Fakulta informatiky a informačných technológií
STU Bratislava

# Objektovo-orientované programovanie

## **Správa o realizácii projektu**

Jakub Povinec
čas cvičení: streda 18:00

# Zámer projektu

## PlaneTracker

Moje poňatie rámcovej témy „Dôveryhodný softvér“ je aplikácia na sledovanie pohybu lietadiel.

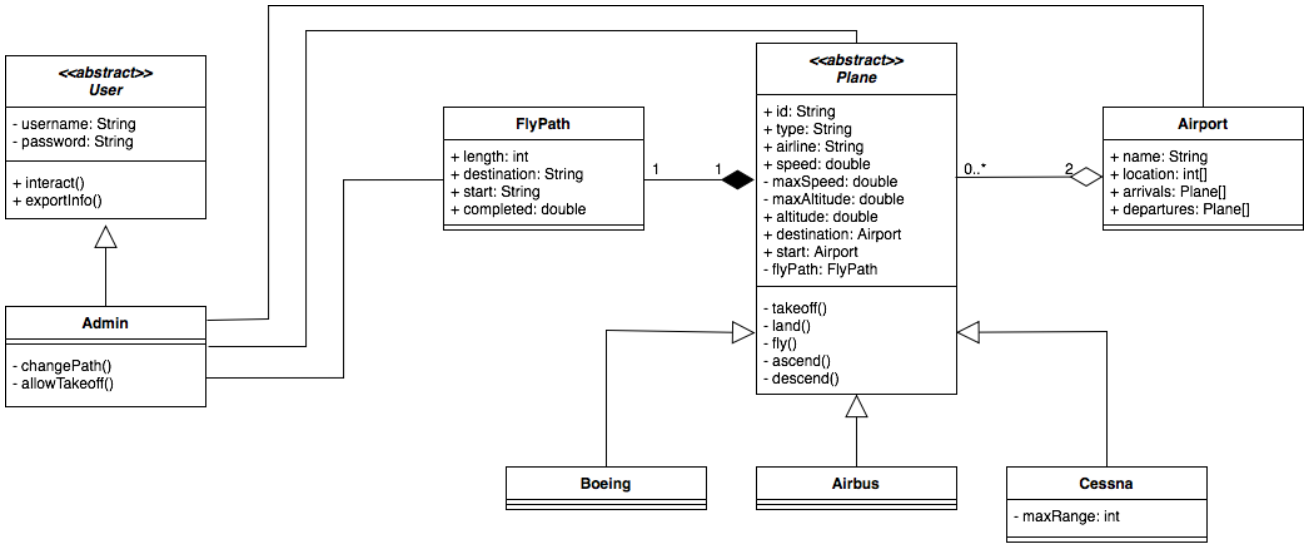
Používateľské rozhranie bude spočívať z mapy, na ktorej budú body reprezentujúce letiská. Lietadlá, ktoré sa budú pohybovať medzi letiskami, budú znázornené obrázkom lietadla. Trasa lietadiel bude reprezentovaná úsečkou medzi dvoma letiskami, kde časť, ktorú lietadlo už prešlo, bude označená inou farbou.

Štandardný používateľ bude môcť sledovať pohyb lietadiel a interagovať s mestami a lietadlami na mape. Po kliknutí na lietadlo sa zobrazia informácie o lietadle, ako je výška, rýchlosť, mesto odkiaľ vzlietlo a mesto kam smeruje. Taktiež sa bude dať kliknúť aj na letiská, kde sa používateľovi zobrazia lietadlá, ktoré práve z letiska odleteli alebo na letisko smerujú. Používateľ si bude môcť vygenerovať textový súbor s týmito informáciami o lietadle alebo letisku. Informácie o lietadlách sa budem snažiť nasimulovať podľa typu lietadla (Boeing 797, Airbus A350, Cessna 172) a v akej časti letu sa práve nachádzajú (či stúpajú, klesajú...).

Do aplikácie sa bude môcť prihlásiť aj administrátor, ktorý bude oprávnený meniť dráhu lietadiel alebo dať lietadlám, ktoré sa nachádzajú na letisku, povolenie na štart.

Môj softvér sa dá považovať za dôveryhodný, keďže len oprávnený používateľ dokáže zmeniť stav lietadla.

**Diagram tried zo zámeru:**



## Pracovná verzia

V pracovnej verzií som sa sústredil hlavne na vytvorenie grafického rozhrania, hlavne zobrazenia informácií o letiskách a lietadlách.

V aplikácií sa v tomto bode dali:

- zobraziť prílety a odlety zvoleného letiska
- pridať nové lietadlá so štartom vo zvolenom letisku
- dali sa zobraziť všetky lietadlá
- po dvojkliku na lietadlo v tabuľkách sa o ňom zobrazia informácie

Funkcionalita, ktorá mojej aplikácií chýbala a chcel som ju do finálneho odovzdania doplniť:

- simulácia letu lietadla
- vedieť zobraziť lietadlo a jeho dráhu na mape
- registrácia nových používateľov
- ošetriť výnimky
- vedieť vygenerovať textový súbor s informáciami o lietadle/letisku

## Finálne odovzdanie projektu

Vo finálnom odovzdaní sa mi podarilo dokončiť simuláciu letu lietadla, v aplikácií je na mape vidno lietadlá a po kliknutí na nich aj ich dráhu. Taktiež sa mi podarilo pridať registráciu nových používateľov a vytvoril som aj pár vlastných výnimiek. Nepodarilo sa mi ale dokončiť generovanie textového súboru. Veľa času mi zabrala práve simulácia letu lietadiel a ich vykresľovanie a keďže som už mal málo času, rozhodol som sa, že túto funkcionalitu z môjho riešenia vypustím.

Myslím, že tú podstatnú časť zadania, ktorú som si určil nazačiatku v zámere projektu som splnil. Jediná funkcionalita, ktorá chýba je generovanie textových súborov a rozlišovanie medzi normálnym používateľom a administrátorom.

### Hlavné kritéria:

Podrobnejší opis hlavných aj vedľajších kritérií je dostupný v README.

**dedenie** sa napríklad nachádza v triedach Airbus, Boeing a Cessna, ktoré dedia z abstraktnej triedy Plane.

**agregácia** sa nachádza v triede Airport, kde jedno letisko môže mať lietadlá, ktoré doň smerujú alebo z neho odchádzajú. Ak by letisko zaniklo, tak nezaniknú lietadlá.

**kompozícia** sa nachádza v triede Plane, kde jedno lietadlo má práve jednu dráhu letu FlightPath, ktorá zanikne spolu s lietadlom.

**polymorfizmus** sa nachádza napríklad v triede PlaneInfoController, kde sa zavolá metóda plane.getManufacturer, na ktorú rôzne podtriedy triedy Plane reaguju iným výpisom. Premennú manufacturer má každá trieda "na pevno" danú.

## Vedľajšie kritéria:

použil som **návrhový vzor** mediator, ktorý som uplatnil pri aktualizovaní stavu lietadla. Každé lietadlo komunikuje so svojim letiskom start. Toto letisko im následne môže zmeniť stav, či majú stúpať, klesať, pristávať alebo len tak letieť.

**vlastné výnimky** som uplatnil pri serializácií v triede Serialization, kde sa keď je súbor user.txt prázdny vyhodí výnimka NoRegisteredUsersException. Druhú výnimku som uplatnil v rozhraní PlaneInfo, kde sa keď používateľ zvolí prázdnu bunku v TableView vyhodí výnimka BlankTableException.

**multithreading** som využil v triedach AirTraffic a PlaneInfoController.

**RTTI** som využil v MapController v metode initializeAirports(), ďalej v triede AirTraffic a v triede PlaneRenderer.

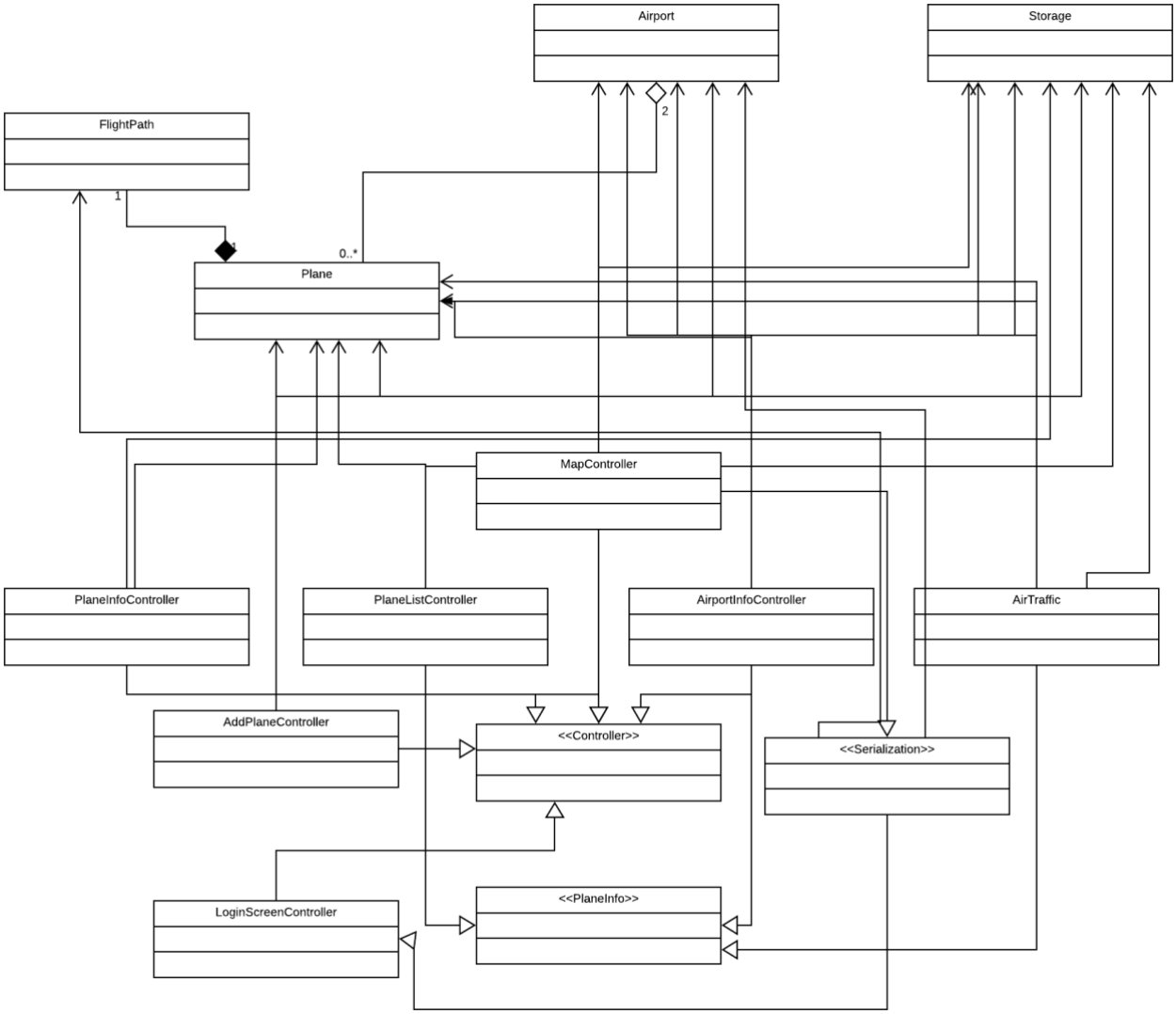
použitie **vzhniezdených rozhraní** v triede AirTraffic, AirportInfoController, PlaneListController.

použitie **lambda výrazov a referencií na metódy** lambda výrazy som použil skoro všade, ale napríklad v triede AirTraffic som použil aj lambda výrazy aj referencie na metódy.

**implicitné implementácie metód v rozhraniach** som použil v rozhraniach Controller a PlaneInfo

**serializáciu** som použil v triedach Serialization, LoginScreenController, RegisterController, MapController.

## Môj UML diagram tried:




Toto je diagram tried, ktorý som sa pokúsil vytvoriť. Neobsahuje všetky triedy, len tie čo som považoval za dôležité a taktiež som im nedal atribúty a metódy, lebo by to bolo ešte viac neprehľadné.

Trieda **Plane** je s triedou **FlightPath** v **kompozícií**, pretože trieda FlightPath sa vytvorí len v triede Plane, takže keď prestane existovať objekt triedy Plane, tak prestane existovať aj objekt triedy FlightPath. Trieda **Airport** je s triedou Plane zas v **agregácií**. Pretože keby jeden z tých objektov zanikol, neznamenalo by to, že zanikne aj ten druhý.

Všetky **Controllery** implementujú rozhranie **Controller**. Triedy **AirTraffic**, **AirportInfoController** a **PlaneInfoController** implementujú rozhranie **PlaneInfo**. Triedy **MapController** a **LoginScreenController** zas dedia z abstraktnej triedy **Serialization**.

Ostatné vzťahy som dal len ako asociácie.

## Vygenerovaný UML diagram tried: