DOCUMENT DE DESIGN

PROIECT

Monitorizarea altitudinii unui avion

REALIZATOR(I)

Budulan Mihai

VERIFICATOR(I)

Draga Marius

Draga Mihaela

VERSIUNE CURENTĂ

1.1

DATA ULTIMEI VERSIUNI

26.01.2019

CUPRINS

1. Versiunile documentului

Versiunea	Data realizării versiunii	Descriere
1.0	25.01.2019	Prima versiune a documentului de design.
1.1	26.01.2019	Actualizare matrice de trasabilitate

2. Scopul documentului

Documentul a fost realizat de echipa formată din:

- Budulan Mihai
- Draga Marius
- Draga Mihaela

în cadrul proiectului de la disciplina „Sisteme Informatice Critice” Acest document prezintă design-ul pentru proiectul „Monitorizarea altitudinii unui avion”.

3. Documente asociate

3.1.Documente aplicabile

Identificator	Document
AD1	-
AD2	Document de Specificații,

3.2.Documente referință

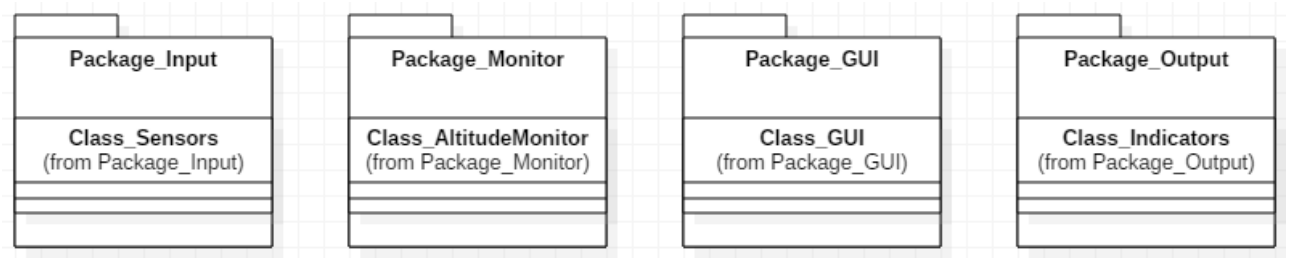
Identificator	Document
RD1	http://staruml.io/
RD2	https://www.smartdraw.com/uml-diagram/

4. Abrevieri

Abreviere	Semnificație
GUI	Graphical User Interface
SIC	Sistem Informatic Critic

5. Design

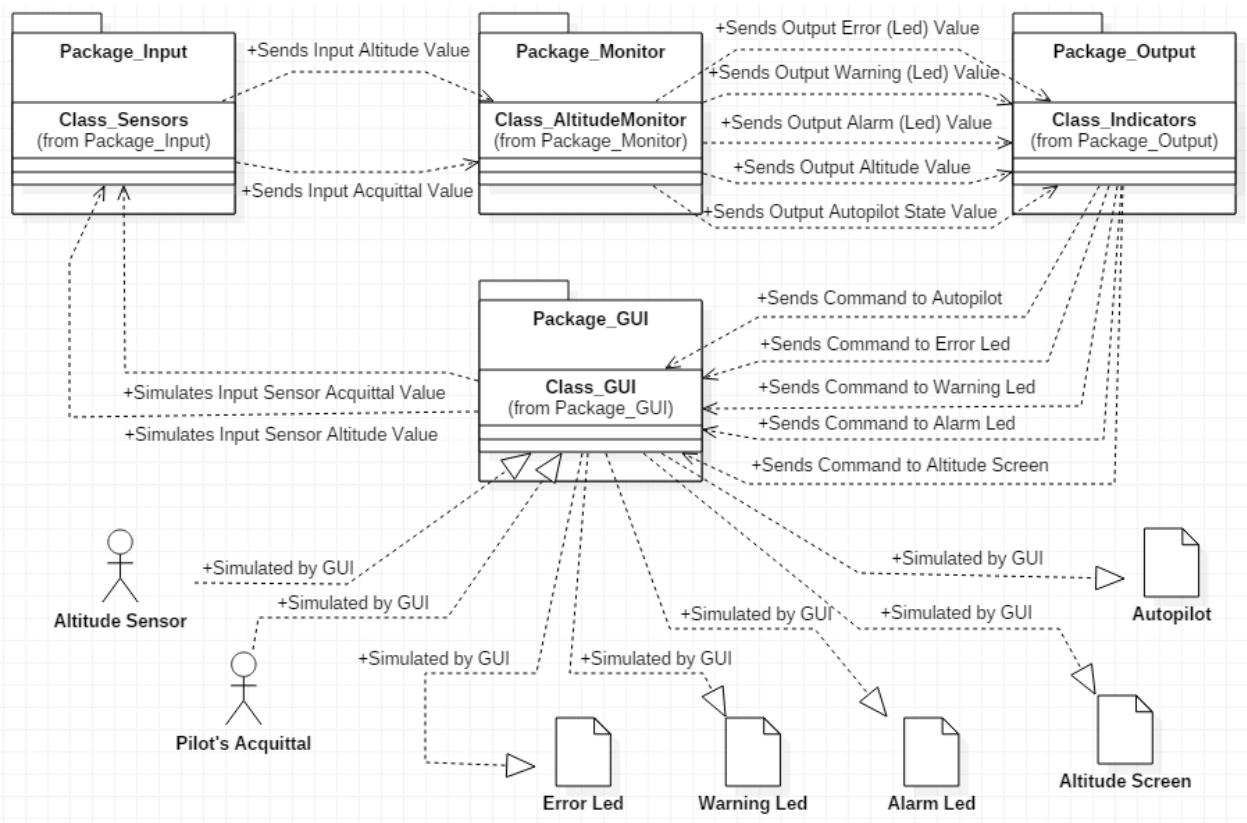
5.1. Arhitectură statică



Pachet	Descriere pachet
Package_Input	Se ocupă cu preluarea/colectarea valorilor intrărilor în SIC. Intrările în sistem sunt: - Altitudinea avionului citită de la senzorul de altitudine. - Butonul de achitare acționat de pilot al semnalului Warning. Conține clasa Class_Sensors.
Package_Monitor	Se ocupă cu monitorizarea altitudinii unui avion: - Preia valorile de intrare prin intermediul Package_Input. - Prelucrează informațiile, calculând comenzile de ieșire. - Transmite valorile de ieșire către Package_Output. Conține clasa Class_AltitudeMonitor.
Package_GUI	Se ocupă cu realizarea unui GUI pentru: - Simularea intrărilor în sistem utilizate de Package_Input. - Vizualizarea ieșirilor din sistem transmise de Package_Output. Conține clasa Class_GUI.

Package_Output	Se ocupă cu transmiterea comenzilor ieșirilor din SIC. Ieșirile din sistem sunt: - LED-urile corespunzătoare semnalelor „Error”, „Warning”, „Alarm”. - Afișarea altitudinii pe un ecran de tip ceas. - Afișarea stării pilotului automat: activat / dezactivat. Conține clasa Class_Indicators.

5.2. Arhitectură dinamică



Package_GUI permite simularea atât a senzorilor de altitudine și de achitare de la intrarea în sistem, cât și simularea indicatoarelor (LED-uri, ecran cu ceas, autopilot) de la ieșirea sistemului.

Valorile simulate ale senzorilor sunt colectate de Package_Input care poate să prelucreze aceste valori dacă este necesar înainte să le transmită mai departe către Package_Monitor care calculează comenzile pentru LED-urile Error, Warning și Alarm, pentru afișorul altitudinii și pentru autopilot.

Comenzile sunt primite de Package_Output care va transmite efectiv comanda către indicatoarele simulate de Package_GUI.

6. Design detaliat

6.1. Package_Input

6.1.1. Descriere

Se ocupă cu preluarea/colectarea valorilor intrărilor în SIC. Intrările în sistem sunt:

- Altitudinea avionului citită de la senzorul de altitudine.
- Butonul de achitare acționat de pilot al semnalului Warning.

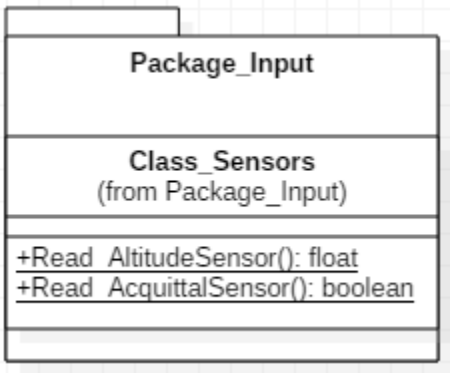
Conține clasa Class_Sensors.

6.1.2. Interfață

Pachetul Package_Input primește date din exteriorul sistemului de la senzorul de altitudine și de la butonul de achitare al semnalului Warning de către pilot, sau, în cazul simulat, primește valorile simulate ale acestor senzori de la pachetul Package_GUI.

Pachetul Package_Input transmite către pachetul Package_Monitor valorile primite de la senzori sau, în cazul simulat, de la pachetul Package_GUI.

6.1.3. Conținut



Metodă	Descriere
Read_AltitudeSensor()	Citește și returnează valoarea altitudinii de la senzor/GUI (în cazul simulat). `public static float Read_AltitudeSensor()` `{` `  return Class_GUI.Get_AltitudeInputVariable();` `}`

Read_AcquittalSensor()	Citește și returnează valoarea stării de achitare de la buton/GUI (în cazul simulat). `public static boolean Read_AcquittalSensor()` `{` `  return Class_GUI.Get_AcquittalInputVariable();` `}`

6.2. Package_Output

6.2.1. Descriere

Se ocupă cu transmiterea comenzilor ieșirilor din SIC. Ieșirile din sistem sunt:

- LED-urile corespunzătoare semnalelor „Error”, „Warning”, „Alarm”.
- Afișarea altitudinii pe un ecran de tip ceas.
- Afișarea stării pilotului automat: activat / dezactivat.

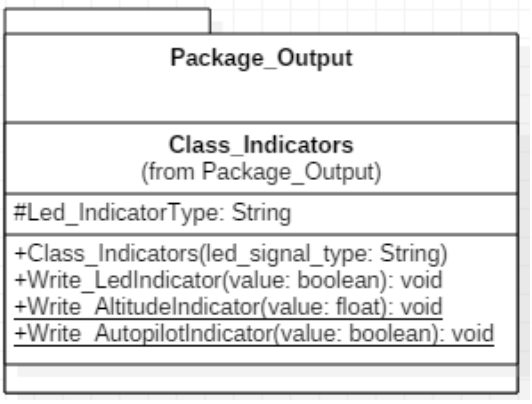
Conține clasa Class_Indicators.

6.2.2. Interfață

Pachetul Package_Output primește date de la pachetul Package_Monitor, date ce reprezintă anumite valori ale comenzilor pe care trebuie să le transmită mai departe.

Pachetul Package_Output transmite către exteriorul sistemului comenzi către cele 3 tipuri de LED-uri (Error, Warning, Alarm), către afișorul cu ceas al altitudinii și către un afișor al stării (on, off) a pilotului automat, sau, în cazul simulat, transmite comenzile către pachetul Package_GUI care simulează indicatoarele menționate.

6.2.3. Conținut



Metodă	Descriere
Class_Indicators()	Constructorul clasei inițializează tipul obiectului LED: Error, Warning, Alarm. `public Class_Indicators(String led_signal_type)` `{` `  this.Led_IndicatorType = led_signal_type;` `}`
Write_LedIndicator()	Transmite comenzi (culoare verde, roșu) LED-urilor/GUI-ului (în cazul simulat). `public void Write_LedIndicator(boolean value)` `{` `  if (this.Led_IndicatorType == "Error")` `  {` `  if (value == false)` `  {` `  Class_GUI.Set_ErrorLedOutputVariable(Color.GREEN);` `  }` `  else` `  {` `  Class_GUI.Set_ErrorLedOutputVariable(Color.RED);` `  }` `  }` `  else if (this.Led_IndicatorType == "Warning")` `  {` `  if (value == false)` `  {` `  Class_GUI.Set_WarningLedOutputVariable(Color.GREEN);` `  }` `  else` `  {` `  Class_GUI.Set_WarningLedOutputVariable(Color.RED);` `  }` `  }` `  else if (this.Led_IndicatorType == "Alarm")` `  {` `  if (value == false)` `  {` `  Class_GUI.Set_AlarmLedOutputVariable(Color.GREEN);` `  }` `  else` `  {` `  Class_GUI.Set_AlarmLedOutputVariable(Color.RED);` `  }` `  }` `}`
Write_AltitudeIndicator()	Transmite comanda de afișare a altitudinii ceasului/GUI-ului (în cazul simulat). `public static void Write_AltitudeIndicator(float value)` `{` `  Class_GUI.Set_AltitudeOutputVariable(value);` `}`

Write_AutopilotIndicator()	Transmite comanda de afișare a stării autopilotului/GUI-ului (în cazul simulat). `public static void Write_AutopilotIndicator(boolean value)` `{` `  if (value == false)` `  {` `  Class_GUI.Set_AutopilotOutputVariable("OFF");` `  }` `  else` `  {` `  Class_GUI.Set_AutopilotOutputVariable("ON");` `  }` `}`

6.3. Package_GUI

6.3.1. Descriere

Se ocupă cu realizarea unui GUI pentru:

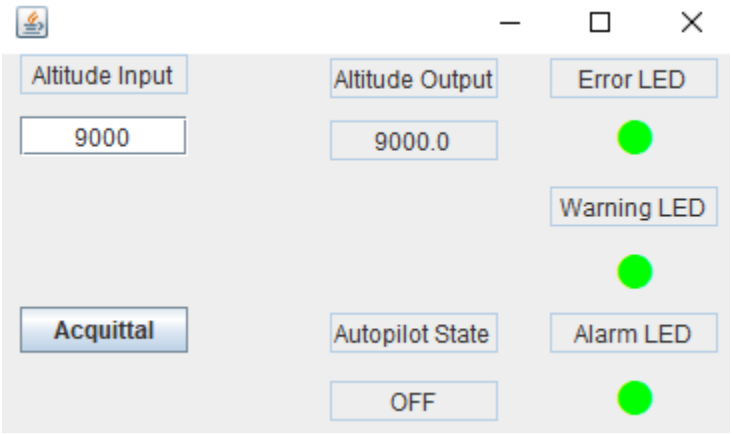
- Simularea intrărilor în sistem utilizate de Package_Input.
- Vizualizarea ieșirilor din sistem transmise de Package_Output.

Conține clasa Class_GUI.

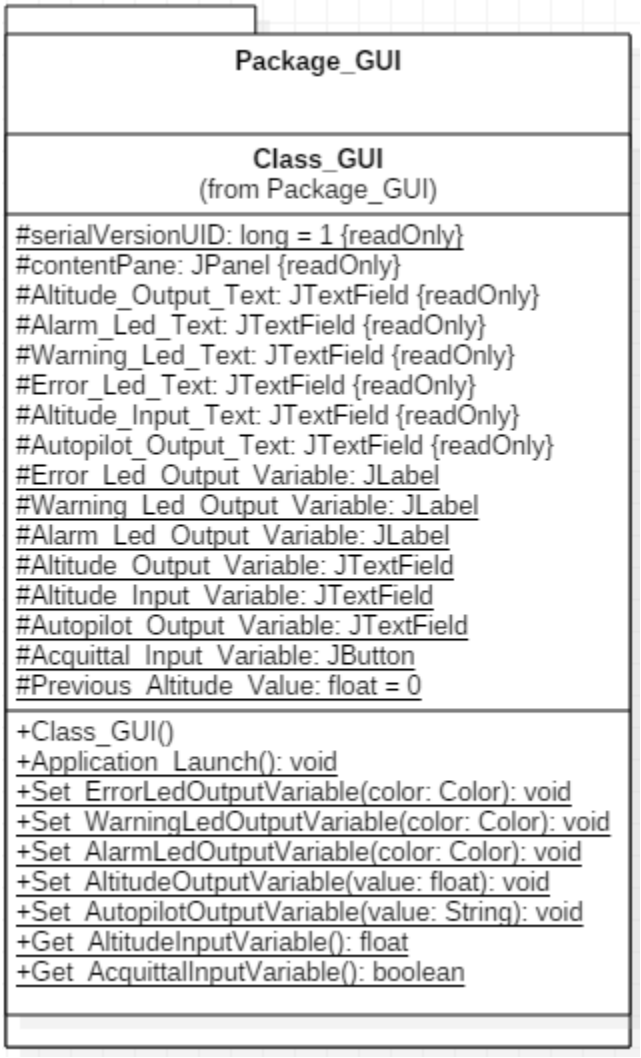
6.3.2. Interfață

Pachetul Package_GUI primește date de la pachetul Package_Output, date ce reprezintă anumite valori ale comenzilor indicatoarelor: LED-uri, afișorul cu ceas al altitudinii și afișorul stării pilotului automat.

Pachetul Package_GUI transmite date către pachetul Package_Input ale senzorilor de altitudine și de achitare a semnalului Warning de către pilot.



6.3.3. Conținut



Metodă	Descriere
Application_Launch()	Creează frame-ul GUI-ului și îl afișează pe ecran. `public static void Application_Launch()` `{` `  EventQueue.invokeLater(new Runnable()` `  {` `  public void run()` `  {` `  try` `  {` `  Class_GUI frame = new Class_GUI();` `  frame.setVisible(true);` `  }` `  catch (Exception e) { }` `  }` `  });` `}`

Class_GUI()	Constructorul clasei inițializează elementele GUI-ului. `public Class_GUI()` `{` `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` `setBounds(100, 100, 385, 229);` `contentPane = new JPanel();` `contentPane.setForeground(Color.BLACK);` `contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));` `setContentPane(contentPane);` `contentPane.setLayout(null);` `Error_Led_Output_Variable.setHorizontalAlignment(SwingConstants.CENTER);` `Error_Led_Output_Variable.setFont(new Font("Tahoma", Font.PLAIN, 50));` `Error_Led_Output_Variable.setForeground(Color.GREEN);` `Error_Led_Output_Variable.setBounds(295, 20, 45, 36);` `contentPane.add(Error_Led_Output_Variable);` `Warning_Led_Output_Variable.setHorizontalAlignment(SwingConstants.CENTER);` `Warning_Led_Output_Variable.setFont(new Font("Tahoma", Font.PLAIN, 50));` `Warning_Led_Output_Variable.setForeground(Color.GREEN);` `Warning_Led_Output_Variable.setBounds(295, 87, 45, 36);` `contentPane.add(Warning_Led_Output_Variable);` `Alarm_Led_Output_Variable.setHorizontalAlignment(SwingConstants.CENTER);` `Alarm_Led_Output_Variable.setFont(new Font("Tahoma", Font.PLAIN, 50));` `Alarm_Led_Output_Variable.setForeground(Color.GREEN);` `Alarm_Led_Output_Variable.setBounds(295, 153, 45, 30);` `contentPane.add(Alarm_Led_Output_Variable);` `Altitude_Output_Text.setHorizontalAlignment(SwingConstants.CENTER);` `Altitude_Output_Text.setText("Altitude Output");` `Altitude_Output_Text.setEditable(false);` `Altitude_Output_Text.setBounds(165, 2, 84, 20);` `Altitude_Output_Text.setColumns(10);` `contentPane.add(Altitude_Output_Text);` `Altitude_Output_Variable.setHorizontalAlignment(SwingConstants.CENTER);` `Altitude_Output_Variable.setEditable(false);` `Altitude_Output_Variable.setColumns(10);` `Altitude_Output_Variable.setBounds(165, 33, 84, 20);` `contentPane.add(Altitude_Output_Variable);` `Altitude_Input_Variable.setText("9000");` `Altitude_Input_Variable.setHorizontalAlignment(SwingConstants.CENTER);` `Altitude_Input_Variable.setBounds(10, 31, 84, 20);` `Altitude_Input_Variable.setColumns(10);` `contentPane.add(Altitude_Input_Variable);` `Alarm_Led_Text.setHorizontalAlignment(SwingConstants.CENTER);` `Alarm_Led_Text.setText("Alarm LED");` `Alarm_Led_Text.setEditable(false);` `Alarm_Led_Text.setColumns(10);` `Alarm_Led_Text.setBounds(275, 129, 84, 20);` `contentPane.add(Alarm_Led_Text);` `Warning_Led_Text.setText("Warning LED");` `Warning_Led_Text.setHorizontalAlignment(SwingConstants.CENTER);` `Warning_Led_Text.setEditable(false);` `Warning_Led_Text.setColumns(10);` `Warning_Led_Text.setBounds(275, 66, 84, 20);` `contentPane.add(Warning_Led_Text);`

	`` Error_Led_Text.setText("Error LED"); Error_Led_Text.setHorizontalAlignment(SwingConstants.CENTER); Error_Led_Text.setEditable(false); Error_Led_Text.setColumns(10); Error_Led_Text.setBounds(275, 2, 84, 20); contentPane.add(Error_Led_Text); Altitude_Input_Text.setText("Altitude Input"); Altitude_Input_Text.setHorizontalAlignment(SwingConstants.CENTER); Altitude_Input_Text.setEditable(false); Altitude_Input_Text.setColumns(10); Altitude_Input_Text.setBounds(10, 0, 84, 20); contentPane.add(Altitude_Input_Text); Acquittal_Input_Variable.setBounds(10, 126, 84, 23); contentPane.add(Acquittal_Input_Variable); Autopilot_Output_Text.setText("Autopilot State"); Autopilot_Output_Text.setHorizontalAlignment(SwingConstants.CENTER); Autopilot_Output_Text.setEditable(false); Autopilot_Output_Text.setColumns(10); Autopilot_Output_Text.setBounds(165, 129, 84, 20); contentPane.add(Autopilot_Output_Text); Autopilot_Output_Variable.setText("OFF"); Autopilot_Output_Variable.setHorizontalAlignment(SwingConstants.CENTER); Autopilot_Output_Variable.setEditable(false); Autopilot_Output_Variable.setColumns(10); Autopilot_Output_Variable.setBounds(165, 163, 84, 20); contentPane.add(Autopilot_Output_Variable); } ```
Set_ErrorLedOutputVariable()	Modifică valoarea LED-ului asociat semnalului Error. ``` public static void Set_ErrorLedOutputVariable(Color color) { try { Error_Led_Output_Variable.setForeground(color); } catch (Exception e) { } } ```
Set_WarningLedOutputVariable()	Modifică valoarea LED-ului asociat semnalului Warning. ``` public static void Set_WarningLedOutputVariable(Color color) { try { Warning_Led_Output_Variable.setForeground(color); } catch (Exception e) { } } ```

Set_AlarmLedOutputVariable()	Modifică valoarea LED-ului asociat semnalului Alarm. `public static void Set_AlarmLedOutputVariable(Color color)` `{` `  try` `  {` `  Alarm_Led_Output_Variable.setForeground(color);` `  }` `  catch (Exception e) { }` `}`
Set_AltitudeOutputVariable()	Modifică valoarea altitudinii afișate pe display. `public static void Set_AltitudeOutputVariable(float value)` `{` `  try` `  {` `  Altitude_Output_Variable.setText(Float.toString(value));` `  }` `  catch (Exception e) { }` `}`
Set_AutopilotOutputVariable()	Modifică valoarea afișată a stării pilotului automat. `public static void Set_AutopilotOutputVariable(String value)` `{` `  try` `  {` `  Autopilot_Output_Variable.setText(value);` `  }` `  catch (Exception e) { }` `}`
Get_AltitudeInputVariable()	Citește și returnează valoarea altitudinii de la senzorul de altitudine. `public static float Get_AltitudeInputVariable()` `{` `  float Current_Altitude = Previous_Altitude_Value;` `  try` `  {` `  Current_Altitude = Float.parseFloat(Altitude_Input_Variable.getText());` `  Previous_Altitude_Value = Current_Altitude;` `  }` `  catch (Exception e) { }` `  return Current_Altitude;` `}`
Get_AcquittalInputVariable()	Citește și returnează starea achitării de la butonul acționat de pilot. `public static boolean Get_AcquittalInputVariable()` `{` `  boolean Current_Acquittal_State = false;` `  try` `  {` `  Current_Acquittal_State = Acquittal_Input_Variable.getModel().isPressed();` `  }` `  catch (Exception e) { }` `  return Current_Acquittal_State;` `}`

6.4. Package_Monitor

6.4.1. Descriere

Se ocupă cu monitorizarea altitudinii unui avion:

- Preia valorile de intrare prin intermediul Package_Input.
- Prelucrează informațiile, calculând comenzile de ieșire.
- Transmite valorile de ieșire către Package_Output.

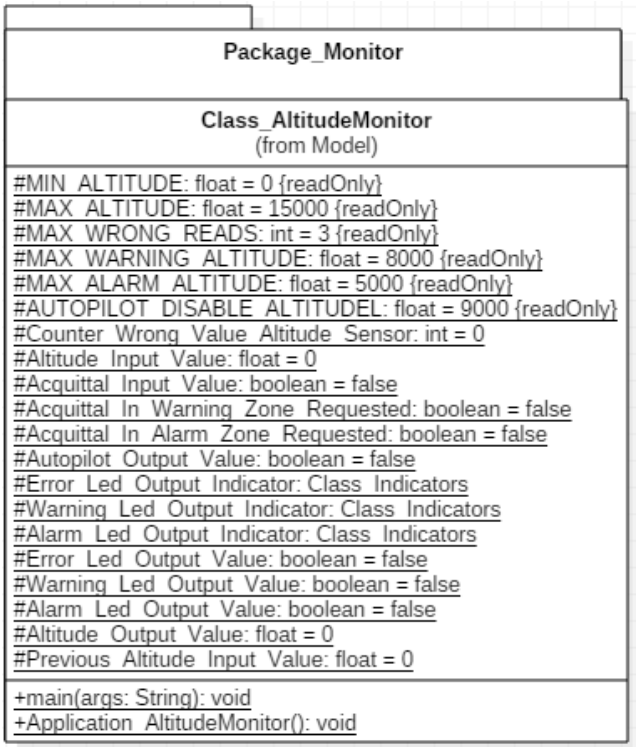
Conține clasa Class_AltitudeMonitor.

6.4.2. Interfață

Pachetul Package_Monitor primește date de la pachetul Package_Input, date ce reprezintă valorile primite de la senzorii de altitudine și de achitare a stării semnalului de Warning.

Pachetul Package_Monitor transmite date către pachetul Package_Output, date ce reprezintă anumite valori ale comenzilor indicatoarelor: LED-uri, afișorul cu ceas al altitudinii și afișorul stării pilotului automat.

6.4.3. Conținut



Metodă	Descriere
main()	Este punctul principal de intrare al programului. Pornește interfața din pachetul Package_GUI cu ajutorul căreia se simulează intrările și ieșirile. Se apelează metoda de monitorizare Application_Launch. `public static void main(String[] args)` `{` `  Class_GUI.Application_Launch();` `  while(true)` `  {` `  Application_AltitudeMonitor();` `  }` `}`
Application_AltitudeMonitor()	Prelucrează intrările de la senzori, calculează și transmite comenzile ce trebuie aplicate ieșirilor. Citește ciclic la o secundă intrările simulate de GUI și calculează ieșirile conform algoritmului. `public static void Application_AltitudeMonitor()` `{` `  // Valorile senzorilor sunt citite la intervale de o secunda` `  try` `  {` `  // asteapta o secunda` `  TimeUnit.SECONDS.sleep(1);` `  }` `  catch (InterruptedException e) { }` `  // Altitudinea este citita cu ajutorul unui "senzor" (GUI).` `  Altitude_Input_Value = Class_Sensors.Read_AltitudeSensor();` `  Acquittal_Input_Value = Class_Sensors.Read_AcquittalSensor();` `  if (Acquittal_Input_Value == true)` `  {` `  if (Altitude_Input_Value < MAX_ALARM_ALTITUDE)` `  {` `  Acquittal_In_Alarm_Zone_Requested = true;` `  }` `  else if (Altitude_Input_Value < MAX_WARNING_ALTITUDE)` `  {` `  Acquittal_In_Warning_Zone_Requested = true;` `  }` `  }` `  // Toate valorile citite trebuie sa fie intr-un interval predefinit...` `  if ( (Altitude_Input_Value < MIN_ALTITUDE) || (Altitude_Input_Value > MAX_ALTITUDE))` `  {` `  // ...Valorile in afara intervalului sunt ignorate.` `  Altitude_Input_Value = Previous_Altitude_Input_Value;` `  Counter_Wrong_Value_Altitude_Sensor++;` `  }` `  else` `  {` `  Previous_Altitude_Input_Value = Altitude_Input_Value;` `  Counter_Wrong_Value_Altitude_Sensor = 0;` `  }` `  // Cand trei citiri succesive ale unui senzor genereaza valori in afara intervalului predefinit...` `  if (Counter_Wrong_Value_Altitude_Sensor == MAX_WRONG_READS)` `  {` `  // ...un led, ce corespunde senzorului respectiv, isi schimba culoarea` `  //din verde in rosu (se genereaza ERROR)` `  Error_Led_Output_Value = true;` `  }` `  // In functie de valorile citite se pot genera semnale WARNING sau ALARM`

```
    // Atunci cand altitudinea este este mai mica decat 8000m, un semnal WARNING este generat
    // Dupa achitare un nou WARNING va fi generat doar daca a
    //existat cel putin o citire pentru care nu s-a generat WARNING sau ALARM.
    // Cand un semnal WARNING este generat...
    if ( (Altitude_Input_Value < MAX_ALARM_ALTITUDE) && (Acquittal_In_Alarm_Zone_Requested == false) )
    {
        // ...un led isi schimba culoarea din verde in rosu.
        Warning_Led_Output_Value = true;
    }
    else if ( (Altitude_Input_Value >= MAX_ALARM_ALTITUDE) && (Altitude_Input_Value < MAX_WARNING_ALTITUDE)
        && (Acquittal_In_Warning_Zone_Requested == false) )
    {
        // ...un led isi schimba culoarea din verde in rosu.
        Warning_Led_Output_Value = true;
    }
    else
    {
        // Un WARNING persista pana cand este achitat de pilot sau dispar conditiile care l-au generat.
        // Daca semnalul WARNING este achitat atunci se activeaza pilotul automat.
        if (Altitude_Input_Value >= MAX_WARNING_ALTITUDE)
        {
            Acquittal_In_Alarm_Zone_Requested = false;
            Acquittal_In_Warning_Zone_Requested = false;
        }
        else if (Acquittal_In_Warning_Zone_Requested == true)
        {
            Autopilot_Output_Value = true;
        }

        Warning_Led_Output_Value = false;
    }

    // Atunci cand altitudinea este este mai mica decat 5000, un semnal ALARM este generat
    // Un ALARM persista pana cand dispar conditiile care l-au generat
    // Dupa aparitia semnalului ALARM se activeaza pilotul automat.
    // Cand un semnal ALARM este generat...
    if (Altitude_Input_Value < MAX_ALARM_ALTITUDE)
    {
        // ...un led isi schimba culoarea din verde in rosu.
        Alarm_Led_Output_Value = true;
        Autopilot_Output_Value = true;
    }
    else
    {
        Alarm_Led_Output_Value = false;
    }

    // Pilotul automat se dezactiveaza daca altitudinea este mai mare decat 9000m
    if (Altitude_Input_Value > AUTOPILOT_DISABLE_ALTITUDE)
    {
        Autopilot_Output_Value = false;
    }

    Error_Led_Output_Indicator.Write_LedIndicator(Error_Led_Output_Value);
    Warning_Led_Output_Indicator.Write_LedIndicator(Warning_Led_Output_Value);
    Alarm_Led_Output_Indicator.Write_LedIndicator(Alarm_Led_Output_Value);

    // Altitudinea este afisata pe un "ecran de tip ceas" (GUI)
    Altitude_Output_Value = Altitude_Input_Value;
    Class_Indicators.Write_AltitudeIndicator(Altitude_Output_Value);

    // afiseaza starea pilotului automat in GUI
    Class_Indicators.Write_AutopilotIndicator(Autopilot_Output_Value);
}
```

7. Matricea de trasabilitate

Identificator Specificație	Componenta software	Comentarii
S001	Class_GUI::Get_AltitudeInputVariable() Class_Sensors::Read_AltitudeSensor()	
S002	Class_GUI::Set_AltitudeOutputVariable() Class_Indicators:: Write_AltitudeIndicator()	
S003	Class_AltitudeMonitor::Application_AltitudeMonitor() Class_Sensors::Read_AltitudeSensor()	
S004	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S005	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S006	Class_AltitudeMonitor	
S007	Class_AltitudeMonitor::Application_AltitudeMonitor() Class_Indicators::Write_LedIndicator() Class_GUI::Set_ErrorLedOutputVariable()	
S008	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S009	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S010	Class_Indicators::Write_LedIndicator() Class_GUI::Set_WarningLedOutputVariable()	
S011	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S012	Class_AltitudeMonitor::Application_AltitudeMonitor() Class_Indicators::Write_AutopilotIndicator()	
S013	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S014	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S015	Class_Indicators::Write_LedIndicator() Class_GUI::Set_AlarmLedOutputVariable()	
S016	Class_AltitudeMonitor::Application_AltitudeMonitor() Class_Indicators::Write_AutopilotIndicator()	
S017	Class_AltitudeMonitor::Application_AltitudeMonitor()	
S018	Class_AltitudeMonitor::Application_AltitudeMonitor()	

	Class_Indicators::Write_AutopilotIndicator()	
S019	Class_GUI::Get_AcquittalInputVariable() Class_Sensors:: Read_AcquittalSensor()	
S020	Class_Indicators::Write_AutopilotIndicator() Class_GUI::Set_AutopilotOutputVariable()	