Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»

(национальный исследовательский университет)

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)_____

О т ч е т

по домашнему заданию № 3

Название домашнего задания:

Часть 1. Простые объекты

Часть 2. Наследование

Часть 3. Композиция

Дисциплина: Основы программирования

Студент гр. ИУ6-13Б _____02.11.2022___ __В. К. Залыгин___

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Часть 1. Простые объекты

Цель работы

Изучить принципы построения простых объектов.

Задание

Описать объект, включающий заданные поля и методы. Написать программу, которая создает объект и тестирует его методы.

Объект – окружность. Параметры: координаты центра, радиус. Методы: процедура инициализации полей, процедура вывода на экран значения полей объекта и функция, определяющая, находится ли некоторая точка с координатами x,y внутри окружности.

Проект программы

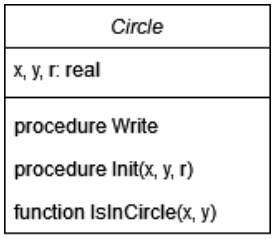

Рисунок 1 - диаграмма класса

Текст программы

```
program hw31;
uses Math;

type Circle = object
  private
    x, y, r: real;
  public
    procedure Init(new_x: real; new_y: real; new_r: real);
    procedure Write();
    function IsInCircle(point_x: real; point_y: real): boolean;
end;

procedure Circle.Init(new_x: real; new_y: real; new_r: real);
begin
  x := new_x;
  y := new_y;
  r := new_r;
end;

function Circle.IsInCircle(point_x: real; point_y: real): boolean;
begin
  Result := power(point_x - x, 2) + power(point_y - y, 2) <= power(r, 2);
end;

procedure Circle.Write;
begin
  WriteLn('Circle {x: ', x:4:1, ' y: ', y:4:1, ' r: ', r:4:1, '}');
end;

var a: Circle;
begin
  a.Init(1, 1, 1);
  WriteLn('Принадлежит ли точка 0 0 кругу: ', a.IsInCircle(0, 0));
  WriteLn('Принадлежит ли точка 0 1 кругу: ', a.IsInCircle(0, 1));
  WriteLn('Принадлежит ли точка 5 -1 кругу: ', a.IsInCircle(5, -1));
  a.Write;

  ReadLn;
end.
```

Рисунок 2 - код программы

Тестовые данные

```
Принадлежит ли точка 0 0 кругу: FALSE
Принадлежит ли точка 0 1 кругу: TRUE
Принадлежит ли точка 5 -1 кругу: FALSE
Circle {x:  1.0 y:  1.0 r:  1.0}
```

Рисунок 3 - результат работы программы

Вывод

Был изучен механизм создания простых объектов.

Часть 2. Наследование

Цель работы

Изучить и применить механизм наследования.

Задание

Разработать и реализовать иерархию классов для описанных объектов предметной области, используя механизмы наследования.

Объект – шоколадное изделие. Поля: название, масса, энергетическая ценность на 100 грамм продукта. Методы: процедура инициализации, процедура вывода информации об объекте на экран, функция определения энергетической ценности изделия.

Объект – шоколадная плитка. Поля: название, масса, энергетическая ценность на 100 грамм, число долек в плитке. Методы: процедура инициализации, процедура вывода информации об объекте на экран, функция определения энергетической ценности плитки и функция определения энергетической ценности одной дольки.

Проект программы

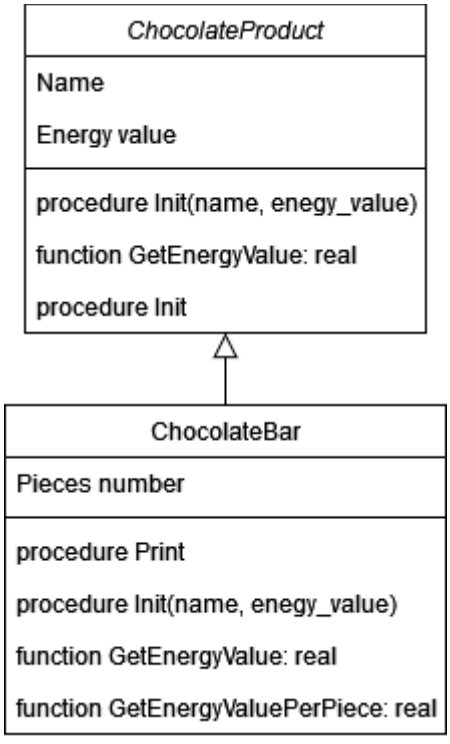

Рисунок 4 - диаграмма классов

Текст программы

```
program hw32;

type ChocolateProduct = object
  private
    name: string;
    energy_value: real;
  public
    procedure Init(new_name: string; new_energy_value: real);
    function GetEnergyValue(): real;
    procedure Print;
end;

procedure ChocolateProduct.Init(new_name: string; new_energy_value: real);
begin
  name := new_name;
  energy_value := new_energy_value;
end;

function ChocolateProduct.GetEnergyValue: real;
begin
  Result := energy_value
end;

procedure ChocolateProduct.Print;
begin
  WriteLn('ChocolateProduct { name: ', name, ' energy_value: ', energy_value:4:1, ' }')
end;
```

Рисунок 5 - код класса ChocolateProduct

```
type ChocolateBar = object(ChocolateProduct)
  private
    pieces_number: integer;
  public
    procedure Init(new_name: string; new_energy_value: real; new_pieces_number: integer);
    procedure Print;
    function GetEnergyValuePerPiece: real;
end;

procedure ChocolateBar.Init(new_name: string; new_energy_value: real; new_pieces_number: integer);
begin
  inherited Init(new_name, new_energy_value);
  pieces_number := new_pieces_number;
end;

procedure ChocolateBar.Print;
begin
  WriteLn('ChocolateBar {');
  Write('  Base: ');
  inherited Print;
  WriteLn('  pieces_number: ', pieces_number, ' }');
end;

function ChocolateBar.GetEnergyValuePerPiece: real;
begin
  Result := energy_value / pieces_number
end;

var a: ChocolateBar;
begin
  a.Init('mars', 93.2, 2);
  a.Print;
  WriteLn('Energy value: ', a.GetEnergyValue:4:1);
  WriteLn('Energy value per piece: ', a.GetEnergyValuePerPiece:4:1);

  ReadLn;
end.
```

Рисунок 6 - класс ChocolateBar и тестирующая программа

Тестовые данные

```
ChocolateBar {
  Base: ChocolateProduct { name: mars energy_value: 93.2 }
  pieces_number: 2 }
Energy value: 93.2
Energy value per piece: 46.6
```

Рисунок 7 - результат работы программы

Вывод

Был изучен и применён механизм наследования.

Часть 3. Композиция

Цель работы

Изучить и применить механизм композиции.

Задание

Разработать и реализовать диаграмму классов для описанных объектов предметной области, используя механизмы композиции.

Объект – скаковая лошадь. Параметры: кличка и массив рекордов, содержащий 5 лучших результатов, показанных лошадью на скачках. Методы: процедура инициализации полей, процедура вывода на экран значений полей, и функция, определяющая среднее время, показанное лошадью.

Объект – конюшня, для которой определен перечень лошадей (количество лошадей и данные о каждой из них). Объект умеет инициализировать свои поля, выводить на экран их значения и определять среднее время лошадей всей конюшни.

Проект программы

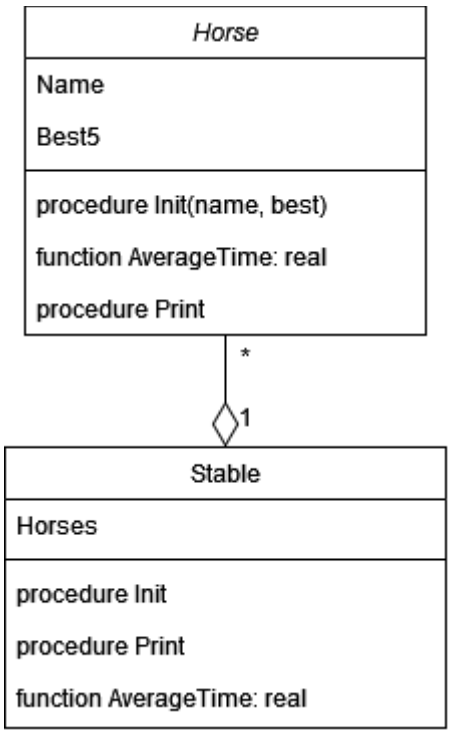

Рисунок 8 - диаграмма классов

Текст программы

```
program hw33;

type Horse = object
  private
    name: String;
    best: array[1..5] of real;
  public
    constructor Init(nname: string; best1: real; best2: real; best3: real; best4: real; best5: real);
    procedure Print;
    function AverageTime: real;
end;

constructor Horse.Init(nname: string; best1: real; best2: real; best3: real; best4: real; best5: real);
begin
  name := nname;
  best[1] := best1;
  best[2] := best2;
  best[3] := best3;
  best[4] := best4;
  best[5] := best5;
end;

type thorses = array of horse;

procedure Horse.Print;
var i: integer;
begin
  WriteLn('Horse { name: ', name, ' best:');
  for i := 1 to 5 do
    WriteLn('time: ', best[i]:4:1);
  WriteLn('}');
end;

function Horse.AverageTime: real;
var i: integer;
    sum: real;
begin
  sum := 0;
  for i := 1 to 5 do
    sum += best[i];
  Result := sum / 5;
end;
```

Рисунок 9 - объявление класса Horse

```
type Stable = object
  private
    horses: thorses;
  public
    constructor Init(nhorses: thorses);
    procedure Print;
    function AverageTime: real;
end;

constructor Stable.Init(nhorses: thorses);
begin
  horses := nhorses;
end;

procedure Stable.Print;
var i: integer;
begin
  WriteLn('Stable {');
  for i := 0 to high(horses) do
    horses[i].Print;
  WriteLn('}');
end;

function Stable.AverageTime: real;
var i: integer;
    sum: real;
begin
  sum := 0;
  for i := 0 to high(horses) do
    sum += horses[i].AverageTime;
  Result := sum / (high(horses)+1);
end;

var my_stable: Stable;
    my_horses: thorses;
begin
  SetLength(my_horses, 5);
  my_horses[0].Init('1', 1.0, 1.1, 1.2, 1.3, 1.4);
  my_horses[1].Init('2', 2.0, 2.1, 2.2, 2.3, 2.4);
  my_horses[2].Init('3', 3.0, 3.1, 3.2, 3.3, 3.4);
  my_horses[3].Init('4', 4.0, 4.1, 4.2, 4.3, 4.4);
  my_horses[4].Init('5', 5.0, 5.1, 5.2, 5.3, 5.4);

  my_stable.Init(my_horses);

  WriteLn('Horse 2: ');
  my_horses[1].Print;
  WriteLn('Stable: ');
  my_stable.Print;

  WriteLn('Average time of horse 2: ', my_horses[1].AverageTime:4:1);
  WriteLn('Average time of all horses: ', my_stable.AverageTime:4:1);
  ReadLn;
end.
```

Рисунок 10 - класс Stable и тестирующая программа

Тестовые данные

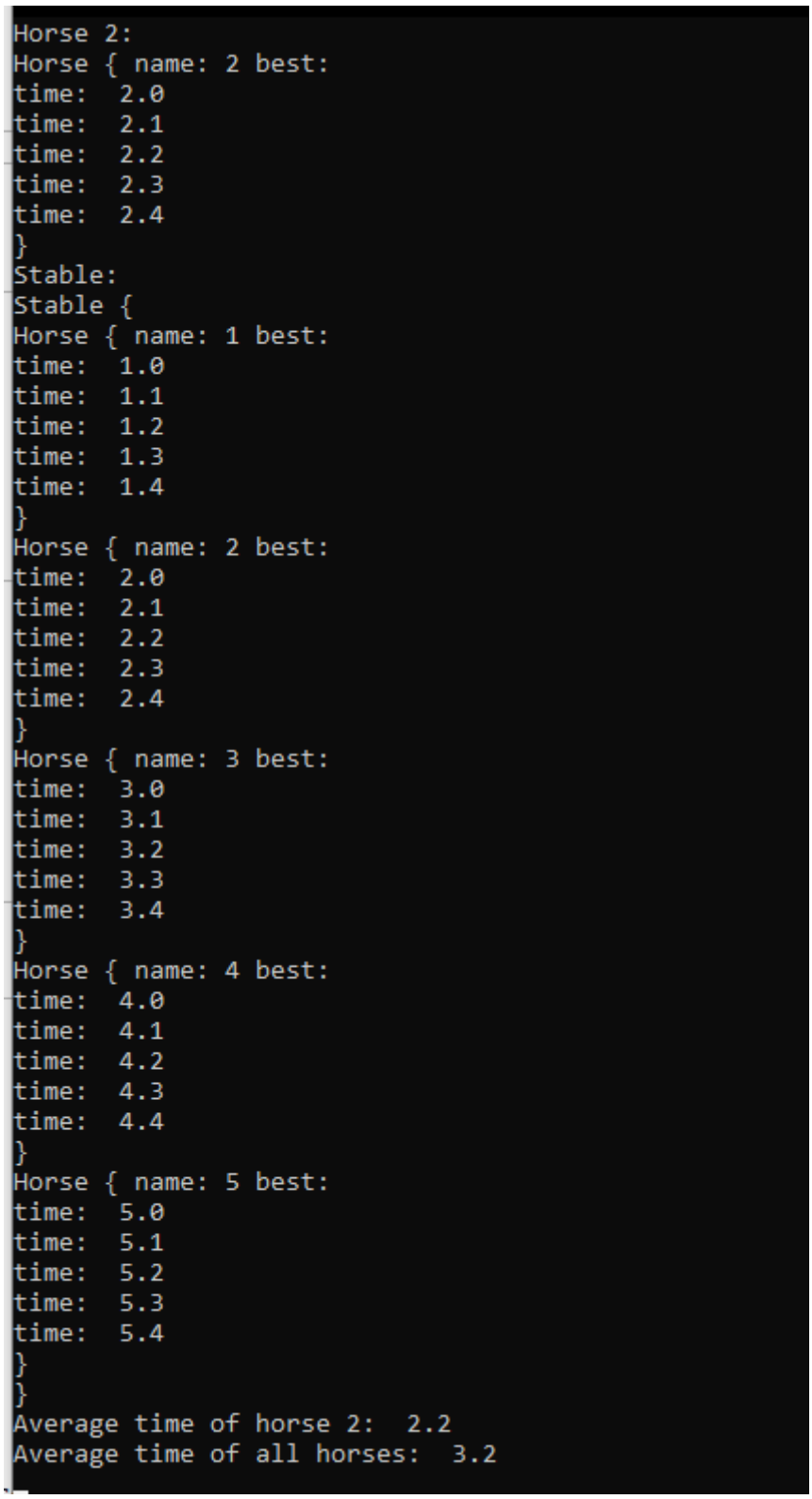

Рисунок 11 - результат тестирования программы

Вывод

Был изучен и применён механизм композиции. Реализована программа, удовлетворяющая требованиям задания.