TModel]

Update (m : TModel)
Save (m : TModel)
RetrieveById (id : int): TModel
ListAll (): List <TModel>
Delete (id : long)

RetrieveOneByField (predicate : Func<TModel, bool>): TModel

---

**«interface»**
**IUserDao**

+ RetrieveByEmail (email : string): User
+ RetrieveByLastName (lastName : string): User

---

**AbstractDao** [TModel : class]

+ Update (m : TModel)
+ Save (m : TModel)
+ RetrieveById (id : int): TModel

# RetrieveDbSet (trackerContext : TrackerContext): DbSet<TModel>

+ ListAll (): List<TModel>
+ Delete (id : long)

+ RetrieveOneByField (predicate : Func<TModel, bool>): TModel

---

**UserDao**

# RetrieveDbSet (trackerContext : TrackerContext): DbSet<User>

+ RetrieveByEmail (email : string): User
- SortLectures (user : User)
+ RetrieveByLastName (lastName : string): User

---

**TrackerContext**

+ «property» Tasks: DbSet<Task>
+ «property» Lectures: DbSet<Lecture>
+ «property» Courses: DbSet<Course>
+ «property» TaskSets: DbSet<TaskSet>
+ «property» Users: DbSet<User>
+ «property» Addresses: DbSet<Address>
+ «property» Semesters: DbSet<Semester>
+ «property» DbPath: string

+ «create» TrackerContext ()

## «interface»
## IPrinter

+ ShowMenu (printable : IPrintable)

---

## Logger

- Lazy: Lazy<Logger> = new(() => new Logger()) {read Only}
+ Instance: Logger = > Lazy.Value
- «create» Logger ()
+ «async» Write (message : string)

---

## ConsolePrinter

+ ShowMenu (printable : IPrintable)
- ShowProgressableMenu (printable : IPrintable)
- ReturnLine (): string
- CenteredHeader (oh : OptionsWrapper): string
- WriteColorInString (message : string, color : Console Color, splitchar : char)

---

## «static»
## ConfigFile

+ programFilesDir: string = $"{Environment.GetFolderPath (Environment.SpecialFolder.LocalApplicationData)}{Path .DirectorySeparatorChar}" {readOnly}
+ AddDummyData ()
- ClearDataFromDb ()
- AddCourseData (semesters : Semester[])
- RandomBool (): bool
- RandomDescription (): string

---

## «static»
## UserAuthenticator

+ Authenticate (email : string, password : string): User

```
«interface»
IReader

+ ReadLine (): string
```

```
«static»
Program

- Main ()
```

```
Reader

+ ReadLine (): string
```