

# Projekt zaliczeniowy



Programowanie obiektowe  
Rok akademicki 2024/2025

## Autorzy:

Piotr Otręba  
Karol Odój  
Maximilian Szemik  
Aleksander Potok

## Lotnisko

Nasz projekt jest symulacją lotniska w Krakowie. Obsługuje on przyloty i odloty, każdy samolot jest przypisany do konkretnego lotu, posiada nazwę, określone parametry związane z pojemnością baku, ilością pasażerów czy maksymalną wagą.

## Podział ról

Piotr Otręba: Klasa Bagaż, Bilet, Samolot, Pasażer, Lot

Karol Odój: Klasa Lotnisko, Program, FazaLotu, Sprawozdanie

Maximilian Szemik: Diagram klas, Testy,

Aleksander Potok: GUI

## Klasa Bramka

### Opis:

Klasa reprezentuje bramkę na lotnisku, do której mogą być przypisywane samoloty. Zawiera podstawowe informacje, takie jak numer bramki, status zajętości oraz kategoria bramki (od A do D, gdzie A to najmniejsza bramka, a D to największa).

### Rola w projekcie:

Bramki są niezbędne do zarządzania ruchem samolotów na lotnisku. Każdy samolot potrzebuje odpowiednio dopasowanej bramki (pod względem wielkości) podczas obsługi pasażerów i bagażu.

### Uzasadnienie modyfikatorów dostępu:

- **numer** – public: jest to kluczowe pole, które musi być dostępne poza klasą do identyfikacji bramki.
  - **czyZajety** – public: dostępny publicznie, ponieważ wiele operacji zewnętrznych musi wiedzieć, czy bramka jest zajęta.
  - **kategoria** – public: informacja o kategorii bramki musi być łatwo dostępna podczas przypisywania samolotów.
- 

## Klasa AktualnyCzas

### Opis:

Reprezentuje aktualny czas w symulacji. Wartość czasu jest przechowywana jako statyczne pole, co pozwala na globalny dostęp w całym programie.

### Rola w projekcie:

Służy do symulacji upływu czasu na lotnisku. Kluczowe dla monitorowania czasu przylotu, odlotu i innych operacji związanych z samolotami.

### Uzasadnienie modyfikatorów dostępu:

- **aktualnyCzas** – public static: globalny dostęp umożliwia użycie tej wartości w różnych klasach, bez potrzeby tworzenia instancji klasy.
-

## **Klasa Lotnisko**

### **Opis:**

Reprezentuje lotnisko, które zarządza bramkami, samolotami oraz historią lotów. Klasa obejmuje funkcjonalności dodawania, sortowania oraz obsługi przylotów i odlotów.

### **Rola w projekcie:**

Stanowi główny punkt zarządzania ruchem lotniczym. Odpowiada za przydzielanie bramek, zarządzanie przylotami i odlotami, a także zapis i odczyt historii lotów.

### **Uzasadnienie modyfikatorów dostępu:**

- **nazwa** – public: nazwa lotniska jest kluczowym elementem identyfikacyjnym i może być używana w wielu miejscach.
- **bramki, odloty, przyloty, historia** – private: te listy powinny być zarządzane wyłącznie wewnątrz klasy, aby zapewnić integralność danych.
- **Właściwości Przyloty i Odloty** – public: umożliwiają bezpieczny dostęp do list poprzez gettery i settery.

### **Dodatkowe funkcjonalności:**

- **DodajSamolot** – umożliwia dodanie samolotu do listy przylotów.
- **SortujLotyPoGodzinieWylotu** – sortuje listy lotów po godzinach przylotów i odlotów.
- **WyświetlLoty** – wyświetla aktualne przyloty, odloty oraz historię lotów.
- **LosujSamoloty** – symuluje generowanie losowych samolotów na potrzeby przylotów.
- **RuchCzasu** – symuluje upływ czasu, aktualizując stany lotów i czas.

### **Uzasadnienie dla funkcji:**

- Funkcje takie jak **LosujSamoloty** i **RuchCzasu** są kluczowe dla symulacji operacji na lotnisku, umożliwiając dynamiczne zarządzanie stanem lotów i przypisywanie zasobów.
-

## Klasa Program

### Opis:

Jest to klasa główna, zawierająca metodę Main, która stanowi punkt startowy programu.

### Rola w projekcie:

Symuluje działanie lotniska poprzez wywoływanie metod klas zarządzających (np. Lotnisko). Służy do uruchamiania symulacji, wyświetlania lotów.

---

## Klasa BlednaGodzinaException

### Opis:

Wyjątek niestandardowy, używany do obsługi błędów związanych z nieprawidłową godziną lotu.

### Rola w projekcie:

Zwiększa czytelność i bezpieczeństwo kodu poprzez dedykowaną obsługę błędów dla godzin w klasie Lot.

### Uzasadnienie modyfikatorów dostępu:

- **BlednaGodzinaException** – public: dostępna globalnie, aby mogła być używana w całym projekcie.
- 

## Klasa Lot

### Opis:

Reprezentuje informacje o locie, takie jak godzina przylotu/wylotu, faza lotu, miasto docelowe, czas lotu oraz opóźnienia.

### Rola w projekcie:

Zarządza danymi i logiką dotyczącą pojedynczego lotu. Jest podstawowym budulcem list przylotów, odlotów i historii w klasie Lotnisko.

### Uzasadnienie modyfikatorów dostępu:

- **Pola: godzinaWylotu, godzinaPrzylotu, czasDoOdlotu, opoznienie** – public: muszą być dostępne w całym projekcie, aby inne klasy mogły odczytywać i modyfikować te informacje.
  - **bramka** – public: lot musi być powiązany z konkretną bramką, co wymaga dostępu globalnego.
  - **czasNaLotniku** – public: pozwala określić czas, jaki samolot powinien spędzić na lotniku.
-

## Klasa BlednaWagaBagazuException

### Opis:

Wyjątek niestandardowy, używany do obsługi błędów związanych z nieprawidłową wagą bagażu.

### Rola w projekcie:

Zapewnia bezpieczeństwo i walidację danych w klasie Bagaz.

### Uzasadnienie modyfikatorów dostępu:

- **BlednaWagaBagazuException** – public: wyjątek jest dostępny globalnie.
- 

## Klasa Bagaz

### Opis:

Reprezentuje bagaż pasażera, zawierając informacje o jego wadze i rodzaju (podręczny lub rejestrowany).

### Rola w projekcie:

Modeluje bagaże, które są przypisywane do pasażerów w trakcie boardingu. Pozwala na walidację wagi bagażu.

### Uzasadnienie modyfikatorów dostępu:

- **waga** – private: waga bagażu jest dostępna tylko za pośrednictwem właściwości Waga, aby zapewnić walidację danych.
- **rodzaj** – public: rodzaj bagażu jest podstawową informacją, która musi być dostępna globalnie.

### Funkcjonalności:

- **Waga (właściwość)**: umożliwia sprawdzenie i ustawienie wagi bagażu. W przypadku ujemnej wartości rzuca wyjątek BlednaWagaBagazuException.
  - **ToString()**: zwraca tekstową reprezentację bagażu.
- 

## Klasa Bilet

### Opis:

Reprezentuje bilet lotniczy, zawierający informacje o klasie, cenie i przypisanym locie.

### Rola w projekcie:

Bilet łączy pasażera z lotem i określa warunki podróży, takie jak klasa podróży i cena. Jest używany do sortowania pasażerów w procesie boardingu oraz do sprawdzania równowagi biletów.

### Uzasadnienie modyfikatorów dostępu:

- **klasa** – public: niezbędna do łatwego porównywania biletów oraz dla operacji związanych z klasą podróży.
- **cena** – private: cena biletu jest wewnętrznym szczegółem i powinna być kontrolowana tylko wewnątrz klasy.

- **lot** – private: informacje o locie są dostępne tylko wewnątrz klasy, gdyż to bilet wiąże pasażera z konkretnym lotem.

#### **Implementowane interfejsy:**

- **Comparable<Bilet>** – umożliwia sortowanie biletów według klasy i ceny.
- **Equatable<Bilet>** – pozwala sprawdzić, czy dwa bilety są równoważne (mają tę samą klasę i cenę).

---

### **Klasa Samolot (Abstrakcyjna)**

#### **Opis:**

Stanowi ogólny model samolotu. Klasa ta jest podstawą dla specyficznych modeli, takich jak Airbus320, Boeing737, itp.

#### **Rola w projekcie:**

Zarządza pasażerami, bagażami, paliwem oraz logistyką związanych z lotem. Abstrakcja pozwala na tworzenie różnych typów samolotów z unikalnymi parametrami, które dziedziczą wspólną funkcjonalność.

#### **Uzasadnienie modyfikatorów dostępu:**

- **model, wagaPustegoSamolotu, maxLiczbaPasazerow, maxPaliwo, maxLacznaWaga, liniaLotnicza, spalaniePaliwa, kategoria** – public: potrzebne do obsługi logiki lotów, tankowania, boardingu i innych operacji.
- **stanPaliwa, pasazerowie, bagaze** – private: te dane powinny być zarządzane wyłącznie przez metody i właściwości klasy, aby zapewnić spójność.

#### **Kluczowe funkcjonalności:**

- **LosujPasazerow()** – generuje pasażerów z losowymi biletami i bagażami.
- **Tankowanie()** – oblicza i uzupełnia paliwo zgodnie z potrzebami lotu.
- **Boarding()** – zarządza pasażerami wchodzącymi na pokład, w tym obsługuje sytuacje overbookingu.
- **WagaCalkowita()** – oblicza łączną wagę samolotu, uwzględniając pasażerów, bagaże i paliwo.
- **ZaladunekBagazy()** – zarządza bagażami pasażerów, uwzględniając limity wagowe.

#### **Dziedziczenie:**

Każdy model samolotu (np. Airbus320, Boeing737) dziedziczy z klasy Samolot i definiuje specyficzne parametry, takie jak maksymalna liczba pasażerów, pojemność paliwa, czy kategoria.

---

## Klasy dziedziczące po Samolot

**Przykłady:** Airbus320, Boeing737, Embrayer195, Airbus380

### Opis:

Każda klasa reprezentuje konkretny model samolotu z określonymi parametrami technicznymi.

### Rola w projekcie:

Zapewniają różnorodność w symulacji lotniska, umożliwiając przypisanie różnych typów samolotów do lotów w zależności od potrzeb (np. pojemności pasażerskiej, spalania paliwa).

---

## Klasa Pasazer

### Opis:

Reprezentuje pasażera podróżującego samolotem. Zawiera informacje o imieniu, nazwisku, dacie urodzenia, bilecie oraz bagażach.

### Rola w projekcie:

Jest podstawowym elementem zarządzania pasażerami w symulacji. Obsługuje bagaże oraz proces boardingu.

### Uzasadnienie modyfikatorów dostępu:

- **imie, nazwisko, dataUrodzenia** – private: dane pasażera są dostępne tylko wewnątrz klasy, aby zapewnić ich integralność.
- **bilet, bagaze** – public: bilet i bagaże są dostępne dla innych klas, ponieważ są kluczowe dla logiki boardingu.

### Implementowane interfejsy:

- **IComparable<Pasazer>** – umożliwia sortowanie pasażerów, np. według klasy biletu lub daty urodzenia.
- 

## Klasa LosowanieCzasu (Statyczna)

### Opis:

Klasa zawiera metodę do losowania czasu w postaci obiektu TimeSpan w określonym przedziale.

### Rola w projekcie:

Jest wykorzystywana do generowania losowych czasów (np. przylotu lub odlotu samolotów), co dodaje realizmu w symulacji.

### Uzasadnienie modyfikatorów dostępu:

- **LosujCzasMinuty** – public static: metoda jest statyczna, ponieważ operacja losowania czasu jest niezależna od instancji klasy. Publiczny dostęp pozwala na jej użycie w innych częściach projektu.

**Walidacja:**

- Sprawdza, czy minimalny czas nie jest większy od maksymalnego. W przeciwnym razie rzuca wyjątek `ArgumentException`.
- 

**Klasa `LiniaLotnicza`****Opis:**

Reprezentuje linię lotniczą, definiującą szczegóły dotyczące bagażu oraz statusu jako tania linia.

**Rola w projekcie:**

Linia lotnicza wpływa na limity bagażu pasażerów, ceny biletów oraz sposób obsługi pasażerów w procesie boardingu i rezerwacji.

**Uzasadnienie modyfikatorów dostępu:**

- **`nazwa`, `maxWagaBagazuPodrecznego`, `maxWagaBagazuRejestrowanego`, `czyTaniaLinia`** – public: dane te muszą być dostępne w innych klasach, takich jak `Samolot` czy `Pasazer`, do obsługi logiki bagażowej i cenowej.

**Funkcjonalności:**

- Konstruktor przyjmuje wszystkie niezbędne parametry, co umożliwia łatwe tworzenie instancji z różnymi liniami lotniczymi.
  - Atrybut `czyTaniaLinia` wpływa na generowanie pasażerów i biletów (np. w klasie `Samolot`).
- 

**Klasa `ZlaKategoriaStanowiskaException`****Opis:**

Wyjątek niestandardowy, używany do obsługi błędów związanych z nieprawidłową kategorią stanowiska (np. bramki lub pasa startowego).

**Rola w projekcie:**

Zwiększa bezpieczeństwo kodu i obsługę wyjątków w sytuacjach, gdy zasób (np. bramka lub pas) jest przypisany do nieodpowiedniego samolotu.

**Uzasadnienie modyfikatorów dostępu:**

- **`ZlaKategoriaStanowiskaException`** – public: musi być dostępna globalnie, aby umożliwić rzucanie i obsługę w różnych częściach projektu.



[illegible]

Aplikacja umożliwia zarządzanie ruchem lotniczym na lotnisku.

- ## 1. Główne okno aplikacji, posiada takie funkcjonalności jak:
- **Zarządzanie lotami:**
    - Wyświetlanie listy przylotów i odlotów w kontrolkach.
    - Prezentowanie szczegółów wybranego lotu w polu tekstowym
  - **Aktualizacja czasu:**
    - Symulacja upływu czasu na lotnisku po naciśnięciu przycisku "Przesuń czas"
    - Automatyczna aktualizacja listy lotów oraz czasu w widoku.
  - **Wyświetlanie szczegółów lotu:**
    - Po kliknięciu na przylot/odlot użytkownik może otworzyć nowe okno z bardziej szczegółowymi informacjami o wybranym samolocie i pasażerach.
  - **Przekierowanie lotu:**
    - Przycisk "Przekieruj do Katowic" pozwala na usunięcie lotu z listy przylotów, pod warunkiem, że samolot nie wylądował jeszcze na lotnisku.
  - **Reset widoku:**

- Przycisk "Wyczyść" pozwala wyczyścić szczegóły lotów i odświeżyć listę.
- 

## **2. Okno szczegółów posiada takie funkcjonalności jak:**

- **Lista pasażerów:**
    - Wyświetlanie listy pasażerów przypisanych do samolotu. W przypadku braku pasażerów wyświetla komunikat "Brak pasażerów".
  - **Usuwanie pasażera:**
    - Przycisk "Usuń pasażera" pozwala na usunięcie wybranego pasażera z samolotu. Po usunięciu lista pasażerów jest automatycznie aktualizowana.
  - **Dodawanie pasażera:**
    - Przycisk "Dodaj pasażera" otwiera nowe okno (DodajPaszeraWindow), umożliwiające dodanie pasażera wraz z jego bagażem do samolotu.
    - W przypadku, gdy liczba pasażerów osiągnie maksymalną pojemność samolotu, wyświetlany jest odpowiedni komunikat.
- 

## **3. Okno dodawania pasażera posiada takie funkcjonalności jak:**

- **Dodawanie pasażera:**
    - Umożliwia wprowadzenie imienia, nazwiska, daty urodzenia, klasy biletu oraz ceny biletu dla nowego pasażera.
    - Walidacja danych wejściowych, takich jak poprawność daty urodzenia i ceny biletu.
    - Wyświetlenie ostrzeżenia, jeśli pasażer nie ma bagażu.
  - **Dodawanie i usuwanie bagaży:**
    - Użytkownik może dodawać bagaże (podręczne i rejestrowane) wraz z ich wagą.
    - Sprawdzane są limity wagowe dla bagażu, zależne od linii lotniczej.
    - Przycisk "Usuń bagaż" pozwala na usunięcie wybranego bagażu z listy.
  - **Automatyczna aktualizacja ceny biletu:**
    - Cena biletu jest dynamicznie aktualizowana w zależności od klasy podróży oraz liczby i wagi bagaży.
- 

## **Dodatkowe informacje:**

- **Walidacja i obsługa błędów:**
  - Wszystkie krytyczne dane wejściowe, takie jak waga bagażu, data urodzenia czy cena biletu, są weryfikowane przed zatwierdzeniem.
  - W przypadku błędnych danych lub nieprawidłowych operacji (np. dodanie pasażera do pełnego samolotu), użytkownik otrzymuje odpowiednie komunikaty.

- **Interakcja z danymi domenowymi:**
  - Aplikacja korzysta z obiektów takich jak Samolot, Pasazer, Bagaz, Lot, oraz Bilet, co umożliwia efektywne zarządzanie symulacją lotniska.