

## Лабораторна робота №2

### Швець Е.Я. ІІІ-61 ФІОТ

Мною була розроблена архітектура майбутнього додатку згідно з варіантом завдання лабораторної роботи (варіант 13).

#### *Варіант 13. Система керування турнікетом на станції метро*

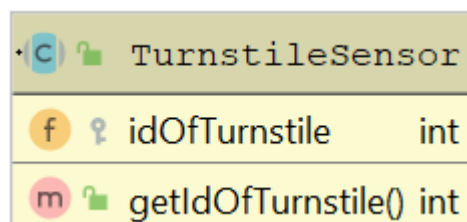
За допомогою турнікета контролюється прохід пасажирів у метро та збирається вхідна платня. Турнікет має приймач карток, пристрій для перекривання доступу, таймер, три оптичні датчики для визначення проходження пасажира, пристрій для подачі звукових сигналів, індикатори «Прохід» та «Стоп».

У початковому стані турнікета висвічується індикатор «Стоп», індикатор «Прохід» не горить. Якщо один з датчиків надсилає сигнал, прохід через турнікет одразу ж перекривається та надсилається попереджувальний звуковий сигнал. Для того, щоб пройти, пасажир повинен помістити картку в приймач карток. Турнікет зчитує з неї дані: термін придатності картки та кількість «одиниць» на ній. Якщо дані не зчитуються, картка прострочена або заблокована, то вона повертається пасажирові, і турнікет залишається в початковому стані. В іншому випадку з картки списується одна «одиниця», картка повертається з приймача, індикатор «Стоп» гасне, засвічується індикатор «Прохід», і пасажир може пройти через турнікет. Отримавши від одного з датчиків сигнал, турнікет очікує час, визначений на проходження пасажира (5 секунд), після чого він повертається в початковий стан.

Наявність трьох датчиків у турнікеті гарантує, що в разі проходження пасажира хоч би один з них подасть сигнал. Під час проходження пасажира можлива ситуація, коли всі три датчика надсилають сигнали. У цьому випадку приймається тільки перший сигнал і від моменту його прийому відраховується призначений час. Решта сигналів ігнорується.

Турнікет заносить у свою пам'ять кількість усіх сплачених проходжень. У кінці робочого дня він передає всю інформацію, накопичену за день, в автоматизовану систему керування метрополітену.





Спочатку були розроблені класи, які описують сенсори турнікету. Було розроблено абстрактний клас **TurnstileSensor**, який має ідентифікатор турнікету **idOfTurnStile** та відповідний геттер.



•	Ⓢ	TurnstileSensor	
f	?	idOfTurnstile	int
m	?	getIdOfTurnstile()	int

Наступними були створені класи, які описують сенсори турнікету, які були успадковані від абстрактного класу **TurnstileSensor**.





























Було визначено, що деякі сенсори повинні працювати в постійному режимі (наприклад, якщо користувач вставив картку у приймач карток – це не означає, що оптичні сенсори руху не повинні працювати, та не працювати індикатори проходу), тому був створений ще один абстрактний клас **AllTimeWorkSensors**, що був успадкований від **TurnstileSensor**.

		AllTimeWorkSensors
		work() boolean

Цей клас має абстрактний метод work().

Нижче приведені сенсори, які повинні постійно працювати, вони були успадковані від **AllTimeWorkSensors**:

### 1) **CardsReceiver** (приймач карток)















		CardsReceiver
		currentDate Date
		isCardInReceiver boolean
		CardsReceiver(int)
		getCurrentDate() Date
		setCurrentDate(Date) void
		readCard(Card) boolean
		work() boolean
		giveCard() void
		tryDecrementTrip(Card) boolean
		checkExpirationDateOfCard(Card) boolean
		checkNumberOfTrips(Card) boolean
		isCardInReceiver() boolean
		setCardInReceiver(boolean) void

Клас має:

- **currentDate** - поле поточної дати для перевірки строку придатності карти;
- **isCardInReceiver** – поле для того, щоб розуміти є карта в приймачі чи ні;
- **getCurrentDate()** – геттер для поточної дати;
- **setCurrentDate()** - сеттер для поточної дати;
- **CardsReceiver(int)** – перевантажений конструктор, який приймає ідентифікатор турнікету;
- **readCard(Card)** - метод для читання з картки;









- **work()** - метод, який буде виконуватись в окремому потоці, тобто працювати постійно;
- **giveCard()** – метод для віддачі карти;
- **tryDecrementTrip(Card)** – метод для списання поїздки;
- **checkExpirationDateOfCard(Card)** – метод для перевірки строку придатності картки;
- **checkNumberOfTrips(Card)** – метод для перевірки кількості поїздок на карті;
- **isCardInreceiver()** – метод для перевірки наявності карти у приймачі;
- **setCardInReceiver(boolean)** – сеттер для картки у приймачі;

## 2) **Indicator** (індикатор)

		Indicator
		indicatorString String
		onn boolean
		Indicator(int, String)
		work() boolean
		on() void
		off() void

- **indicatorString** – повідомлення індикатора;
- **onn** – включений або вимкнений індикатор;
- **Indicator(int, String)** – перевантажений конструктор, що приймає ідентифікатор турнікету та повідомлення індикатора;
- **work()** – метод для постійної роботи індикатора;
- **on()** – метод для включення поточного індикатора;
- **off()** – метод для вимкнення поточного індикатора;









### 3) **OpticalSensor** (оптичний сенсор)

		<b>OpticalSensor</b>
		<b>OpticalSensor(int)</b>
		<b>work()</b> <b>boolean</b>
		<b>sendSignal()</b> <b>boolean</b>

- **OpticalSensor(int)** – перевантажений конструктор, що приймає ідентифікатор турнікету;
- **work()** – метод для постійної роботи оптичного сенсора;
- **sendSignal()** – метод для відправлення сигналу;

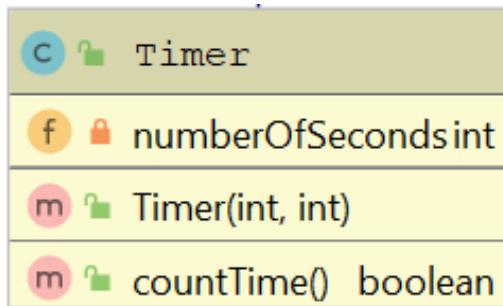
Тепер приведемо сенсори, які не повинні постійно працювати, (вони будуть включатися, коли турнікет їм про це повідомить)

#### 1) **DeviceForSoundSignaling** (пристрій для подачі звукових сигналів)

		<b>DeviceForSoundSignaling</b>
		<b>warningSound</b> <b>TargetDataLine</b>
		<b>DeviceForSoundSignaling(int, TargetDataLine)</b>
		<b>playWarningSound()</b> <b>void</b>

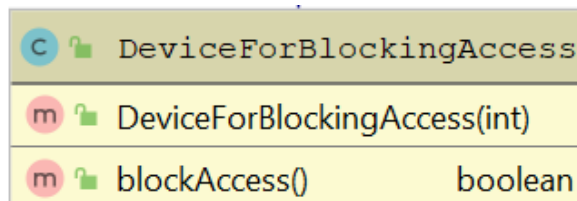
- **warningSound** – звуковий сигнал;
- **DeviceForSoundSignaling(int, TargetDataLine)** – перевантажений конструктор, що приймає ідентифікатор турнікету та звуковий сигнал;
- **playWarningSound()** – метод для відтворення звуку;

## 2) Timer (таймер)



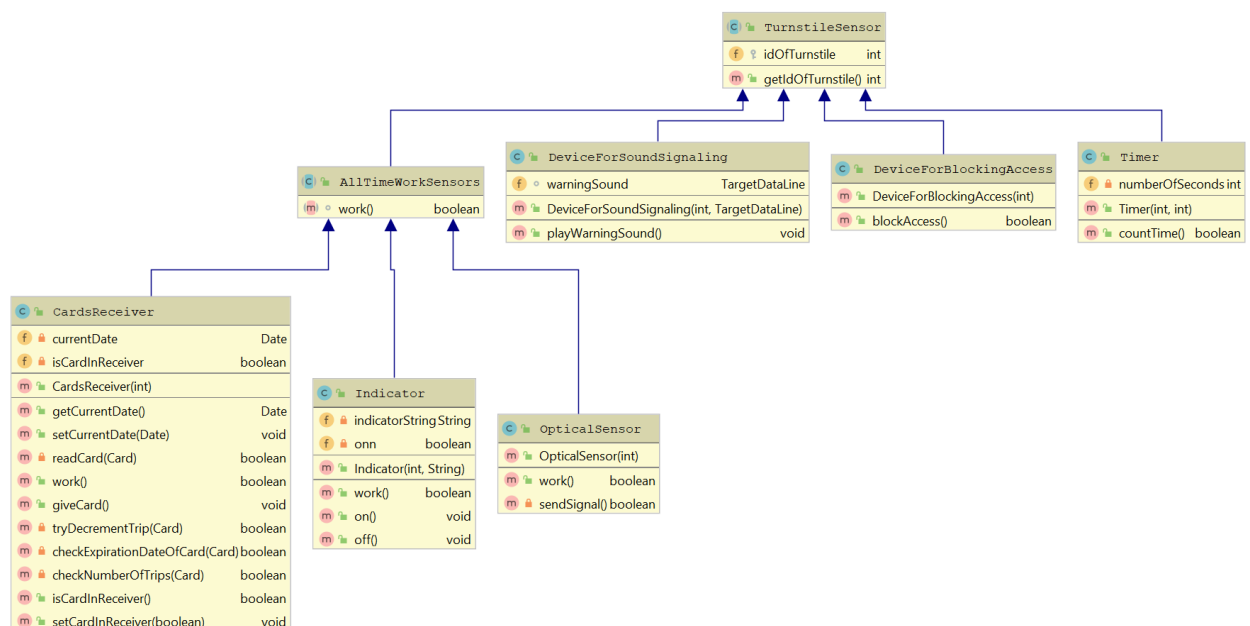
- **numberOfSeconds** – кількість секунд для відліку;
- **Timer(int, int)** – перевантажений конструктор, що приймає ідентифікатор турнікету та кількість секунд для відліку;
- **countTime ()** – метод для відліку часу;

## 3) DeviceForBlockingAccess (девайс для блокування проходу)



- **DeviceForBlockingAccess(int)** – перевантажений конструктор, що приймає ідентифікатор турнікету;
- **blockAccess()** – метод для блокування доступу;

Далі приведена уся ієрархія сенсорів:



Після розроблення сенсорів були створені клас **Turnstile** (турнікет) і **Card** (картка метрополітену).

**Card** (картка метрополітену)

C Card		
f	id	int
f	expirationDate	Date
f	numberOfTrips	int
m	Card(int, Date, int)	
m	getExpirationDate()	Date
m	setExpirationDate(Date)	void
m	getNumberOfTrips()	int
m	setNumberOfTrips(int)	void
m	getId()	int

- **id** – ідентифікатор картки;
- **expirationDate** – строк придатності картки;
- **numberOfTrips** – кількість поїздок на карті;
- **Card(int, Date, int)** – перевантажений конструктор, що приймає ідентифікатор турнікету, строк придатності карти та кількість поїздок на ній;
- **getExpirationDate()** – геттер для строку придатності карти;
- **setExpirationDate(Date)** – сеттер для строку придатності карти;
- **getNumberOfTrips()** – геттер для кількості поїздок на карті;
- **setNumberOfTrips(int)** – сеттер для кількості поїздок на карті;
- **getId()** – геттер для ідентифікатора картки;

## Turnstile (турнікет)

Turnstile		
f	id	int
f	executorService	ExecutorService
f	numberOfPasses	int
f	state	State
f	cardsReceiver	CardsReceiver
f	deviceForBlockingAccess	DeviceForBlockingAccess
f	timer	Timer
f	opticalSensor1	OpticalSensor
f	opticalSensor2	OpticalSensor
f	opticalSensor3	OpticalSensor
f	deviceForSoundSignaling	DeviceForSoundSignaling
f	indicatorPASS	Indicator
f	indicatorSTOP	Indicator
m	Turnstile()	
m	Turnstile(int, ExecutorService, int, State, CardsReceiver, DeviceForBlockingAccess, Timer, OpticalSensor, OpticalSensor, OpticalSensor, DeviceForSoundSignaling, Indicator, Indicator)	
m	getId()	int
m	getExecutorService()	ExecutorService
m	getNumberOfPasses()	int
m	work()	void
m	workCardReceiver(Card)	void
m	workOpticalSensors(boolean)	void
m	workIndicators()	void
m	changeState(State)	void
m	getState()	State
m	setId(int)	void
m	setExecutorService(ExecutorService)	void
m	setNumberOfPasses(int)	void
m	setState(State)	void
m	setCardsReceiver(CardsReceiver)	void
m	setDeviceForBlockingAccess(DeviceForBlockingAccess)	void
m	setTimer(Timer)	void
m	setOpticalSensor1(OpticalSensor)	void
m	setOpticalSensor2(OpticalSensor)	void
m	setOpticalSensor3(OpticalSensor)	void
m	setDeviceForSoundSignaling(DeviceForSoundSignaling)	void
m	setIndicatorPASS(Indicator)	void
m	setIndicatorSTOP(Indicator)	void
m	newBuilder()	Builder
m	doConnection()	boolean

- **id** – ідентифікатор турнікету;
- **executorService** – пул потоків для паралельної роботи декількох сенсорів;
- **numberOfPasses** – кількість сплачених проходжень через турнікет;
- **state** – стан турнікету (STOP, PASS);
- **cardsReceiver** – приймач карток;
- **deviceForBlockingAccess** – девайс для блокування проходження;
- **timer** - таймер;
- **opticalSensor1** – перший оптичний сенсор;
- **opticalSensor2** – другий оптичний сенсор;
- **opticalSensor3** – третій оптичний сенсор;
- **deviceForSoundSignaling** – девайс для подачі звукового сигналу;
- **indicatorPASS** – індикатор «Pass»
- **indicatorSTOP** – індикатор «Stop»;

- **Turnstile()** – конструктор за замовчуванням;
- **Turnstile(int, ExecutorService, int, State, CardsReceiver, DeviceForBlockingAccess, Timer, OpticalSensor, OpticalSensor, OpticalSensor, DeviceForSoundSignaling, Indicator, Indicator)** – перевантажений конструктор, що приймає ідентифікатор турнікету, усі датчики турнікету, стан турнікету, та пул потоків;
- **getId()** - геттер для ідентифікатору турнікету;
- **getExecutorService()** – геттер для пула потоків;
- **getNumberOfPasses()**- геттер для кількості проходжень;
- **work()**- метод, який запускає турнікет у роботу;
- **workCardReceiver(Card)** – метод, який запускає у роботу приймач карток;
- **workOpticalSensors(boolean)** - метод, який запускає у роботу оптичні сенсори;
- **workIndicators()**- метод, який запускає у роботу індикатори;
- **changeState(State)** – метод, що змінює стан турнікету;
- **getState()**- геттер для стану;
- **setId(int)** – сеттер для ідентифікатору турнікету;
- **setExecutorService(ExecutorService)** – сеттер для пулу потоків;
- **setNumberOfPasses(int)** – сеттер для кількості проходжень;
- **setState(State)** – сеттер для стану;
- **setCardsReceiver(CardsReceiver)** – сеттер для приймача карток;
- **setDeviceForBlockingAccess(DeviceForBlockingAccess)** – сеттер для девайсу для блокування проходження;
- **setTimer(Timer)** – сеттер для таймеру;
- **setOpticalSensor1(OpticalSensor)** – сеттер для першого оптичного сенсору;
- **setOpticalSensor2(OpticalSensor)** – сеттер для другого оптичного сенсору;
- **setOpticalSensor3(OpticalSensor)** – сенсор для третього оптичного сенсору;



- **setDeviceForSoundSignaling(DeviceForSoundSignaling)** – сеттер для девайсу для подачі звукового сигналу;
- **setIndicatorPASS(Indicator)** – сеттер для індикатора «PASS»;
- **setIndicatorSTOP(Indicator)** – сеттер для індикатора «STOP»;
- **newBuilder()**- метод для створення білдеру;
- **doConnection()**- метод для створення з'єднання;

У класі **Turnstile** (турнікет) було створено клас **Builder** - це породжуючий патерн проектування, який дозволяє створювати складні об'єкти покроково. **Builder** дає можливість використовувати один і той же код будівництва для отримання різних уявлень об'єктів.

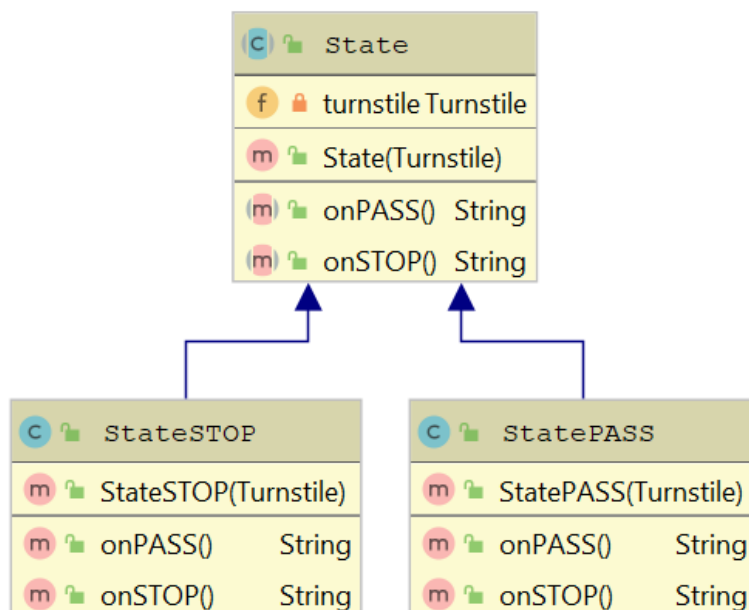
## Builder

Builder	
Builder()	
setId(int)	Builder
setExecutorService(ExecutorService)	Builder
setNumberOfPasses(int)	Builder
setState(State)	Builder
setCardsReceiver(CardsReceiver)	Builder
setDeviceForBlockingAccess(DeviceForBlockingAccess)	Builder
setTimer(Timer)	Builder
setOpticalSensor1(OpticalSensor)	Builder
setOpticalSensor2(OpticalSensor)	Builder
setOpticalSensor3(OpticalSensor)	Builder
setDeviceForSoundSignaling(DeviceForSoundSignaling)	Builder
setIndicatorPASS(Indicator)	Builder
setIndicatorSTOP(Indicator)	Builder
build()	Turnstile

- **Builder()** – конструктор за замовчанням для створення білдеру;
- **setId(int)** – сеттер для ідентифікатору турнікету;
- **setExecutorService(ExecutorService)** – сеттер для пулу потоків;
- **setNumberOfPasses(int)** – сеттер для кількості проходжень;
- **setState(State)** – сеттер для стану;
- **setCardsReceiver(CardsReceiver)** – сеттер для приймача карток;
- **setDeviceForBlockingAccess(DeviceForBlockingAccess)** – сеттер для девайсу для блокування проходження;
- **setTimer(Timer)** – сеттер для таймеру;

- **setOpticalSensor1(OpticalSensor)** – сеттер для першого оптичного сенсору;
- **setOpticalSensor2(OpticalSensor)** – сеттер для другого оптичного сенсору;
- **setOpticalSensor3(OpticalSensor)** – сенсор для третього оптичного сенсору;
- **setDeviceForSoundSignaling(DeviceForSoundSignaling)** – сеттер для девайсу для подачі звукового сигналу;
- **setIndicatorPASS(Indicator)** – сеттер для індикатора «PASS»;
- **setIndicatorSTOP(Indicator)** – сеттер для індикатора «STOP»;
- **build()**- метод для створення екземпляру турнікета;

Був створений абстрактний клас **State** - це поведінковий патерн, що дозволяє динамічно змінювати поведінку турнікету при зміні його стану. Поведінки, які залежать від стану, переїжджають в окремі класи. Початковий клас зберігає посилання на один з таких об'єктів-станів і делегує йому роботу. Отже, також були створені класи **StateSTOP**, **StatePASS**.



**State** має посилання на турнікет, конструктор з параметром турнікетом, та методи абстрактного стану. В успадкованих класах **StateSTOP** та **StatePASS** ці методи будуть перевизначені.

В умові зазначено, що:

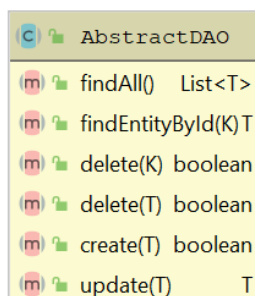
Турнікет заносить у свою пам'ять кількість усіх сплачених проходжень. У кінці робочого дня він передає всю інформацію, накопичену за день, в автоматизовану систему керування метрополітену.

Було розроблено базу даних (1 таблиця) для зберігання інформації про кількість усіх сплачених проходжень.

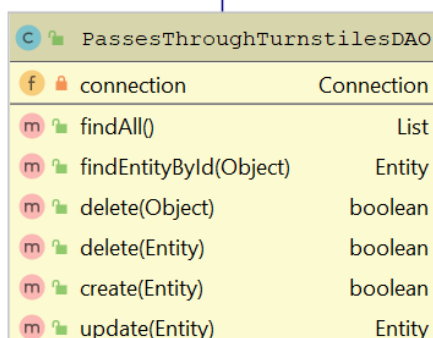
PassesStatistic		
PK	id	INT
	turnstile_id	INT
	num_of_passes	INT

- **id** – ідентифікатор запису у таблицю (**primary key**)
- **turnstile\_id** – ідентифікатор турнікету, з якого отримали інформацію
- **num\_of\_passes** – кількість сплачених проходжень

Також був створений прошарок між додатком і СУБД - DAO (Data Access Object). Вершина ієрархії DAO - це абстрактний клас з загальними методами взаємодії з БД. Мною були оголошені методи вибору, пошуку за ознакою, додавання, видалення і заміни даних.



В успадкованому класі був визначений **connection** - з'єднання з БД для виконання декількох методів мого DAO-класу (**PassesThroughTurnstilesDAO**).



Була визначена структура проекту – MVC - схема поділу даних програми на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

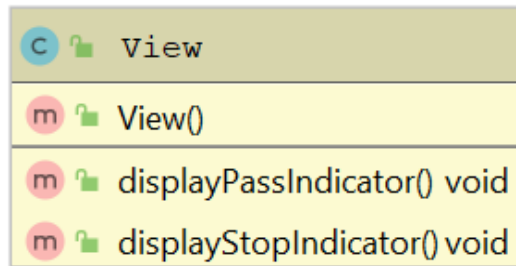
Були створені класи Controller та View.

## Controller

Controller		
f	turnstile	Turnstile
f	view	View
m	Controller(Turnstile, View)	
m	workTurnstile()	void
m	workCardReceiver()	void
m	workOpticalSensors()	void
m	workIndicators()	void
m	changeState(State)	void
m	getState()	State

- **turnstile** - турнікет;
- **view** - представлення;
- **Controller(Turnstile, View)** – конструктор контролеру, який приймає турнікет та уявлення;
- **workTurnstile()** – метод для запуску роботи турнікету;
- **workCardReceiver()** – метод для запуску роботи приймача карток;
- **workOpticalSensors()** – метод для запуску роботи оптичного датчика;
- **workIndicators()** – метод для запуску роботи індикаторів;
- **changeState(State)** – метод для зміни стану турнікета;
- **getState()** – геттер для стану турнікета;

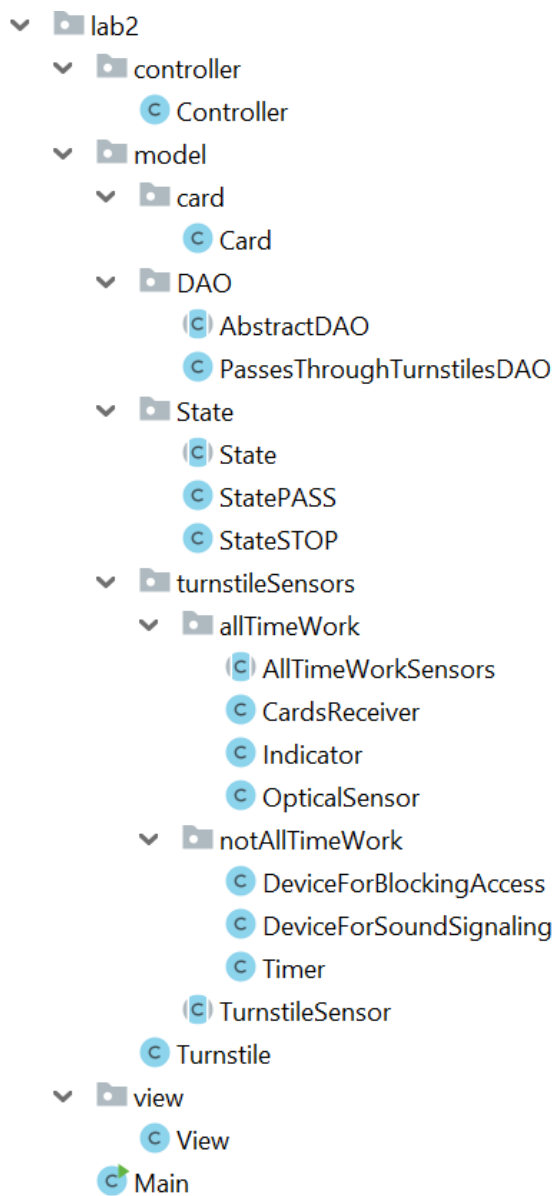
## View



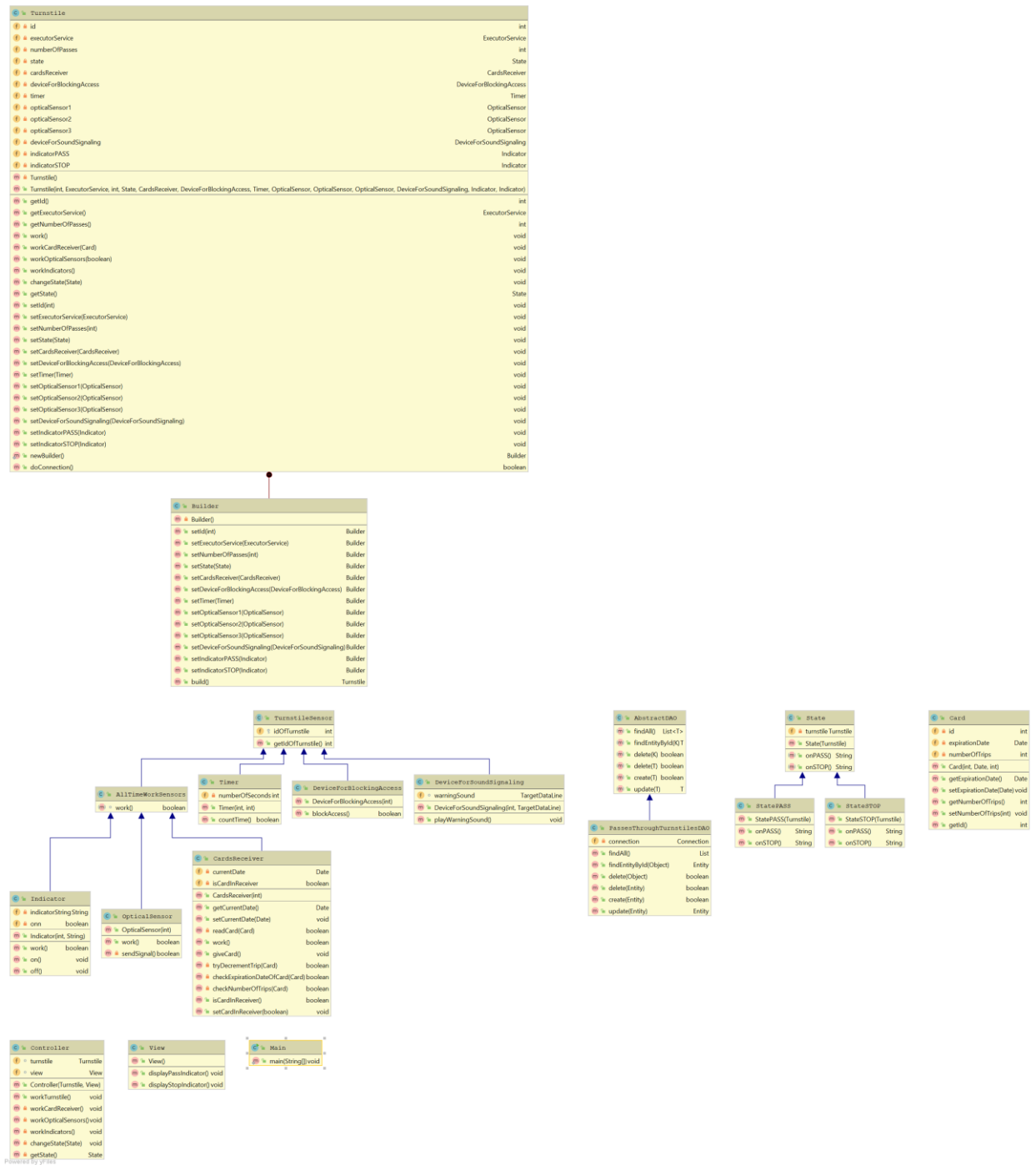
Поки що **View** має тільки методи відображення індикаторів STOP та PASS.

Усе інше, а саме: турнікет, сенсори турнікету, картка метрополітену, DAO, State – є моделлю.

Нижче приведено дерево класів:



Powered by yFiles



## Діаграма класів з усіма залежностями:

