Graz University of Technology

Maschinenbau- und Betriebsinformatik

Ingenieurinformatik II

Basisprojekt – Simulationstool eines Flughafens zur Berechnung der Auslastung

Wintersemester 2023/2024

In der vorliegenden Prüfungsangabe wird auf die gleichzeitige Nennung der geschlechterspezifischen Sprachformen aus Gründen der Lesbarkeit verzichtet. Das Abstellen auf die männliche Form soll hierbei geschlechterunabhängig verstanden und Personen **aller Zugehörigkeiten** angesprochen werden.

Picture source: Clipart Airplane Tower - Cartoon Pictures Of A Airport , Transparent Cartoon - Jing.fm, last access: 12.06.2023

Änderungsindex

Index	Folien-nummer	Änderung
v3	gesamter Foliensatz	Grammatik- und Rechtschreibfehler ausgebessert, sowie Verweise adjustiert
v2	2	Ergänzung der Erläuterung der Koordinaten
v2	20	DoSimulation()-Flowchart Ergänzung der SetStatus()
v3	15	checkBriefing() und checkWalkAorund() erklärt
v3	19	path bei Read… erklärt

Anmerkung/Tipps

- Einfache Get- und Set-Methoden sind im Klassendiagramm angegeben, werden aber im weiteren Verlauf nicht mehr erwähnt, es wird davon ausgegangen, dass diese Methoden ohne weitere Hilfestellung verarbeitet werden können. Achtung: manchmal heißen die Get- und Set-Methoden nicht Get... oder Set..., sollten diese Methoden nicht beschrieben sein, so setzen oder geben Sie das entsprechende Attribut zurück.

- Achten Sie auf Trennzeichen in csv-Dateien, diese können variieren. Diesbezüglich achten Sie auch auf das Kommatrennzeichen, je nach Version, und Spracheinstellung können hier Stolperfallen sein.

- Bei den Koordinaten wird immer zuerst die Latitude coordinate und dann die Longitude Koordinate angegeben

- Gehen Sie wirklich Schritt für Schritt vor und arbeiten Sie sich von Methode zu Methode.

Problembeschreibung

Am Flughafen in Graz soll eine neue Start- und Landebahn errichtet werden. Für die Genehmigung dieses Projekts soll ermittelt werden, welche Auslastung der neue Flughafen erbringt und ob der Bau damit gerechtfertigt ist.

Dafür wird ein Flughafen herangezogen und ein typischer Tagesablauf durch ein Programm simuliert. Hier soll ein Einblick gegeben werden, wie reale Abläufe als Modell abgebildet werden können und welche Auswirkungen unterschiedliche Zuweisungen von Flugzeugen, Besatzung sowie die Dauer des Boardings auf die Flüge und damit auch auf den zeitlichen Ablauf des Flughafens haben.

Der Ablauf eines Flughafens ist mehr als nur Einchecken, SafetyCheck und Boarding. Um diesen Sachverhalt auf einer höheren Ebene zu betrachten, soll nun ein Programm geschrieben werden, welches auch andere Facetten dieses Ablaufs darstellt.

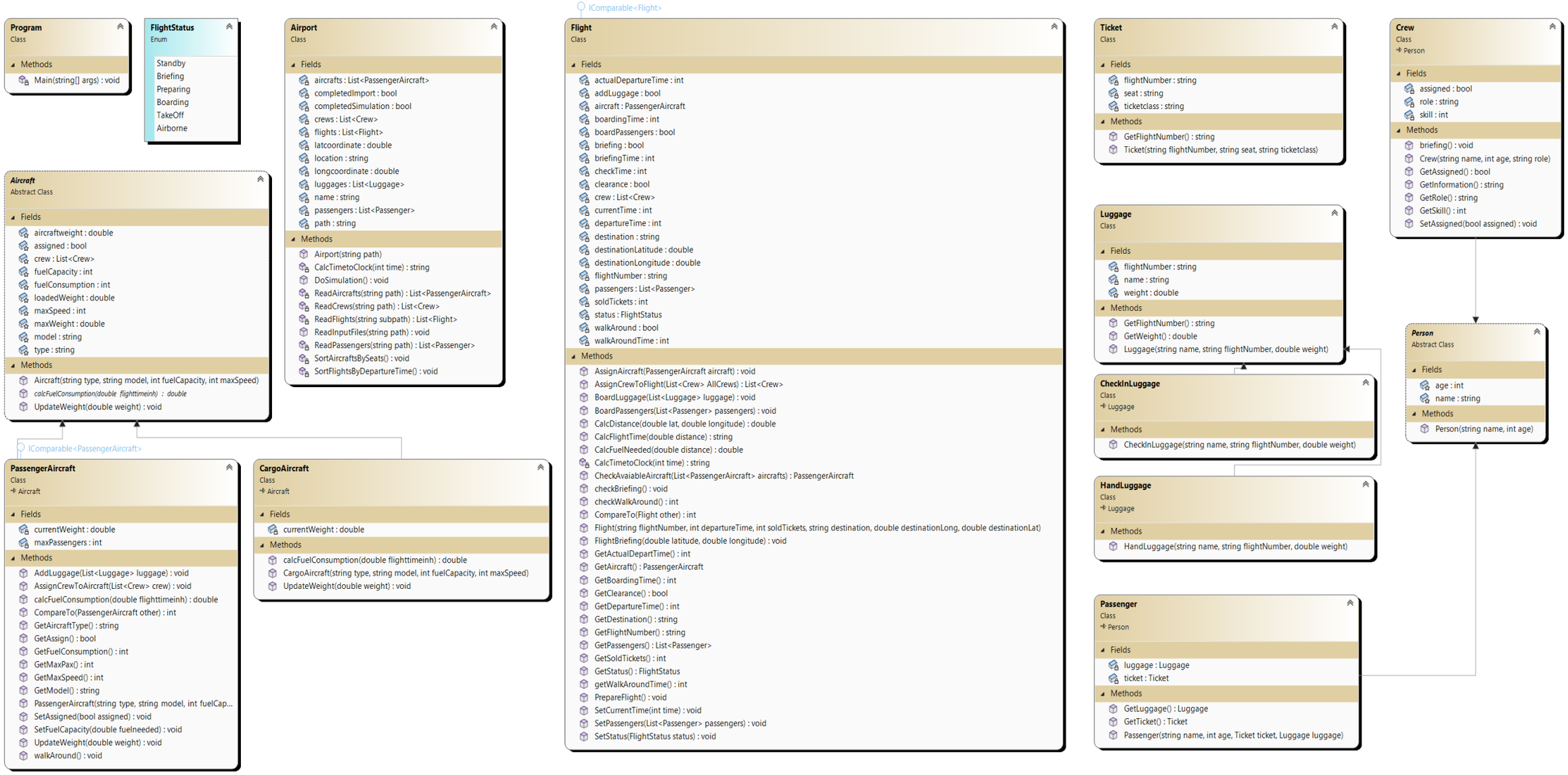
Als Inputdaten für Ihr Programm wurden Daten für den neuen Flughafen erzeugt. Diese beinhalten, Flüge, Flugzeuge, Besatzung als auch Passagiere mit Ticket und Gepäck. Aus diesen Daten soll nun ein erster Prototyp für den Tagesablauf des Flughafens erstellt werden.

Grundidee

Die folgende Beschreibung soll den grundsätzlichen Ablauf des Programmes skizzieren. Genauere Beschreibung folgt. Folgende Programmschritte sollten in dieser empfohlenen Reihenfolge programmiert und getestet werden:

1. Einlesen der Inputdaten
   Mithilfe des hinterlegten Dateipfades des Flughafens, sollen gemäß der Flüge-, Flugzeug- und Personenlisten die entsprechenden Objekte erstellt und in separaten Listen abgespeichert werden.
   (Zugeordnete Methoden: `ReadFlights()`, `ReadAircrafts()`, `ReadCrew()` und `ReadPassengers()`)
2. Simulation
   a) Passendes Flugzeug finden
      Hier soll die Flugzeugliste durchsucht werden, um ein freies Flugzeug mit ausreichend Plätzen zu finden
   b) Crew für Flug finden
      Die Crewliste soll je nach Flugzeugtyp eine Liste zurückgeben, welche das passende Personal dem Flug zuordnet
   c) Flug vorbereiten
      Distanz berechnen, Treibstoffmenge bestimmen sowie Briefing und WalkAround sind Tätigkeiten, die vor dem Boarding durchgeführt werden müssen
   d) Boarding
      Die Tickets der Passagiere müssen überprüft werden, damit sie und ihr Gepäck boarden können
   e) Take Off
      Das Flugzeug verlässt den Flughafen, in der Simulation wird dieser Flug nun als `Airborne` gekennzeichnet

Entwurf der Klassenstruktur:



Klassenbeschreibung

Enumeration: FlightStatus

Attribute:
`Standby, Briefing, Preparing, Boarding, TakeOff, Airborne`

Funktion:
Um die Lesbarkeit und das Verständis zu erleichtern wird mit Enums gearbeitet, um nicht Abfragen nach Status 2 zu machen, ohne zu wissen, welche Bedeutung Status 2 bedeutet (hat im Hintergrund eine int-Repräsentation)

KLASSE: Ticket

Attribute:

`flightNumber`	Flugnummer, z.B.: „II234“ - (string)
`seat`	Sitzplatzbezeichnung, z.B.: F16 - (string)
`ticketclass`	Ticketkategorie, z.B.: Economy, Business, First - (string)

Konstruktor:

`Ticket()`	Dem Konstruktor werden folgende Parameter übergeben: Flugnummer, Sitzplatz und Ticketklasse. Diese Parameter müssen im Konstruktor den entsprechenden Attributen zugeordnet werden.

Klassenbeschreibung

KLASSE: Luggage

Attribute:

`name`	Namen des Besitzes des Gepäcksstücks - (string)
`flightNumber`	Flugnummer des Besitzers des Gepäcksstücks - (string)
`weight`	Gewicht des Gepäckstücks - (double)

Konstruktor:

`Luggage()`	Dem Konstruktor werden folgende Parameter übergeben: Name des Besitzes, Flugnummer und Gewicht. Diese Parameter müssen im Konstruktor den entsprechenden Attributen zugeordnet werden.

KLASSE: HandLuggage : Luggage

Konstruktor:

`HandLuggage()`	Der Konstruktor ruft den base-Konstruktor auf und das Gewicht des Gepäckstücks wird automatisch auf 8 kg gesetzt

KLASSE: CheckInLuggage : Luggage

Konstruktor:

`CheckInLuggage()`	Der Konstruktor ruft den base-Konstruktor auf

Klassenbeschreibung

Abstrakte KLASSE: Person

Attribute:

`age`	Alter der Person - (int)
`name`	Namen der Person - (string)

Konstruktor:

`Person()`	Dem Konstruktor werden folgende Parameter übergeben: Alter und Name. Diese Parameter müssen im Konstruktor den entsprechenden Attributen zugeordnet werden.

KLASSE: Passenger : Person

Attribute:

`ticket`	Objekt vom Typ Ticket
`luggage`	Objekt vom Typ Luggage

Konstruktor:

`Passenger()`	Dem Konstruktor werden folgende Parameter übergeben: Name, Alter, Ticket und Gepäck. Diese Parameter müssen im Konstruktor den entsprechenden Attributen zugeordnet werden.

Klassenbeschreibung

KLASSE: Crew : Person

Attribute:

`skill`	Arbeitsbezeichnung als Zahl - (int)
`role`	Rolle des Crewmembers - (string)
`assigned`	Variable, die angibt, ob die Person schon zu einem Flug zuteilt worden ist - (bool)

Konstruktor:

`Crew()` Dem Konstruktor werden folgende Parameter übergeben: Alter, Name, Role. Diese Parameter müssen im Konstruktor den entsprechenden Attributen zugeordnet werden. Zusätzlich soll im Konstruktor je nach role der richtige Skill vergeben werden. `Assigned` ist auf `false` zu setzen

Skill 1 → „pilot“
Skill 2 → „FO“
Skill 3 → „flightattendant“

Methoden:

`GetInformation()` Gibt die Information des Crewmembers in folgender Form wieder:

```
          pilot:      Irmi age 22
             FO:     Franz age 44
flightattendant:      Anna age 58
flightattendant:   Micheal age 41
flightattendant:    Sophia age 51
flightattendant:     Franz age 58
flightattendant:     Franz age 61
```

Klassenbeschreibung

Abstrakte KLASSE: Aircraft

Attribute:

`aircraftWeight`	Flugzeugeigengewicht - (double)
`assigned`	Variable, die angibt, ob Flugzeug schon einem Flug zugeteilt worden ist - (bool)
`crew`	List vom Typ „Crew“
`fuelCapacity`	Treibstoffmenge - (int)
`fuelConsumption`	Treibstoffverbrauch in l/h - (int)
`loadedWeight`	geladenes Gewicht - (double)
`maxSpeed`	max. Geschwindigkeit - (int)
`maxWeight`	max. TakeOff Gewicht - (double)
`model`	Flugzeugmodel (747, 787, A350, A350XLR, ATR-72) - (string)
`type`	Flugzeugtyp (jumbojet, jet, propeller) - (string)

Konstruktor:

`Aircraft()` Dem Konstruktor werden folgende Parameter übergeben: Typ, Model, Treibstoffmenge, max. Geschwindigkeit. Diese Parameter müssen im Konstruktor den entsprechenden Attributen zugeordnet werden. Folgende Tabelle muss im Konstuktor einarbeitet werden:

Flugzeugtyp	fuel Consumption	maxWeight	aircraft Weight	maxPassengers
jumbojet	16000	380000	185000	500
jet	7500	250000	110000	290
propeller	150	23000	6500	75

Klassenbeschreibung

Abstrakte KLASSE: Aircraft - Fortsetzung

Methoden:

`calcFuelConsumption()`	abstakte Methode, die das Attribute `fuelConsumption` zurückgibt
`UpdateWeight()`	virtuelle Methode, die ein Gewicht (double) übergeben bekommt

KLASSE: CargoAircraft : Aircraft

Attribute:

`currentWeight`	Variable für das akutelle Gewicht - (double)

Konstruktor:

`CargoAircraft()`	der Konstruktor ruft den base-Konstruktor auf und berechnet sich das aktuelle Gewicht mit `aircraftWeight` und `loadedWeight`

Methoden:

`calcFuelConsumption()`	überschreibt die abstrakte Methode nach folgender Vorschrift: $fuelConsumption - (200 * flightTimeinh)$
`UpdateWeight()`	überschreibt die virtuelle Methode, wobei das `currentWeight` um das übergegebene Gewicht erhöht wird

Klassenbeschreibung

KLASSE: PassengerAircraft : Aircraft

Attribute:

`currentWeight`	Variable für das akutelle Gewicht - (double)
`maxPassengers`	Anzahl der max. Passagieranzahl - (int)

Konstruktor:

`PassengerAircraft()`	Dem Konstruktor wird je nach Flugzeugtyp die maximale Passagieranzahl gesetzt, siehe Tabelle auf Folie 15 und berechnet sich das aktuelle Gewicht mit `aircraftWeight` und `loadedWeight`

Methoden:

`UpdateWeight()`	überschreibt die virtuelle Methode, wobei das `currentWeight` um das übergegebene Gewicht erhöht wird
`AddLuggage()`	überprüft, für jedes der übergebenen Gepäcksstücke aus der List, ob das maximale Gewicht erreicht ist, falls nicht kann dieses Gepäcksstück geladen werden (`loadedWeight` erhöht sich)
`AssignCrewToAircraft()`	bekommt eine Liste an Crewmembers und setzt das entsprechende Attribut
`walkAround()`	Konsolenausgabe, dass der WalkAround abgeschlossen ist.
`CompareTo()`	`Imcompareable`-Interface: vergleicht zwei Passagierflugzeuge miteinander und sortiert das mit der geringeren maximal Passagieranzahl nach vorne.
`calcFuelConsumption()`	überschreibt die abstrakte Methode nach folgender Vorschrift: $fuelConsumption - (flightTimeinh * 0.97 * maxPassengers)$
`SetFuelCapacity()`	bekommt `fuelNeeded` übergeben und setzt das Attribut `fuelCapacity`

Klassenbeschreibung

KLASSE: Flight

Attribute:

`actualDepartureTime`	wirkliche TakeOff Zeit - (int)
`addLuggage`	Variable, die angibt, ob das Gepäck schon fertig geladen ist - (bool)
`aircraft`	Flugzeug, welches dem Flug zugeordnet wird vom Typ PassengerAircraft
`boardingTime`	benötigte Zeit für das Boarden der Passagiere - (int)
`boardPassengers`	Variable, die angibt, ob die Passagiere schon fertig geboardet sind - (bool)
`briefing`	Variable, die angibt, ob die Crew schon den Flug besprochen hat - (bool)
`briefingTime`	benötigte Zeit für das Briefen der Crew - (int)
`clearance`	Variable, die angibt, ob das Flugzeug abheben darf - (bool)
`crew`	List vom Typ „Crew“, die dem Flug zugeteilt worden sind
`currentTime`	aktueller Zeitpunkt - (int)
`departureTime`	geplante TakeOff Zeit des Fluges als - (int)
`destination`	Zielflughafen - (string)
`destinationLat`	Latitude des Zielflughafens - (double)
`destinationLong`	Longitude des Zielflughafens - (double)
`flightNumber`	Flugnummer - (string)
`passengers`	zugehörige Passagiere als Liste vom Typ Passager
`soldTickets`	Anzahl der verkauften Tickets des Flugzugs - (int)
`status`	Flugstatus als Enum vom Typ FlightStatus
`walkAround`	Variable, die angibt, ob der Walkaround schon durchgeführt worden ist - (bool)
`walkAroundTime`	benötigte Zeit für den walkAround (abhängig vom Flugzeugtyp) - (int)

Klassenbeschreibung

KLASSE: Flight-Fortsetzung

Konstruktor:

`Flight()` Dem Konstruktor werden folgende Parameter übergeben: Flugnummer, DepatureTime, soldtickets, Zielflughafen, Latitude & Longitude des Zielflughafens. Hier müssen die bool Variablen: `briefing`, `walkAround`, `cleareance`, `boardPassengers` und `addLuggage` auf false gesetzt werden.

Methoden:

`AssignCrewToFlight()` sucht aus der übergebenen List von Crew die Besatzung aus, die frei ist und benötigt wird. Genauer Ablauf siehe Flowchart Folie 12

`BoardPassengers()` bekommt eine List von Passagieren übergeben, setzt das entsprechende Attribut und berechnet anhand der Anzahl der Passagiere die `boardingTime` mittels folgender Formel:

$$boardingTime = passengercount * 0.1 + 2$$

ACHTUNG: für Jumbojets werden zum Boarden zwei Eingänge verwendet, hier muss die Formel angepasst werden!

`BoardLuggage()` ruft die Methode `aircraft.AddLuggage()`, setzt `addLuggage` auf `true`

`CalcDistance()` berechnet mit den übergebenen Koordinaten die Distanz in km zwischen den Orten mittels folgender Formel: wobei r = 6371 km (Erdradius) und die übergegben Koordinaten in Radianten zu übergeben sind

$$distance = r \cdot Acos(\sin(lat1) \cdot \sin(lat2) + \cos(lat1) \cdot \cos(lat2) \cdot \cos(long2 - long1))$$

`CheckAvialableAircraft()` sucht aus der übergebenen List von Flugzeugen ein Flugzeug aus, in dem am wenigsten leere Plätze übrig bleiben, aber trotzdem alle Passagiere mit Ticket boarden können.

Klassenbeschreibung

KLASSE: Flight-Fortsetzung

`Methoden:`

`checkBriefing()`	so lange die Zeit für das Briefing (25 Minuten) noch nicht verstrichen ist, wird gewartet, bis die Endzeit des Briefings und die currentTime aus der Simulation gleich sind, erst dann kann der nächste Status gesetzt werden. Hier erfolgt auch die Ausgabe von crew.briefing()
`checkWalkAround()`	gleich die wie bei checkBriefing(): es wird so lange gewartet, die die Endzeit des WalkAround die currentTime der Simulation ist und erst dann kann der nächste Status gesetzt werden.

Achtung: Beide Methoden dürfen nur ein einziges Mal die Endzeit berechnen: Arbeiten Sie hier mit den bool Attributen briefing und walkAround!

Klassenbeschreibung

KLASSE: Flight-Fortsetzung

Methoden:

`CompareTo()` `Imcompareable`-Interface: vergleicht die DeparetureTime von zwei Flugzeugen

`FlightBriefing()` berechnet die Distanz, setzt die benötigte Treibstoffmenge und checkt, ob die 25 Minuten für briefing schon vorbei sind um den Status `Preparing` zu setzten

`PrepareFlight()` checkt, ob der WalkAround schon durchgeführt worden ist und setzt dann den Status `Boarding`

`GetClearance()` checkt, ob das Gepäck verlanden und die Passagiere geboardet sind (`addLuggage` & `boardPassengers`) und setzt dementsprechend das Attribut

`GetWalkAroundTime()` gibt die jeweilige Zeit für den WalkAround zurück:

Flugzeugtyp	WalkaroundTime [in min]
Jumbojet	15
Jet	10
Propeller	5

`CalcFlightTime()` berechnet die Flugdauer in Minuten laut:

$$flightTime = \frac{distance}{maxspeed * 0.8}$$

`CalcFuelNeeded()` berechnet die benötigte Treibstoffmenge:

$$fuelNeeded = (flightTime\ [in\ h] + 2.5) * fuelConsumption$$

Klassenbeschreibung

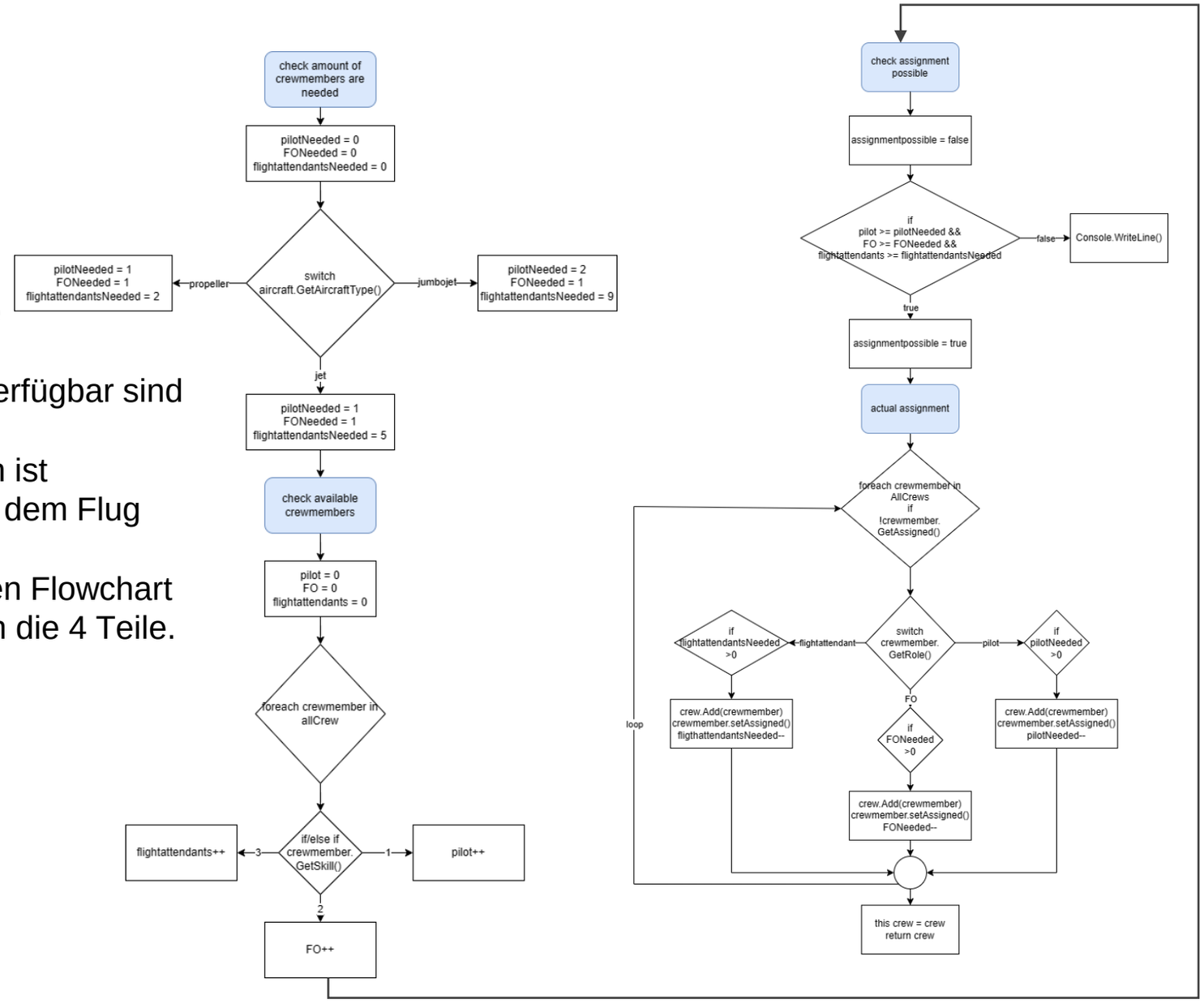
KLASSE: Flight - Fortsetzung

`FlowChart:`
`AssignCrewToFlight()`

Die Methode besteht aus hier 4 Teilen:

1. Zählen wie viele Besatzungsmitglieder je Flugzeugtyp benötigt werden
2. Zählen wie viele Besatzungsmitglieder verfügbar sind `(assigned == false)`
3. Prüfen, ob Zuweisung überhaupt möglich ist
4. Zuweisen von Besatzungsmitgliedern zu dem Flug

Der genauere Ablauf ist dem nebenstehenden Flowchart zu entnehmen. Die farbigen Zellen markieren die 4 Teile.



Klassenbeschreibung

KLASSE: Airport

In der Klasse „Airport“ sollen die Aufgaben, die für einen Flug benötigt werden, durchgeführt werden (DoSimulation). Die Kapselung der einzelnen Methoden soll eine bessere Übersicht im Programm ermöglichen.

Attribute:

`aircrafts`	Liste vom Typ „PassengerAircraft“
`completedImport`	Variable, die nach erfolgreichem Import auf „true“ gesetzt wird - (bool)
`completedSimulation`	Variable, die nach erfolgreicher Simulation auf „true“ gesetzt wird - (bool)
`crews`	Liste vom Typ „Crew“
`fligths`	Liste vom Typ „Flight“
`latcoordinate`	Latitude Koordinates des Flughafens - (double)
`longcoordinate`	Longitude Koordinates des Flughafens - (double)
`location`	Standort des Flughafens - (string)
`luggages`	Liste vom Typ „Luggage“
`name`	Namen des Flughafens - (string)
`passengers`	Liste vom Typ „Passengers“
`path`	Speicherpfad der Input-Dateien (Ordner) - (string)

Konstruktor:

`Airport()`	Dem Konstruktor wird der Speicherpfad der CSV-Dateien übergeben, der Name, Location und Koordinaten (46,99 | 15,43) werden direkt gesetzt

Methoden:

`ReadInputFiles()`	ruft die Methoden: ReadFlights(), ReadAircrafts(), ReadCrew() und ReadPassengers() auf und speicherte diese in den jeweiligen Attributen.

Klassenbeschreibung

KLASSE: Airport - Fortsetzung

Methoden:

`ReadFlights()`	private Methode zum Einlesen der Flüge, bekommen der Dateiname als path übergeben
`ReadAircrafts()`	private Methode zum Einlesen der Flugzeuge, , bekommen der Dateiname als path übergeben
`ReadCrew()`	private Methode zum Einlesen der Besatzungsmitglieder, bekommen der Dateiname als path übergeben
`ReadPassengers()`	private Methode zum Einlesen von Passagieren mit deren Tickets und Gepäck, bekommen der Dateiname als path übergeben
`DoSimulation()`	public Methode zur Ausführung von anderen Methoden (siehe Folie 20: Flowchart)
`CalcTimeToClock()`	private Methode zum Umrechnen von Minuten in eine Uhrzeit für schönere Ausgabe
`SortAircraftsBySeats()`	private Methode, die `aircrafts.Sort()` aufruft, die aufsteigend vom Minimum zum Maximum der Platzanzahl sortiert
`SortFlightsByDepartureTime()`	private Methode, die `flights.Sort()` aufruft, die aufsteigend vom frühsten bis zum spätetesten Flug sortiert

Klassenbeschreibung

KLASSE: Airport - Fortsetzung

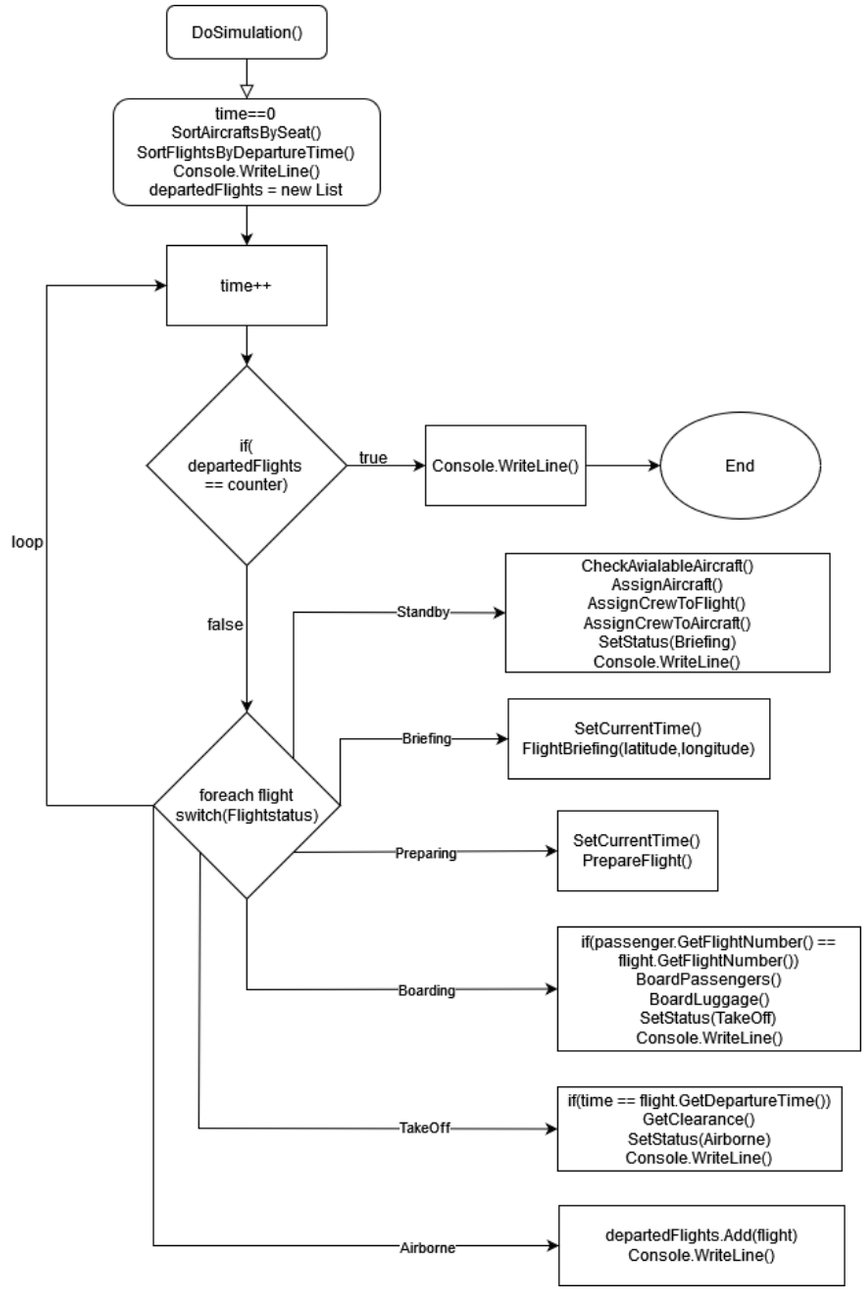
`Flowchart:`
`DoSimulation()`

Diese Methode beinhaltet die Abarbeitung der Flugliste. Dabei stellt ein Durchlauf der Schleife eine Minute dar. Die Schleife wird beendet, wenn alle Flüge abgearbeitet wurden.

Als erstes ist ein passendes Flugzeug mit benötigter Besatzung für den Flug zu finden. Die Crew bereitet den Flug vor (Distanz berechnen, Treibstoffmenge bestimmen, Briefing, WalkAround), bevor die jeweiligen Passagiere boarden und deren Gepäck verladen wird. Abschließend sind letzte Checks durchzuführen und das Flugzeug kann abheben.

Die verschiedenen Zustände der Einsätze werden im Attribut „FlightStatus“ berücksichtigt.

Crew Assignment: (siehe Flowchart auf Folie 16).



Konsolenausgabe:

```
Welcome to MBI Airport! Let's check out todays Schedule:
Flight MBI23 is scheduled for departure from Graz to        London at 07:15
Flight  II24 is scheduled for departure from Graz to         Miami at 09:35
Flight MBI24 is scheduled for departure from Graz to      Istanbul at 12:05
Flight II234 is scheduled for departure from Graz to     Amsterdam at 15:15
Flight  II23 is scheduled for departure from Graz to         Tokio at 17:45
-----------------------------------------------------------------------------------------


05:45: Flight MBI23 is preparing the flight with following crew scheduled at 07:15
                    FO:     Daniel age 63
                 pilot:      Franz age 29
        flightattendant:      David age 51
        flightattendant:     Selina age 43
        flightattendant:       Liam age 22
        flightattendant:     Selina age 30
        flightattendant:       Liam age 63
        Using the Boeing 787 with approximatly 186 Passengers and the Destination London


05:46:  Flight MBI23
               Crew member    Daniel (            FO) is at the briefing
               Crew member     Franz (         pilot) is at the briefing
               Crew member     David (flightattendant) is at the briefing
               Crew member    Selina (flightattendant) is at the briefing
               Crew member      Liam (flightattendant) is at the briefing
               Crew member    Selina (flightattendant) is at the briefing
               Crew member      Liam (flightattendant) is at the briefing
06:12:  Flight MBI23
               Walkaround on Aircraft Boeing 787 is completed
06:22:  Flight MBI23 is starts boarding
               Boeing 787: current weight: 113383.6 kg (weight:maxTakeoffRatio: 45.35 %)
06:42: Flight MBI23 has finished boarding the 186 Passengers and waits for departure
07:15: Flight MBI23 on the taxiway and on its way to take off
07:23: Flight MBI23 is the air with 186 passengers on board, flying from Graz to London (1194.15 km)
       with a flighttime: 02:38 h
-----------------------------------------------------------------------------------------


08:05: Flight II24 is preparing the flight with following crew scheduled at 09:35
                 pilot:       Irmi age 22
                    FO:      Franz age 44
        flightattendant:       Anna age 58
        flightattendant:    Micheal age 41
        flightattendant:     Sophia age 51
        flightattendant:      Franz age 58
        flightattendant:      Franz age 61
        Using the A350 with approximatly 273 Passengers and the Destination Miami
```

```
14:12:  Flight II234
               Walkaround on Aircraft ATR-72 is completed
14:17:  Flight II234 is starts boarding
               ATR-72: current weight: 7124.2 kg (weight:maxTakeoffRatio: 30.97 %)
14:22: Flight II234 has finished boarding the 35 Passengers and waits for departure
15:15: Flight II234 on the taxiway and on its way to take off
15:17: Flight II234 is the air with 35 passengers on board, flying from Graz to Amsterdam (966.16 km)
       with a flighttime: 02:19 h
-----------------------------------------------------------------------------------------


16:15: Flight II23 is preparing the flight with following crew scheduled at 17:45
                 pilot:  Alexander age 26
                    FO:      Franz age 43
                 pilot:  Alexander age 56
        flightattendant:     Sophia age 43
        flightattendant:       Liam age 40
        flightattendant:       Irmi age 58
        flightattendant:       Liam age 39
        flightattendant:     Daniel age 22
        flightattendant:      Julia age 34
        flightattendant:    Micheal age 46
        flightattendant:     Sophia age 33
        flightattendant:      Sarah age 23
        Using the Boeing 747-400 with approximatly 435 Passengers and the Destination Tokio


16:16:  Flight II23
               Crew member Alexander (          pilot) is at the briefing
               Crew member     Franz (            FO) is at the briefing
               Crew member Alexander (          pilot) is at the briefing
               Crew member    Sophia (flightattendant) is at the briefing
               Crew member      Liam (flightattendant) is at the briefing
               Crew member      Irmi (flightattendant) is at the briefing
               Crew member      Liam (flightattendant) is at the briefing
               Crew member    Daniel (flightattendant) is at the briefing
               Crew member     Julia (flightattendant) is at the briefing
               Crew member   Micheal (flightattendant) is at the briefing
               Crew member    Sophia (flightattendant) is at the briefing
               Crew member     Sarah (flightattendant) is at the briefing
16:42:  Flight II23
               Walkaround on Aircraft Boeing 747-400 is completed
16:57:  Flight II23 is starts boarding
               Boeing 747-400: current weight: 192796.1 kg (weight:maxTakeoffRatio: 50.74 %)
17:20: Flight II23 has finished boarding the 435 Passengers and waits for departure
17:45: Flight II23 on the taxiway and on its way to take off
17:47: Flight II23 is the air with 435 passengers on board, flying from Graz to Tokio (9292.99 km)
       with a flighttime: 13:46 h
-----------------------------------------------------------------------------------------

Todays schedule at MBI Airport in Graz is finished! Thank you!
```