Katedra Mechaniki Stosowanej i Robotyki
Wydział Budowy Maszyn i Lotnictwa, Politechnika Rzeszowska

Techniki wirtualnej rzeczywistości w mechatronice

Laboratorium nr 02_04

Temat: Programowanie w języku C# w Visual Studio Code

1. Wstęp

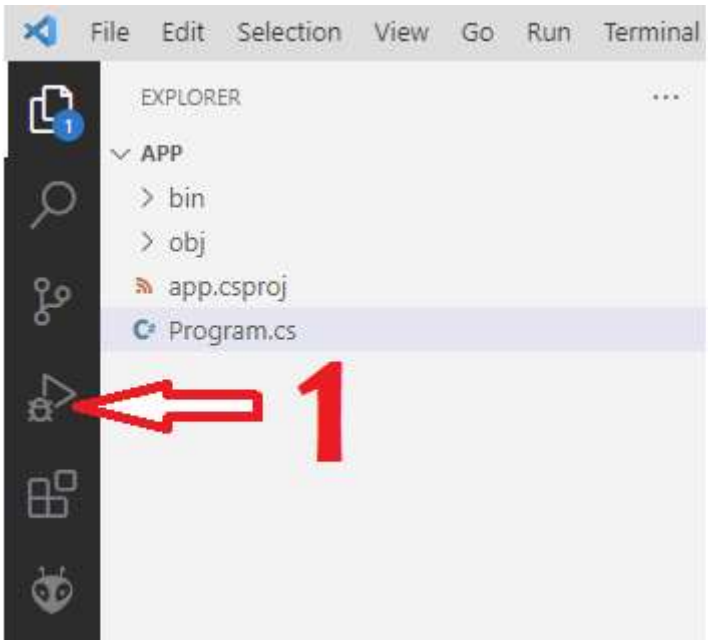
Instrukcja ma na celu zapoznanie studenta z programowaniem w języku C# w Visual Studio Code na podstawie 3 przykładów programów.

2. Przygotowanie projektu programu w Visual Studio Code

Tworzymy folder w którym mają znajdować się programy studenta. Otwieramy program Visual Studio Code. Klikamy File -> Open Folder, zaznaczamy utworzony folder i klikamy Wybierz folder. Klikamy Terminal -> New Terminal i w terminalu piszemy:

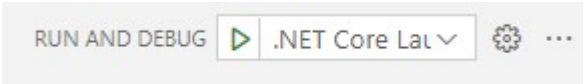
```
dotnet new console -o app_1
cd app_1
```

Gdzie za app_1 wpisujemy nazwę swojego projektu. Klikamy enter. Po całej procedurze powinniśmy uzyskać drzewko programu takie jak na rys. 1.



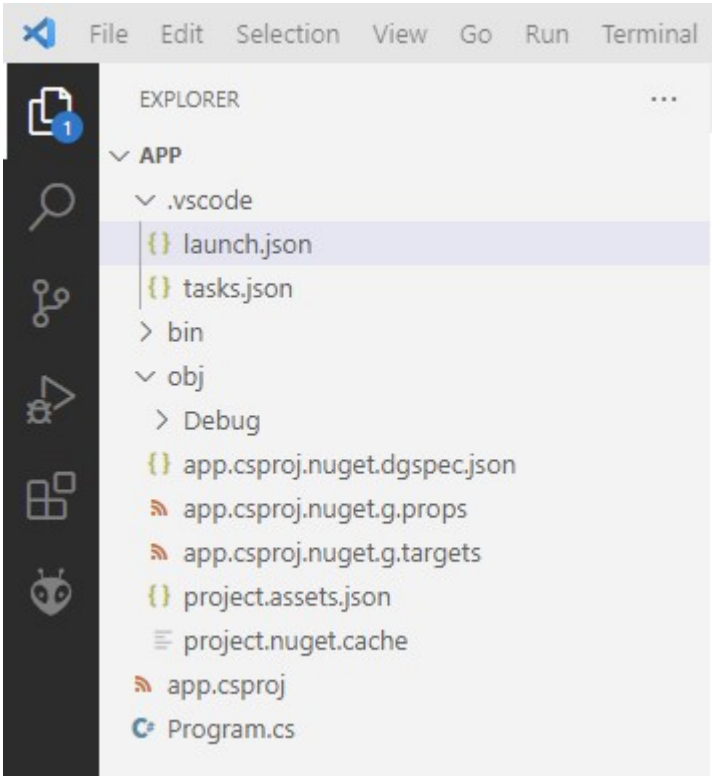
Rys. 1. Stworzony program

Klikamy ikonę oznaczoną jako 1 na rys. powyżej i klikamy przycisk „Generate c# Assets for Build and Debug". Pojawi się nam ikona debugowania jak na rys. 2.



Rys. 2 Uruchomienie i debugowanie programu.

Aby programy poprawnie włączały terminal, w którym będziemy mogli wchodzić w interakcje z programem musimy zmodyfikować jedną linijkę kodu w pliku launch.json (rys.3).



Rys. 3 Położenie pliku launch.json w drzewku programu

Należy zmienić instrukcję "console": "internalConsole", na "console": "externalTerminal",

3. Przykłady programów

Poniżej przedstawiono przykłady programów na których można się wzorować na laboratorium.

a. Kalkulator z obsługą błędów

Program jest kalkulatorem, który umożliwia przeprowadzanie operacji +,-,*,/ na liczbach zmiennoprzecinkowych. Program wyposażony jest w obsługę błędów: niedzielenie przez zero, wprowadzanie odpowiednich danych i konwersacja z użytkownikiem.

```
using System;

namespace CalculatorProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            string op;
            double result = 0; // wynik
            double num1 = 0; // liczba pierwsza
            double num2 = 0; // liczba druga
            bool loop = true; // warunek powtórzenia obliczeń od początku
            bool loop_check_number = true; //

            while (loop)
            {

                Console.WriteLine("Podaj pierwszą liczbę: ");
                while (true)
                {
                    try
                    {
```

```
            num1 = double.Parse(Console.ReadLine());
            break;
        }
        catch (FormatException e)
        {
            Console.WriteLine("Podaj cyfrę.");
            continue;
        }
    }
    Console.WriteLine("Podaj drugą liczbę: ");
    while (true)
    {
        try
        {
            num2 = double.Parse(Console.ReadLine());
            break;
        }
        catch (FormatException e)
        {
            Console.WriteLine("Podaj cyfrę.");
            continue;
        }

    }
    Console.WriteLine("Podaj symbol operacji (+, -, *, /): ");
    while (true)
    {
        op = Console.ReadLine();
        if(op == "+" || op == "-" || op == "*" || op == "/")
        {
            break;
        }
        else
        {
            Console.WriteLine("Podaj właściwy symbol (+, -, *, /).");
        }

    }

    while (true)
    {
            switch (op)
                {
                    case "+":
                        result = num1 + num2;
                        break;
                    case "-":
                        result = num1 - num2;
                        break;
                    case "*":
                        result = num1 * num2;
                        break;
                    case "/":
                        if (num2 == 0)
                        {
                            Console.WriteLine("Nie można dzielić przez zero.");
                            loop_check_number = false;
                        }
                        else
                            result = num1 / num2;
                        break;
                }
                if(loop_check_number)
                Console.WriteLine("Wynik: " + result);
                loop_check_number = true;
                Console.WriteLine("Naciśnij dowolny przycisk by kontynuować albo Q by wyłączyć
kalkulator.");
                if (Console.ReadLine().ToUpper() == "Q")
```

```
                {

                    loop = false;
                }
                else
                {
                    loop = true;
                }
                break;

            }
        }

    }
}
```

b. Program demonstrujący działanie dziedziczenia w klasach

Program wyznacza pole kwadratu i koła na podstawie wcześnie zdefiniowanej bazowej klasy „kształt”.

```
using System;

public abstract class Shape
{
    protected double area;
    private double length;
    private double width;

    public virtual void CalculateArea()
    {
        Console.WriteLine("Wyznaczanie pola kształtu...");
    }

    public void DisplayArea()
    {
        Console.WriteLine("The area of the shape is: " + area);
    }
}

public class Rectangle : Shape
{
    private double length;
    private double width;

    public Rectangle(double length, double width)
    {
        this.length = length;
        this.width = width;
    }

    public override void CalculateArea()
    {
        area = length * width;
        Console.WriteLine("Wyznaczanie pola prostokąta...");
    }
}

public class Circle : Shape
{
    private double radius;

    public Circle(double radius)
    {
        this.radius = radius;
    }
```

```
    public override void CalculateArea()
    {
        area = Math.PI * Math.Pow(radius, 2);
        Console.WriteLine("Wyznaczanie pola koła...");
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        Rectangle myRectangle = new Rectangle(5, 3);
        Circle myCircle = new Circle(4);

        myRectangle.CalculateArea();
        myRectangle.DisplayArea();

        myCircle.CalculateArea();
        myCircle.DisplayArea();
        Console.ReadKey();
    }
}
```

c. Program obsługujący bibliotekę

System biblioteczny obejmuje trzy rodzaje pozycji: książki, płyty CD i DVD. Każda pozycja posiada tytuł, autora/artystę, opis oraz unikalny numer identyfikacyjny.

System biblioteczny ma następującą funkcjonalność:

Możliwość dodawania nowych pozycji do systemu bibliotecznego.
Możliwość wyszukiwania pozycji po numerze identyfikacyjnym.
Możliwość wyświetlenia wszystkich pozycji w systemie bibliotecznym.
Możliwość sprawdzenia przedmiotu.
Możliwość zwrotu przedmiotu.
Możliwość usuwania przedmiotu.

Zadanie utworzenia biblioteki:

Utwórz klasę bazową dla elementów w systemie bibliotecznym, a następnie utwórz osobne klasy dla każdego typu elementów dziedziczących z klasy podstawowej. Zaimplementuj funkcjonalność dla każdej klasy i upewnij się, że zasady wyewidencjonowywania elementów są egzekwowane.

Na koniec utwórz program główny, który umożliwia użytkownikowi interakcję z systemem bibliotecznym, w tym dodawanie pozycji, wyszukiwanie pozycji, wypożyczanie pozycji i zwracanie pozycji.

```
using System;
using System.Collections.Generic;

public abstract class LibraryItem
{
// zdefiniowanie zmiennych z operatorami get i set
public string Title { get; set; }
public string AuthorOrArtist { get; set; }
public string Description { get; set; }
public int ItemId { get; set; }
public bool IsCheckedOut { get; set; }
```

```
public LibraryItem(string title, string authorOrArtist, string description, int itemId)
{
    // przypisanie argumetów metody do zmiennych wcześniej zdefiniowanych
    Title = title;
    AuthorOrArtist = authorOrArtist;
    Description = description;
    ItemId = itemId;
    IsCheckedOut = false;
}

public virtual void DisplayItem()
{
    // symbol $ umożliwia wstawianie zmiennych do tekstu
    Console.WriteLine($"Tytuł: {Title}");
    Console.WriteLine($"Author/Artist: {AuthorOrArtist}");
    Console.WriteLine($"Description: {Description}");
    Console.WriteLine($"Item ID: {ItemId}");
}

}

public class Book : LibraryItem
{
    // utworzenie metody tworzączej książkę pobierając elementy z klasy bazowej LibraryItem
    public Book(string title, string author, string description, int itemId) : base(title, author,
description, itemId){}
    public override void DisplayItem()
    {
        Console.WriteLine("Book:");
        base.DisplayItem();
    }
}

public class CD : LibraryItem
{
    // utworzenie metody tworzączej CD pobierając elementy z klasy bazowej LibraryItem
    public CD(string title, string artist, string description, int itemId) : base(title, artist, description,
itemId){}
    public override void DisplayItem()
    {
        Console.WriteLine("CD:");
        base.DisplayItem();
    }
}

public class DVD : LibraryItem
{
    // utworzenie metody tworzączej DVD pobierając elementy z klasy bazowej LibraryItem
    public DVD(string title, string director, string description, int itemId) : base(title, director,
description, itemId){}
    public override void DisplayItem()
    {
        Console.WriteLine("DVD:");
        base.DisplayItem();
    }
}

public class Library
{
    // utworzenie nowej kolekcji (klasy) typy LibraryItem
    private List<LibraryItem> _items;

    public Library()
    {
         // utworzenie nowej listy _items
        _items = new List<LibraryItem>();
    }
```

```
public void AddItem(LibraryItem item)
{
    // dodawanie elementu klasy LibraryItem do Listy _items
    _items.Add(item);
    Console.WriteLine($"Item with ID {item.ItemId} added to library.");
}

public LibraryItem GetItemById(int itemId)
{
    // uzyskiwanie elementów z biblioteki po numerze ID
    foreach (LibraryItem item in _items)
    {
        if (item.ItemId == itemId)
        {
            return item;
        }
    }
    Console.WriteLine($"Item with ID {itemId} not found.");
    return null;
}

public void DisplayAllItems()
{
    // wyświetlanie wszystkich elementów biblioteki
    Console.WriteLine("All items in library:");
    foreach (LibraryItem item in _items)
    {
        item.DisplayItem();
    }
}

public void CheckOutItem(int itemId)
{
    // zaznaczanie że nie ma elementu w bibliotece
    LibraryItem item = GetItemById(itemId);
    if (item != null)
    {
        if (item.IsCheckedOut)
        {
            Console.WriteLine($"Item with ID {itemId} is already checked out.");
        }
        else
        {
            item.IsCheckedOut = true;
            Console.WriteLine($"Item with ID {itemId} checked out.");
        }
    }
}

public void ReturnItem(int itemId)
    {
        // oddanie elementu do biblioteki
    LibraryItem item = GetItemById(itemId);
        if (item != null)
        {
            if (item.IsCheckedOut)
                {
                item.IsCheckedOut = false;
                Console.WriteLine($"Item with ID {itemId} returned.");
                }
                else
                {
                Console.WriteLine($"Item with ID {itemId} is not currently checked out.");
                }
        }
    }

public void DeleteItem(int itemId)
```

```
    {   // usuwanie elementu z listy
        LibraryItem item = GetItemById(itemId);
        if (item != null)
        {
            _items.Remove(item);
            Console.WriteLine($"Item with ID {itemId} deleted from library.");
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        // utworzewnie nowej biblioteki
        Library library = new Library();
        // utworzewnie nowych elementów biblioteki
        Book book1 = new Book("Harry Potter and the Philosopher's Stone", "J.K. Rowling", "The first book in
the Harry Potter series", 1);
        Book book2 = new Book("To Kill a Mockingbird", "Harper Lee", "A classic novel about racism and
injustice in the American South", 2);
        CD cd1 = new CD("Abbey Road", "The Beatles", "The Beatles' eleventh studio album", 3);
        CD cd2 = new CD("Back in Black", "AC/DC", "AC/DC's seventh studio album", 4);
        DVD dvd1 = new DVD("The Godfather", "Francis Ford Coppola", "A classic film about the Corleone crime
family", 5);
        DVD dvd2 = new DVD("The Shawshank Redemption", "Frank Darabont", "A drama film about a man's escape
from prison", 6);

        // dodanie elementów do biblioteki
        library.AddItem(book1);
        library.AddItem(book2);
        library.AddItem(cd1);
        library.AddItem(cd2);
        library.AddItem(dvd1);
        library.AddItem(dvd2);

        library.DisplayAllItems();

        library.CheckOutItem(1);
        library.CheckOutItem(3);
        library.CheckOutItem(5);

        Console.WriteLine();

        library.DisplayAllItems();

        library.ReturnItem(1);
        library.ReturnItem(3);

        Console.WriteLine();

        library.DisplayAllItems();

        library.DeleteItem(2);

        library.DisplayAllItems();
        Console.ReadKey();
    }
}
```

4. Zadania do wykonania

Zad. 1

Utwórz program konwersacyjny (w terminal jak kalkulator) w którym będzie procedura tworzenia konta bankowego. Użytkownik powinien mieć możliwość wyboru banku z 3 dostępnych po 2

oprocentowania oraz podania swoich danych: imienia, nazwiska, wieku, adresu, salda i oprocentowania (system powinien posiadać możliwość obsługi błędów tak jak w programie kalkulator).

Zad. 2

Zmodyfikuj poprzedni program tak by składał się z 3 klas. Klasa bazową „osoba" oraz dziedziczące „dorosły" i „dziecko". Utwórz klasę bazową dla kont, a następnie utwórz osobne klasy dla każdego typu konta, które dziedziczą po klasie bazowej. Klasy „dorosły" „dziecko" dziedziczą dane i metody z klasy „osoba". „Dziecko" ma ograniczenie salda do 20 000 i oprocentowanie 2%, „dorosły" nie ma limitu i ma oprocentowanie 0.5%. Pozostaw funkcjonalności z poprzedniego programu.

Zad. 3

Stwórz program modelujący prosty system bankowy. System bankowy powinien posiadać rachunki z saldem oraz możliwością wypłaty i wpłaty pieniędzy.

System bankowy powinien posiadać następującą funkcjonalność:

Możliwość utworzenia nowego konta z saldem początkowym (tak jak w zad 2).
Możliwość wpłaty pieniędzy na konto.
Możliwość wypłaty pieniędzy z konta.
Możliwość sprawdzenia stanu konta.
Możliwość przelewania pieniędzy z jednego konta na drugie.

Zaimplementuj funkcjonalność dla każdej klasy i upewnij się, że konta nie mogą zostać przekroczone.

Na koniec utwórz główny program, który umożliwi użytkownikowi interakcję z systemem bankowym, w tym tworzenie kont, wpłacanie i wypłacanie pieniędzy, sprawdzanie sald i przelewanie pieniędzy między kontami.

W systemie powinny istnieć już dwa konta utworzone podobnie jak w 3 przykładzie. Trzecie konto powinno być dodawane w systemie konwersacyjnym.

Student dostaje:
ocenę 4 za prawidłowe wykonanie zad 1,
ocenę 4.5 za zad. 1 i 2,
ocenę 5 za zad. 1,2 i 3.

W rozwiązywaniu zadań można posiłkować się materiałami dydaktycznymi, przykładami z instrukcji i Internetu oraz dokumentacji języka C# na stronie: https://learn.microsoft.com/pl-pl/dotnet/csharp/