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 лабораторной работы: _____
Дисциплина: Основы программирования

Студент гр. ИУ6-12Б _____ **С.В.Астахов**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____
(Подпись, дата) (И.О. Фамилия)

Москва, 2019

I вариант

Задание 1

Описать класс, включающий заданные поля и методы. Протестировать все методы. Объект – слово. Поля: строка, содержащая слово, и длина слова. Методы: процедура инициализации полей объекта; функция, определяющая количество гласных букв в слове; процедура вывода информации об объекте на экран.

Код программы:

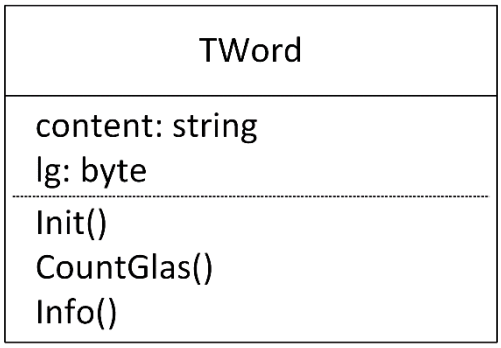
```
program project1;

type
 TWord = object
 private
  content: string;
  lg: byte;
 public
  procedure Init(sData: string);
  function CountGlas: byte;
  procedure Info;
  {non-standarted, but needed in task}
 end;

 procedure TWord.Init(sData: string);
 begin
  Self.content := sData;
  Self.lg := length(sData);
 end;

 function TWord.CountGlas: byte;
 type
  glasnType = set of char;
 var
  glasnSet: glasnType;
  i, cnt: byte;
 begin
  cnt := 0;
  glasnSet := ['e', 'y', 'u', 'i', 'o', 'a', 'j'];
  for i := 1 to length(Self.content) do
   if Self.content[i] in glasnSet then
    cnt := cnt + 1;
```

```
    Result := cnt;
  end;

  procedure TWord.Info;
  begin
    writeln('= Object info =');
    writeln('word: ', Self.content);
    writeln('length: ', Self.lg);
    writeln('Glasnih in word: ', CountGlas);
  end;

 var
  wrd: TWord;
  inputS: string;
 begin
  writeln('Input string to make an object');
  readln(inputS);
  wrd.Init(inputS);
  wrd.Info();
  readln;
end.
```

Тесты

Входные данные	Ожидаемые выходные данные	Выходные данные
vorona	= Object info = word: vorona length: 6 Glasnih in word: 3	= Object info = word: vorona length: 6 Glasnih in word: 3
program	= Object info = word: program length: 7 Glasnih in word: 2	= Object info = word: program length: 7 Glasnih in word: 2
№;$	= Object info = word: №;$ length: 3 Glasnih in word: 0	= Object info = word: №;$ length: 3 Glasnih in word: 0

Диаграмма классов



Задание 2

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

Объект – шахматная фигура. Поля: цвет фигуры (белый, черный) и координаты клетки шахматной доски, на которой она стоит. Методы: процедура инициализации, процедура вывода информации о фигуре на экран и функция, определяющая, совпадает ли цвет фигуры с цветом клетки, на которой она стоит.

Объект – слон (шахматная фигура). Поля: цвет фигуры и координаты клетки, на которой она стоит. Методы: процедура инициализации, процедура вывода информации о фигуре на экран и функция, которая по координатам другой клетки шахматной доски определяет, находится ли она под ударом слона.

Код программы:

```
program project1;

type
  TFigure = object
  private
    fColor: boolean;
    x, y: byte;
  public
    procedure Init(clr: string; i, j: byte);
    function ColorEq: boolean;
    procedure Info;
  end;

  procedure TFigure.Init(clr: string; i, j: byte);
```

```
 begin
  if clr = 'black' then
    Self.fColor := True
  else
    Self.fColor := False;
  Self.x := i;
  Self.y := j;
 end;

 function TFigure.ColorEq: boolean;
 begin
  Result := ((((x + y) mod 2) = 0) = Self.fColor);
  {1;1 is black}
 end;

 procedure TFigure.Info;
 begin
  writeln('= Figure info =');
  writeln('color: ');
  if Self.fColor then
    writeln('black')
  else
    writeln('white');
  writeln('Coords: ', x: 3, y: 3);
  if Self.ColorEq() then
    writeln('Color of figure and field matches')
  else
    writeln('Color of figure and field dont match');
 end;

type
 TSlon = object(TFigure)
  function Danger(fi, fg: byte): boolean;
 end;

 function TSlon.Danger(fi, fg: byte): boolean;
 var
  beaten: boolean;
  diff: integer;
 begin
  beaten := False;
  diff := fi - Self.x;
```

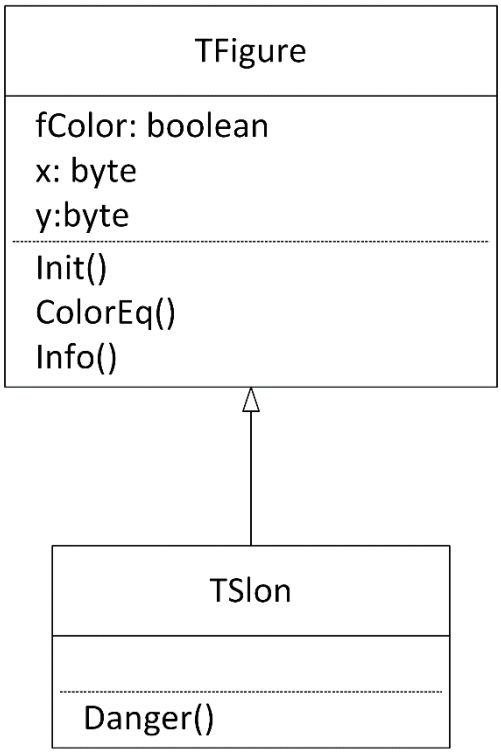
```
    {writeln(x: 3, y: 3, fi: 3, fg: 3, diff: 3);}
    if ((fg = Self.y + diff) or (fg = Self.y - diff)) then
      beaten := True;
    Result := beaten;
  end;

var
  xInp, yInp, xTarget, yTarget: byte;
  fClr: string;
  oneSlon: TSlon;
begin
  writeln('Enter color, "black" ot "white"');
  fClr := 'wait';
  while ((fClr <> 'black') and (fClr <> 'white')) do
    readln(fClr);
  writeln('Enter x,y in [1;8] (one string input)');
  xInp := 13;
  yInp := 13;
  while not ((xInp > 0) and (xInp < 9) and (yInp > 0) and (yInp < 9)) do
    readln(xInp, yInp);
  oneSlon.Init(fClr, xInp, yInp);
  oneSlon.Info;
  writeln;
  writeln('Enter x,y of TARGET in [1;8] (one string input)');
  xTarget := 13;
  yTarget := 13;
  while ((not ((xTarget > 0) and (xTarget < 9) and (yTarget > 0) and
(yTarget < 9))) or
      ((xTarget = xInp) and (yTarget = yInp))) do
    readln(xTarget, yTarget);

  if oneSlon.Danger(xTarget, yTarget) then
    writeln('Figure can be beaten')
  else
    writeln('Figure cannot be beaten');
  writeln;
  writeln('<press any key>');
  readln;

end.
```

Тесты

Входные данные	Ожидаемые выходные данные	Выходные данные
Ghjk Black 9 9 2 2 3 3	= Figure info = color: black Coords: 2 2 Color of figure and field matches Figure can be beaten	= Figure info = color: black Coords: 2 2 Color of figure and field matches Figure can be beaten
White 1 2 1 1	= Figure info = color: white Coords: 1 2 Color of figure and field matches Figure cannot be beaten	= Figure info = color: white Coords: 1 2 Color of figure and field matches Figure cannot be beaten
White 12 23	= Figure info = color: white Coords: 1 2 Color of figure and field matches Figure can be beaten	= Figure info = color: white Coords: 1 2 Color of figure and field matches Figure can be beaten

*Ввод повторяется, пока не будут введены корректные данные

Диаграмма классов



Задание 3

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

Объект – товарный вагон. Поля: грузоподъемность вагона, масса находящегося в нем груза и тип груза. Методы: процедура инициализации, процедура вывода параметров вагона на экран, функция расчета процента заполнения вагона и функции, возвращающие значение каждого поля по запросу.

Объект – товарный поезд. Содержит число вагонов в поезде (по умолчанию 0) и сами вагоны. Методы объекта должны позволять: прицепить к хвосту поезда вагон с заданными параметрами, отцепить последний вагон, вывести на экран информацию обо всех вагонах, рассчитать общую грузоподъемность поезда, получить количество вагонов, заполненных более чем наполовину.

Код программы:

```
program project1;

{
TODO realize class Poezd
Code main program
```

```
get vagon param by id
}

type
 TVagon = object
 private
  gruzpd: integer;
  massa: integer;
  typeG: string;
 public
  procedure Init(maxGM, gm: integer; typeOfG: string);
  procedure Info;
  function percentG: real;
  function getParam(Id: byte): string;
 end;

 procedure TVagon.Init(maxGM, gm: integer; typeOfG: string);
 begin
  Self.gruzpd := maxGM;
  Self.massa := gm;
  Self.typeG := typeOfG;
 end;

 procedure TVagon.Info;
 begin
  writeln('gruzopod: ', Self.gruzpd: 6, ' weight on board: ',
   Self.massa: 6, ' type of gruz: ',
   Self.typeG: 10, ' % zaniatosti: ', Self.percentG: 6: 2);
 end;

 function TVagon.percentG: real;
 begin
  Result := 100 * (Self.massa / Self.gruzpd);
 end;

 function TVagon.getParam(id: byte): string;
 var
  s: string;
 begin
  case id of
   1: str(Self.gruzpd, s);
```

```
    2: str(Self.massa, s);
    3: s := Self.typeG;
    else
      s := 'no data';
  end;
  Result := s;
 end;

type
 TPoezd = object
 private
  vagons: array[1..30] of TVagon;
  n: byte;
 public
  procedure resetP;
  procedure addVagon(maxGM, gm: integer; typeOfG: string);
  procedure removeVagon;
  procedure Info;
  function SumGruzpod: integer;
  function NumOfHalfFull: integer;

  {code that obj}
  {programs to add, delete,info, gruzopod, kolvo more than half
full}
 end;

 procedure TPoezd.resetP;
 begin
  Self.n := 0;
 end;

 procedure TPoezd.addVagon(maxGM, gm: integer; typeOfG:
string);
 var
  newVagon: TVagon;
 begin
  Self.n := Self.n + 1;
  newVagon.Init(maxGM, gm, typeOfG);
  Self.vagons[n] := newVagon;
 end;

 procedure TPoezd.removeVagon;
```

```
begin
  if Self.n > 0 then
    Self.n := Self.n - 1;
end;

procedure TPoezd.Info;
var
  i: byte;
begin
  if n = 0 then
    writeln('The train doesnt exist(0 vagons)');
  for i := 1 to Self.n do
    Self.vagons[i].Info;
end;

function TPoezd.sumGruzpod: integer;
var
  i: byte;
  sum: integer;
begin
  sum := 0;
  for i := 1 to Self.n do
    sum := sum + Self.vagons[i].gruzpd;
  Result := sum;
end;

function TPoezd.NumOfHalfFull: integer;
var
  cnt, i: byte;
begin
  cnt := 0;
  for i := 1 to Self.n do
    if Self.vagons[i].massa > (Self.vagons[i].gruzpd div 2) then
      cnt := cnt + 1;
  Result := cnt;
end;

var
  thomas: TPoezd;
  optionId: byte;
  par1, par2: integer;
  par3: string;
```

```
begin
 thomas.resetP();
 writeln('Chose option: 0-Exit 1-Add 2-Remove 3-info 4-
SumGruzopod 5-HalfFullVagons');
 readln(optionId);
 while optionId <> 0 do
 begin
  case optionId of
   1:
   begin
    writeln('Enter gruzopod and mass: ');
    par1 := -1;
    par2 := -1;
    while not ((par1 > 0) and (par2 > 0) and (par1 > par2)) do
     readln(par1, par2);
    writeln('Enter gruz type: ');
    readln(par3);
    thomas.addVagon(par1, par2, par3);
   end;
   2: thomas.removeVagon();
   3: thomas.Info();
   4: writeln(thomas.sumGruzpod);
   5: writeln(thomas.NumOfHalfFull);
   else
    writeln('no such command');
  end;
  writeln('Chose option: 0-Exit 1-Add 2-Remove 3-info 4-
SumGruzopod 5-HalfFullVagons');
  readln(optionId);
 end;
 writeln;
 writeln('<press any key>');
 readln;
end.
```

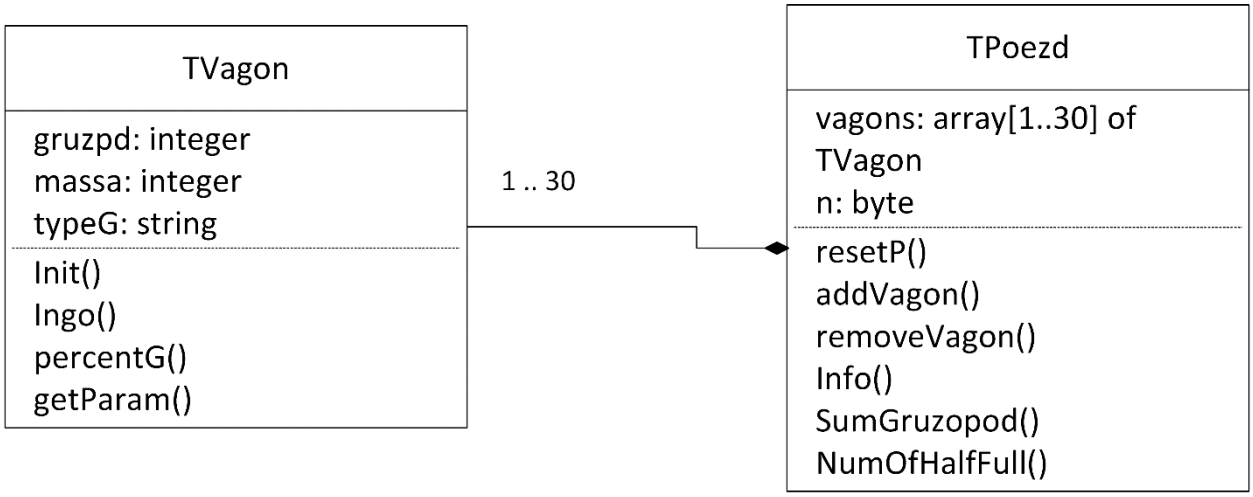
Тесты

Входные данные	Ожидаемые выходные данные	Выходные данные
3 4 5	The train doesnt exist(0 vagons) 0 0	The train doesnt exist(0 vagons) 0 0

0	<press any key>	<press any key>
1 13 2 Steel 1 12 7 Wood 1 13 9 Other 3 4 5 0	 gruzopod: 13 weight on board: 2 type of gruz: steel % zaniatosti: 15.38 gruzopod: 12 weight on board: 7 type of gruz: wood % zaniatosti: 58.33 gruzopod: 13 weight on board: 9 type of gruz: other % zaniatosti: 69.23 38 2 <press any key>	 gruzopod: 13 weight on board: 2 type of gruz: steel % zaniatosti: 15.38 gruzopod: 12 weight on board: 7 type of gruz: wood % zaniatosti: 58.33 gruzopod: 13 weight on board: 9 type of gruz: other % zaniatosti: 69.23 38 2 <press any key>
1 13 4 Auto 1 10 5 Unkonown 2		

3 0	gruzopod: 13 weight on board: 4 type of gruz: auto % zaniatosti: 30.77 <press any key>	gruzopod: 13 weight on board: 4 type of gruz: auto % zaniatosti: 30.77 <press any key>

Диаграмма классов:



Вывод:

- ООП позволяет лучше структурировать код программы и реализовать операции наследования и композиции, что значительно упрощает процесс разработки сложных программ с большим числом подпрограмм и сложными объектами