



Zachodniopomorski  
Uniwersytet Technologiczny  
w Szczecinie

TEMAT:

APLIKACJA TRENINGOWA

Imię i Nazwisko:

MICHAŁ GÓRA  
MARCEL SARNECKI

Wydział Inżynierii  
Mechanicznej i  
Mechatroniki

Grupa: LA01  
MECHATRONIKA  
Semestr VI

PRZEDMIOT:

Inżynieria oprogramowania (projekt)

OCENA: .....

PROWADZĄCY:

dr inż. Kamil Stateczny

PODPIS: .....

## **1. CEL PROJEKTU**

Celem projektu jest poznanie środowiska programistycznego aplikacji, poznanie wzorców projektowych wykorzystywanych w aplikacjach oraz zaprojektowanie aplikacji w języku C#.

## **2. ZAŁOŻENIA PROJEKTU**

Zaprojektowana aplikacja jest przeznaczona dla osoby mającej związek lub nie mającej związku ze sportem. Zaczynając od amatora, a kończąc na zawodowcu.

Założenia funkcjonalne aplikacji:

- Zadaniem aplikacji jest ułożenie odpowiedniego treningu na podstawie testu wykonanego przez użytkownika.
- Za pomocą aplikacji, możemy dobrać plan treningowy z zależności od naszych predyspozycji oraz chęci uzyskania odpowiedniego efektu.
- Aplikacja oblicza na podstawie sprawdzianu wartości, poziom: BMI, siły, zwinności i wydajności, a następnie komentuje je w celu zrozumienia przez większe grono użytkowników.
- Aplikacja pozwala nam sprawdzać progres użytkownika.
- Aplikacja prezentuje treści zachęcające do polepszania swojej siły i stanu kondycyjnego swojego organizmu;
- Aplikacja łatwa w obsłudze.
- W aplikacji nie trzeba podawać za każdym razem wszystkich wyników, możliwe jest podanie każdego wyniku osobno.
- Istnieje możliwość wyczyszczenia widoku lub historii treningowej bez zbędnych komplikacji.
- Aplikacja przeznaczona na telefon.

### 3. WYBÓR ROZWIĄZANIA PROJEKTU

W celu napisania programu wykorzystano zintegrowane środowisko programistyczne Microsoft Visual Studio 2019. Oprogramowanie to zostało wybrane z powodu zaawansowanej pomocy w trakcie pisania kodu programu, oraz przy możliwości sprawdzenia działania programu i analizy przy występujących usterkach.

W trakcie pisania i w celu dobrego działania programu wykorzystano odpowiednio wzorce projektowe:

- **BUILDER (Konstruktor)**, wykorzystany w celu utworzenia złożonych obiektów etapami, krok po kroku. Wzorec ten pozwala produkować różne typy oraz reprezentacje obiektu używając tego samego kodu konstrukcyjnego.

Aplikacja treningowa działa na zasadzie kolejno wykonywanych etapów. Jako pierwszy etap wykonujemy test, bez którego nie będziemy mogli przejść do etapów kolejnych, z powodu braku danych. Kolejnym etapem jest przygotowanie planu treningowego, następnie możemy ale nie musimy zobaczyć efekt pracy włożonej w treningi i porównać nasze dokonania. Jeżeli chodzi o etap wyników, nie jest on wymaganym etapem do wykonania aby aplikacja działała w sposób poprawny a możemy wywołać ten etap w dowolnym momencie użytkowania.

- **WSTRZYKIWANIE ZALEŻNOŚCI (Dependency Injection)**, służy głównie do wstrzykiwania konkretnej implementacji do klasy używającej abstrakcji, np. interfejsu. Główną ideą wstrzykiwania zależności jest redukcja połączeń pomiędzy klasami oraz przeniesienie łączenia abstrakcji z konkretną implementacją poza klasę zależną.

Wstrzykiwanie zależności może odbywać się na trzy sposoby:

- przez konstruktor;
- przez metodę;
- przez właściwość.

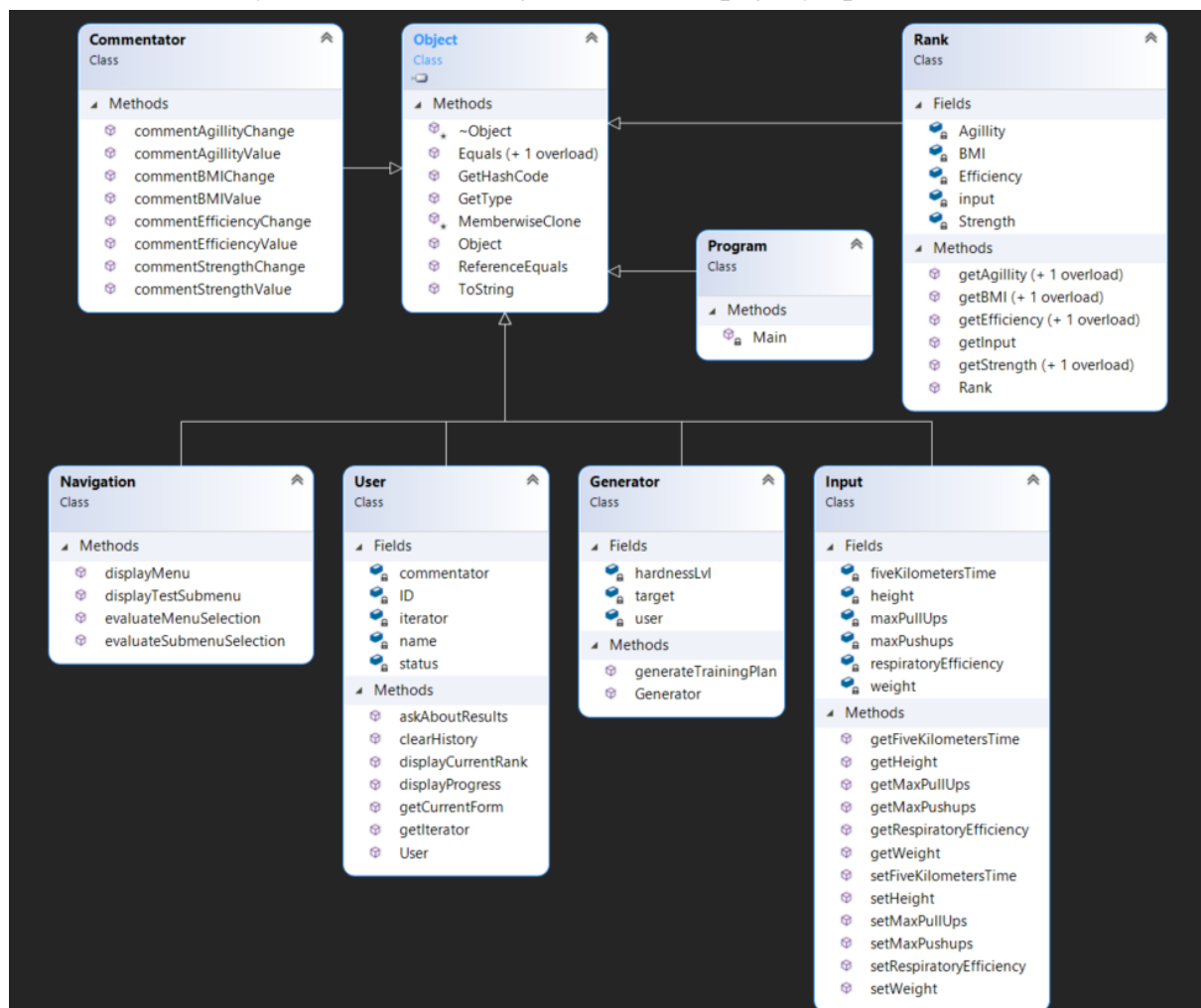
Aplikacja treningowa korzysta z wstrzykiwania zależności przez konstruktor. Polega to na przekazaniu obiektu konkretnej klasy przez konstruktor klasy zależnej. Klasa zależna - class Generator wywołuje

obiekt pochodzący z klasy User, w celu wygenerowania treningu użytkownika na podstawie testu i parametrów wpisanych przez użytkownika aplikacji.

W celu uzyskania odpowiedniego rozkładu treningowego na potrzeby organizmu, wykorzystaliśmy procentowy rozkład treningowy odpowiednio do przypisanych poziomów zaawansowania treningu. Dodatkowo wykorzystany został wzorzec wskaźnika BMI do badania odpowiedniej wagi użytkownika przy swoim wzroście.

## 4. DIAGRAM KLAS

W aplikacji znajduje się siedem klas, które wyglądają w następujący sposób. Zależność między klasami kształtuje się w następujący sposób.



## 5. OMÓWIENIE KLUCZOWYCH CZĘŚCI KODU

*Public class Generator* odpowiada za generowanie treningów. W tej klasie zdefiniowane są dwie zmienne, które należy podać w aplikacji: poziom trudności (*hardnesslvl*) oraz cel treningowy (*target*). Wybierając poziom trudności posiadamy do wyboru trzy opcje: *easy*, *normal*, *hard*. Wybierając cel treningowy mamy dostępne trzy opcje *strength*, *agility*, *general*. Każda z tych opcji przewiduje inny rodzaj treningu i oblicza go na podstawie wprowadzonych danych w punkcie 3 pomnożonych przez współczynnik procentowy w zależności od poziomu trudności.

```
3 references
public class Generator
{ // nr 1.

    public User user;
    public String hardnessLvl; // poziom trudności
    public String target; // cel treningowy

    1 reference
    public Generator(User user, string hardnessLvl, string target)
    {
        this.user = user;
        this.hardnessLvl = hardnessLvl;
        this.target = target;
    }

    1 reference
    public void generateTrainingPlan() //generowanie treningow
    {
        double percentage = 0;

        switch (this.hardnessLvl) //wyrażenie to umożliwia oszacowanie pojedynczego wyrażenia z listy wyrażen
                                   //kandydowania na podstawie dopasowania wzorca switch do wyrażenia wejściowego
        {
            case "easy":
                percentage = 0.25;
                break;
            case "normal":
                percentage = 0.50;
                break;
            case "hard":
                percentage = 0.70;
                break;
        }
    }
}
```

```

switch (this.target)
{
    case "strength":
        Console.WriteLine("Wybrano trening na siłę ");
        Console.Write("Dzień x - 5 serii po ");
        Console.Write(Math.Floor(percentage * user.status[user.iterator - 1].input.maxPushups));
        Console.WriteLine(" pompek");
        Console.Write("Dzień x - 5 serii po ");
        Console.Write(Math.Floor(percentage * user.status[user.iterator - 1].input.maxPullUps));
        Console.WriteLine(" podciągnąć");
        break;
    case "agillity":
        Console.WriteLine("Wybrano trening na zręczność ");
        Console.Write("Dzień x - 5 serii po ");
        Console.Write(Math.Floor(percentage * user.status[user.iterator - 1].input.maxPushups));
        Console.WriteLine(" pompek");
        Console.Write("Dzień x - 3 serie biegu po ");
        Console.Write(Math.Floor(percentage * user.status[user.iterator - 1].input.fiveKilometersTime));
        Console.WriteLine(" minut");
        break;
    case "general":
        Console.WriteLine("Wybrano trening ogólny ");
        Console.Write("Dzień x - 5 serii biegu po ");
        Console.Write(Math.Floor(percentage * user.status[user.iterator - 1].input.fiveKilometersTime));
        Console.WriteLine(" minut");
        break;
    default:
        break;
}
}
}

```

*Public class Navigation* odpowiada za nawigację po menu głównym (6 punktów) oraz menu dodania wyników sprawdzianu w punkcie 3 (5 podpunktów).

```
public class Navigation // klasa odpowiedzialna za nawigację po menu
{
    2 references
    public void displayMenu()
    {
        Console.WriteLine(" ");
        Console.WriteLine("1. - Wyświetl ranking użytkownika");
        Console.WriteLine("2. - Generuj plan treningowy użytkownika");
        Console.WriteLine("3. - Dodaj wyniki sprawdzianu");
        Console.WriteLine("4. - Wyświetl progres użytkownika");
        Console.WriteLine("5. - Wyczyść widok");
        Console.WriteLine("6. - Wyczyść historię użytkownika");
    }
}
```

```
public void evaluateMenuSelection(User user)
{
    //dane które podaje w menu 1-6

    int selection = Int32.Parse(Console.ReadLine());

    switch (selection)
    {
        case 1:
            user.displayCurrentRank();
            break;
        case 2:
            String target; //cel treningowy
            String hardnessLvl; // poziom trudności

            Console.Write("Podaj poziom trudności (easy/normal/hard); //trzy poziomy trudności
            Console.WriteLine();
            hardnessLvl = Console.ReadLine();

            Console.Write("Podaj cel treningu (strength/agility/general); //trzy rodzaje treningu
            Console.WriteLine();
            target = Console.ReadLine();

            Console.Clear();

            Generator generator = new Generator(user, hardnessLvl, target); //przeciążony konstruktor
            generator.generateTrainingPlan();

            break;
        case 3:
            Console.Clear();
            displayTestSubmenu(); // dodanie wyników sprawdzianu
            evaluateSubmenuSelection(user);
            break;
        case 4:
            user.displayProgress(); // wyswietlanie progresu
            break;
        case 5:
            Console.Clear(); //czyszczenie konsoli
            break;
        case 6:
            user.clearHistory(); //czyszczenie całej historii treningowej
            break;
        default:
            break;
    }
}
```

```

public void displayTestSubmenu() //menu dodania wynikow sprawdzianu
{
    Console.Clear();
    Console.WriteLine("1 - Podaj wszystkie dane");
    Console.WriteLine("2 - Podaj wagę");
    Console.WriteLine("3 - Podaj wyniki testu pompek");
    Console.WriteLine("4 - Podaj wyniki testu na drążki");
    Console.WriteLine("5 - Podaj czas biegu na 5 km");
}
1 reference
public User evaluateSubmenuSelection(User userIn)
{
    User user = userIn;

    int selection = Int32.Parse(Console.ReadLine());

    switch (selection) // pytania odnośnie danych
    {
        case 1:
            user.askAboutResults("all"); // wszystkich danych

            break;
        case 2:
            user.askAboutResults("weight"); //wagi

            break;
        case 3:
            user.askAboutResults("pushups"); //ilości pompek

            break;
        case 4:
            user.askAboutResults("pullups"); //podciągnięcie

            break;
        case 5:
            user.askAboutResults("5km run time"); //czasu biegu na 5km

            break;
        default:
            break;
    }

    return user;
}

```



*Public class User* odpowiada za strukturę programu. W klasie tej znajduje się szczegółowe menu podania wszystkich wartości w punkcie 3.

```
case "all":
    if (iterator < 20)
    {
        Console.WriteLine("Podaj wzrost :");
        input.setHeight(Int32.Parse(Console.ReadLine()));

        Console.WriteLine("Podaj wagę :");
        input.setWeight(Int32.Parse(Console.ReadLine()));

        Console.WriteLine("Podaj maksymalną liczbę pompek :");
        input.setMaxPushups(Int32.Parse(Console.ReadLine()));

        Console.WriteLine("Podaj maksymalną liczbę podciągnięć na drążku :");
        input.setMaxPullUps(Int32.Parse(Console.ReadLine()));

        Console.WriteLine("Podaj czas biegu na 5km :");
        input.setFiveKilometersTime(Int32.Parse(Console.ReadLine()));

        Console.WriteLine("Podaj wydolność oddechową :");
        input.setRespiratoryEfficiency(Int32.Parse(Console.ReadLine()));

        status[iterator] = new Rank(input);
        iterator++;
    }
    else
    {
        iterator = 0;
        Console.WriteLine("Gratulacje, oto statystyki postępu z ostatnich 20 treningów :");
        Console.WriteLine("Zalecana jest tygodniowa przerwa, nie przemęczaj się.");
    }
    break;
```

```
case "weight":
    Console.WriteLine("Podaj wagę :");
    input.setWeight(Int32.Parse(Console.ReadLine()));
    status[iterator] = new Rank(input);
    iterator++;
    break;
case "pushups":
    Console.WriteLine("Podaj maksymalną liczbę pompek :");
    input.setMaxPushups(Int32.Parse(Console.ReadLine()));
    status[iterator] = new Rank(input);
    iterator++;
    break;
case "pullups":
    Console.WriteLine("Podaj maksymalną liczbę podciągnięć :");
    input.setMaxPullUps(Int32.Parse(Console.ReadLine()));
    status[iterator] = new Rank(input);
    iterator++;
    break;
case "5km time":
    Console.WriteLine("Podaj czas na 5 km :");
    input.setFiveKilometersTime(Int32.Parse(Console.ReadLine()));
    status[iterator] = new Rank(input);
    iterator++;
    break;
case "resporatory":
    Console.WriteLine("Podaj wydolność oddechową :");
    input.setRespiratoryEfficiency(Int32.Parse(Console.ReadLine()));
    status[iterator] = new Rank(input);
    iterator++;
    break;
default:
    Console.WriteLine("debug - none option selected");
    status[iterator] = new Rank(input);
    iterator++;
    break;
```

W klasie *User* znajduje się menu rankingu użytkownika (punkt 1).

```
1 reference
public void displayCurrentRank()
{
    if (this.iterator == 0)
    {
        Console.WriteLine("Najpierw dodaj dane użytkownika"); //tylko dodanie danych zezwoli nam na otrzymanie wyników
    }
    else
    { //otrzymane wyniki
        Console.WriteLine(" ");
        Console.WriteLine("Obecny poziom użytkownika :"); //poziom ogólny
        Console.WriteLine(" ");

        Console.WriteLine("Twój poziom BMI to : "); // poziom BMI
        Console.WriteLine(status[iterator - 1].BMI);

        Console.WriteLine("Twój poziom siły to : "); // poziom siły
        Console.WriteLine(status[iterator - 1].Strength);

        Console.WriteLine("Twój poziom zwinności to : "); //poziom zwinności
        Console.WriteLine(status[iterator - 1].Agillity);

        Console.WriteLine("Twój poziom wydajności to : "); //poziom wydajności
        Console.WriteLine(status[iterator - 1].Efficiency);
        Console.WriteLine(" ");

        commentator.commentBMIValue(this.status[this.iterator - 1].BMI);
        commentator.commentStrengthValue(this.status[this.iterator - 1].Strength);
        commentator.commentAgillityValue(this.status[this.iterator - 1].Agillity);
        commentator.commentEfficiencyValue(this.status[this.iterator - 1].Efficiency);
        Console.WriteLine(" ");
    }
}
```

oraz menu progresu użytkownika (punkt 4) .

```
public void displayProgress() // menu progresu uzytkownika
{
    double change;

    if (iterator == 0 | iterator == 1)
    {
        //bez wprowadzenia danych treningowych nie poda nam historii
        Console.WriteLine("Nie posiadasz jeszcze historii treningowej");
    }
    else
    {
        double[] BMI = new double[iterator];
        double[] Strength = new double[iterator];
        double[] Agillity = new double[iterator];
        double[] Efficiency = new double[iterator];

        Console.Write("Ilość wykonanych treningow ");
        Console.Write(iterator);
        Console.WriteLine(" ");

        Console.Write("Zmiana poziomu BMI : ");
        change = status[0].BMI - status[iterator - 1].BMI;
        Console.WriteLine(change);
        commentator.commentBMIChange(change);

        Console.Write("Zmiana poziomu sily : ");
        change = status[iterator - 1].Strength - status[0].Strength;
        Console.WriteLine(change);
        commentator.commentStrengthChange(change);

        Console.Write("Zmiana poziomu zwinnosci : ");
        change = status[iterator - 1].Efficiency - status[0].Efficiency;
        Console.WriteLine(change);
        commentator.commentEfficiencyChange(change);

        Console.Write("Zmiana poziomu wydajnosci : ");
        change = status[iterator - 1].Agillity - status[0].Agillity;
        Console.WriteLine(change);
        commentator.commentAgillityChange(change);
    }
}
```

*Public class Input* odpowiada za wprowadzanie danych użytkownika do aplikacji.

```
public class Input
{
    public int weight;
    public int height;
    public int maxPushups;
    public int maxPullUps;
    public int respiratoryEfficiency;
    public int fiveKilometersTime;

    2 references
    public void setWeight(int weightIn)
    {
        weight = weightIn;
    }

    1 reference
    public void setHeight(int heightIn)
    {
        height = heightIn;
    }

    2 references
    public void setMaxPushups(int maxPushupsIn)
    {
        maxPushups = maxPushupsIn;
    }

    2 references
    public void setMaxPullUps(int maxPullUpsIn)
    {
        maxPullUps = maxPullUpsIn;
    }

    2 references
    public void setRespiratoryEfficiency(int respiratoryEfficiencyIn)
    {
        respiratoryEfficiency = respiratoryEfficiencyIn;
    }

    2 references
    public void setFiveKilometersTime(int fiveKilometersTimeIn)
    {
        fiveKilometersTime = fiveKilometersTimeIn;
    }
}
```

*Public class Comentator* odpowiada z prawidłowe analizowanie wprowadzonych danych za pomocą specjalnych wytycznych.  
Analizowanie wartości BMI

```
public class Commentator
{
    1reference
    public void commentBMIValue(double BMI) //jak oceniamy wartosci BMI
    {
        if (BMI > 30 )
        {
            Console.WriteLine("Otyłość: ");
        }
        else if (BMI < 30 && BMI > 25)
        {
            Console.WriteLine("Nadwaga: ");
        }
        else if (BMI <25 && BMI >18)
        {
            Console.WriteLine("Prawidłowa waga: ");
        }
        else if (BMI <18 && BMI>15)
        {
            Console.WriteLine("Wychudzenie: ");
        }
        else if (BMI < 15)
        {
            Console.WriteLine("Wyglodzenie: ");
        }

        Console.WriteLine(BMI);
    }
}
```

Analizowanie wartości siły.

```
1reference
public void commentStrengthValue(double Strength) // jak oceniamy wartosci sily
{
    if (Strength > 50)
    {
        Console.WriteLine("Twoja sila utrzymuje sie na wysokim poziomie: ");
    }
    else if (Strength < 50 && Strength > 25)
    {
        Console.WriteLine("Twoja sila utrzymuje sie na srednim poziomie ");
    }
    else if (Strength < 25 )
    {
        Console.WriteLine("Popracuj jeszcze, twoja sila utrzymuje sie na niskim poziomie: ");
    }

    Console.WriteLine(Strength);
}
```

Analiza wartości zwinności.

```
1 reference
public void commentAgillityValue(double Agillity) //jak oceniamy wartosci zwinnosci
{
    if (Agillity > 200)
    {
        Console.Write("Twoja zwinosc utrzymuje sie na wysokim poziomie: ");
    }
    else if (Agillity < 200 && Agillity > 100)
    {
        Console.Write("Twoja zwinosc utrzymuje sie na tym srednim poziomie: ");
    }
    else if (Agillity < 100)
    {
        Console.Write("Popracuj jeszcze, twoja zwinosc jest na niskim poziomie: ");
    }

    Console.WriteLine(Agillity);
}
```

Analiza wartości wydajności.

```
1 reference
public void commentEfficiencyValue(double Efficiency) // jak oceniamy wartosc wydajnosci
{
    if (Efficiency > 300)
    {
        Console.Write("Twoja wydajnosc utrzymuje sie na wysokim poziomie: ");
    }
    else if (Efficiency < 300 && Efficiency > 150)
    {
        Console.Write("Twoja wydajnosc utrzymuje sie na tym srednim poziomie: ");
    }
    else if (Efficiency < 150)
    {
        Console.Write("Popracuj jeszcze, twoja wydajnosc jest na niskim poziomie: ");
    }
    Console.WriteLine(Efficiency);
}
```

Public class *Commentator* odpowiada także za prawidłowe zinterpretowanie wprowadzonych wartości i skomentowanie ich według specjalnych wytycznych.

Komentowanie zmiany wartości wagi.

```
1 reference
public void commentBMICChange(double change) // informacja
{
    if (change < 0)
    {
        Console.WriteLine("Twoja waga sie zwiekszyla");
    }
    else if (change > 0)
    {
        Console.WriteLine("Twoja waga sie zmniejszyla");
    }
}
```

Komentowanie zmiany wartości siły.

```
1 reference
public void commentStrengthChange(double change)
{
    if (change > 0)
    {
        Console.WriteLine("Twoja sila wzrosla");
    }
    else if (change < 0)
    {
        Console.WriteLine("Twoja sila zmalała");
    }
}
```

Komentowanie zmiany wartości zwinności.

```
1 reference
public void commentAgillityChange(double change)
{
    if (change > 0)
    {
        Console.WriteLine("Twoja zwinosc wzrosła");
    }
    else if (change < 0)
    {
        Console.WriteLine("Twoja zwinosc zmalała");
    }
}
```

Komentowanie zmiany wartości zręczności.

```
1 reference
public void commentEfficiencyChange(double change)
{
    if (change > 0)
    {
        Console.WriteLine("Twoja zrecznosc wzrosła");
    }
    else if (change < 0)
    {
        Console.WriteLine("Twoja zrecznosc zmalała");
    }
}
```

*Public class Rank* ukazuje specjalne wzory na podstawie których wyznaczane są wymagane wartości np. BMI.

```
public class Rank //sposoby obliczenia poszczegolnych wartosci
{
    public Input input;

    public double BMI;
    public double Strength;
    public double Agillity;
    public double Efficiency;

    7 references
    public Rank(Input ins)
    {
        this.input = ins;

        BMI = getBMI(input.weight, input.height);
        Strength = getStrength(input.maxPushups, input.maxPullUps);
        Agillity = getAgillity(input.fiveKilometersTime, input.height);
        Efficiency = getEfficiency(input.respiratoryEfficiency);
    }

    1 reference
    double getBMI(int weight, int height)
    {
        double BMI = ((double)weight / (((double)height / (double)100) * ((double)height / (double)100)));

        return BMI;
    }

    1 reference
    double getStrength(int maxPushUps, int maxPullUps)
    {
        int pushUpWeight = 1;
        int pullUpWeight = 2;

        return pushUpWeight * maxPushUps + pullUpWeight * maxPullUps;
    }

    1 reference
    double getAgillity(int fiveKilometersTime, int height)
    {
        int heightWeight = 1;
        int fiveKilometersTimeWeight = 2;

        return fiveKilometersTimeWeight * (1 / (double)fiveKilometersTime) + heightWeight * height;
    }
}
```

```
1 reference
double getEfficiency(int respiratoryEfficiency)
{
    int agillityWeight = 1;
    int strengthWeight = 2;
    int respiratoryEfficiencyWeight = 1;

    double efficiency = (this.Agillity * agillityWeight + this.Strength * strengthWeight) + respiratoryEfficiencyWeight * respiratoryEfficiency;
    return efficiency;
}
```



## 6. INSTRUKCJA UŻYTKOWANIA APLIKACJI

Po włączeniu aplikacji wyświetli nam się menu główne składa się ono z 6 punktów.

```
C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria op  
1. - Wyświetl ranking użytkownika  
2. - Generuj plan treningowy użytkownika  
3. - Dodaj wyniki sprawdzianu  
4. - Wyświetl progres użytkownika  
5. - Wyczyść widok  
6. - Wyczyść historię użytkownika  
_
```

Na początku musimy wybrać **punkt 3**, w którym to wprowadzimy wyniki naszego treningu. W punkcie 3 ukaże nam się menu dodania wyników sprawdzianu.

```
C:\Users\Michał Góra\Desktop\6 SEMESTR\I  
1 - Podaj wszystkie dane  
2 - Podaj wagę  
3 - Podaj wyniki testu pompek  
4 - Podaj wyniki testu na drążki  
5 - Podaj czas biegu na 5 km
```

W punkcie 3, wybierając **punkt 1** podajemy wszystkie dane począwszy od wzrostu, a skończywszy na wydolności oddechowej. Możemy także podać wagę, wynik testu pompek lub na drążku oraz czas biegu na 5km wybierając pozostałe **punkty 2 - 5**.

```
C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria oprogramow  
1 - Podaj wszystkie dane  
2 - Podaj wagę  
3 - Podaj wyniki testu pompek  
4 - Podaj wyniki testu na drążki  
5 - Podaj czas biegu na 5 km  
1  
Podaj wzrost :190  
Podaj wagę :100  
Podaj maksymalną liczbę pompek :10  
Podaj maksymalną liczbę podciągnięć na drążku :10  
Podaj czas biegu na 5km :10  
Podaj wydolność oddechową :100
```

Aby wyświetlić ranking użytkownika musimy wybrać **punkt 1** oraz wcześniej wprowadzić w punkcie 3 wyniki chociaż jednego sprawdzianu. W przypadku kiedy tych danych nie wprowadzimy, wybierając punkt 1 na ekranie pojawi nam się komunikat :

```
C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria og
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
1
Najpierw dodaj dane użytkownika
```

Jeśli dane zostaną poprawnie wprowadzone, wybierając pkt 1 zostaniemy przekierowani do rankingu użytkownika. W rankingu użytkownika możemy się dowiedzieć jaki jest nasz aktualny poziom BMI, siły, zwinności i wydajności. Następnie za pomocą specjalnych wytycznych wartości te są komentowane.

```
Obecny poziom użytkownika :

Twój poziom BMI to : 22,1606648199446
Twój poziom siły to : 60
Twój poziom zwinności to : 190,08
Twój poziom wydajności to : 510,08000000000004

Komentarze do wartości
Prawidłowa waga: 22,1606648199446
Twoja siła utrzymuje się na wysokim poziomie: 60
Twoja zwinność utrzymuje się na tym średnim poziomie: 190,08
Twoja wydajność utrzymuje się na wysokim poziomie: 510,08000000000004
```

W **punkcie 2** możemy wygenerować odpowiedni dla nas plan treningowy. Podobnie jak w punkcie 1 by plan został wygenerowany trzeba najpierw wprowadzić w punkcie 3 wyniki sprawdzianu. Generując plan treningowy możemy wybrać trzy poziomy trudności oraz trzy rodzaje treningu.

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
2
Podaj poziom trudności (easy/normal/hard)
easy
Podaj cel treningu (strength/agility/general)
general
```

W zależności od wybranej wariantu na ekranie pojawi nam się komunikat jak powinien wyglądać nasz trening. W naszym przypadku wybierając poziom trudności “easy” oraz cel treningu “general” otrzymujemy następujący plan:

```
C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżyn  
Wybrano trening ogólny  
Dzień x - 5 serii biegu po 5 minut
```

W **punkcie 4** możemy zobaczyć nasz progres jaki wykonaliśmy. By progres został obliczony potrzebujemy minimum dwa razy wprowadzić wyniki sprawdzianu w punkcie 3, jeśli to się nie stanie wyświetli nam się komunikat:

```
C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria o  
1. - Wyświetl ranking użytkownika  
2. - Generuj plan treningowy użytkownika  
3. - Dodaj wyniki sprawdzianu  
4. - Wyświetl progres użytkownika  
5. - Wyczyść widok  
6. - Wyczyść historię użytkownika  
4  
Nie posiadasz jeszcze historii treningowej
```

Jeśli dane zostaną wprowadzone na ekranie pojawią nam się wartości w poszczególnych dziedzinach z komentarzami oraz ilość wykonanych treningów.

```
1. - Wyświetl ranking użytkownika  
2. - Generuj plan treningowy użytkownika  
3. - Dodaj wyniki sprawdzianu  
4. - Wyświetl progres użytkownika  
5. - Wyczyść widok  
6. - Wyczyść historię użytkownika  
4  
Ilość wykonanych treningow 2  
Zmiana poziomu BMI : 1,10803324099723  
Twoja waga sie zmniejszyla  
Zmiana poziomu sily : 60  
Twoja sila wzrosła  
Zmiana poziomu zwinnosci : 110,00444444444452  
Twoja zrecznosc wzrosła  
Zmiana poziomu wydajnosci : 0,004444444444445935  
Twoja zwinnosc wzrosła
```

**Punkt 5** odpowiada na czyszczenie widoku konsoli.

C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria oprogramowania

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
4
Ilość wykonanych treningów 2
Zmiana poziomu BMI : 1,10803324099723
Twoja waga się zmniejszyła
Zmiana poziomu siły : 60
Twoja siła wzrosła
Zmiana poziomu zwinności : 110,00444444444452
Twoja zreźność wzrosła
Zmiana poziomu wydajności : 0,004444444444445935
Twoja zwinność wzrosła
```

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
```

5



C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria oprogramowania

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
```

**Punkt 6** odpowiada za czyszczenie całej historii użytkownika. Po usunięciu wszystkich danych nie będziemy mieli już do nich żadnego dostępu i wrócimy do punktu wyjścia, w którym musimy ponownie wprowadzić dane w punkcie 3.

C:\Users\Michał Góra\Desktop\6 SEMESTR\Inżynieria

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
```

6

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
```

1

Najpierw dodaj dane użytkownika

```
1. - Wyświetl ranking użytkownika
2. - Generuj plan treningowy użytkownika
3. - Dodaj wyniki sprawdzianu
4. - Wyświetl progres użytkownika
5. - Wyczyść widok
6. - Wyczyść historię użytkownika
```

1

## 7. WNIOSKI

- Aplikacja spełnia przedstawione na początku założenia projektowe.
- Aplikacja przedstawia prosty sposób jak powinno się programować za pomocą wstrzykiwania zależności.
- Użytkowanie aplikacji nie powinno sprawiać większych trudności, nawet dla osób nie mających do czynienia wcześniej z tego typu aplikacjami.
- Aplikacja napisana z wykorzystaniem języka C#.
- Wzorce projektowe pomagają w programowaniu aplikacji.
- Środowisko Visual Studio w prosty sposób ułatwia tworzenie programów.