Nesne Yönelimli Programlama - Dönem Sonu Sınavı 13.01.2019

1) Arka sayfadaki kodda verilen iki farklı ad alanı içerisinde tanımlı struct ve class yapıları bulunmaktadır. **space1** ad alanında verilen bir yılın artık yıl olup olmadığı **isLeap()** metodu ile ve yılın kaçıncı yüzyıla ait olduğu bilgisi **getCentury()** metodu ile hesaplanmaktadır. **space2** ad alanında da yine yılın hangi yüzyıla ait olduğu hesap edilmek istenmektedir. (bir yılın artık yıl olması için o yılın 4'ün katı olması, 100'ün katı olmaması ve 400'ün katı olması gerekmektedir, yine bir yılın yüzyılı hesap edilirken o yılın kaçıncı yüzyıl diliminde olduğu hesap edilmektedir. örneğin;1500 yılı 15. yüzyıl, 1501 yılı ise 16. yüzyıldır). (***1*PÇ2***)
   a) Kodda verilen **space1** ad alanındaki **isLeap()** ve **getCentury()** metodlarını gerçekleyiniz (10p).
   b) **space2** ad alanındaki **getCentury()** metodunda boş yerleri istenen işlevleri yerine getirecek şekilde tamamlayınız (5p).
   c) Ana fonksiyonda verilen **line1** satırında **y1** isimli yılın hangi yüzyılda olduğunu ve **line2** satırında **y** isimli yılın artık yıl olup olmadığını ekrana yazdıracak şekilde boşlukları doldurunuz ve kod çalıştırıldığında ekran çıktısının ne olacağını yazınız (10p)

Cevap 1

a
yazılan kodun doğru çalışmasına dikkat edilir, fonksiyon prototipi soruda verilmiştir.

```
bool space1::isLeap(){    //  5p
  int div4 = year%4;
  int div100 = year%100;
  int div400 = year%400;
  bool _isLeap = false;
  if(div4==0) _isLeap = true;
  if(div100==0) _isLeap = false;
  if(div400==0) _isLeap =true;
  return _isLeap;
}
int space1::getCentury(){    // 5p
  return (year%100!=0)?(year/100+1):(year/100);
}
```

b
yazılan kodda y değişkeninin elemanlarına erişim, int-double tür dönüşümleri önemli, if vb. program denetim deyimleri kullanılmışsa veya gereksiz yere çok fazla değişken kullanılmışsa puanlar yarı yarıya düşürülür. nesne erişim dikkate alınmamışsa puan alınmaz.

```
int space2::getCentury(Year y){
                          //   4p              1p
  return (y.year%100!=0)?(y.year/100+1):(y.year/100);
}
}
```

c
erişim belirteçleri önemli, direk aynı ifadeleri yazması beklenir, ekran görüntüsü ile cout içerisindeki kodlar tutarlı değilse ve diğerleri doğru ise puan yarıya iner

```
                        //  4p
std::cout<< ......space2::getCentury(y1)..... <<std::endl; // line 1, y1 yüzyıl
                      //  2p
std::cout<< .....y.isLeap()..... <<std::endl; // line 2, y isLeap

ekran → 4 puan → 2 + 2
21
0 (veya false)
```

2) Sizden üç boyutlu bir uzayda tutulacak noktalar üzerinde işlemler yapabileceğiniz bir program yazmanız beklenmektedir. Bu program **point1D** anasınıfından türetilen **point2D** ve **point3D** alt sınıflarını içeren bir hiyerarşiye sahip sınıfları içermektedir. (***0.5*PÇ2+0.5*PÇ3***)
   a) Point hiyerarşisindeki bütün sınıfları gerçekleyerek gerekli yapıcıları, get ve set fonksiyonlarını gerçekleyiniz (15p)
   b) Point hiyerarşisindeki sınıflardan türetilecek nesneleri işleyecek ve nesnelerin toplam uzunluğunu bulacak bir fonksiyonu aşağıda verilen prototipe göre gerçekleyiniz. (15p)

```
double calculateTotalLength(point1d *pointsVector[], int numOfPoints){
 ...
}
int main()
{
  point3d pointEx3d(3, 3, 3);
  point2d pointEx2d(3, 3);
  point1d pointEx1d(3);
```

```
    point1d *points[5];

    points[0] = &pointEx1d;  points[1] = &pointEx2d;  points[2] = &pointEx3d;

    cout << "Total Length= " <<  calculateTotalLength(points, 3) << endl;
    return 0;
}
```

Cevap 2

a
nesne yönelimli programlama ilkelerine (data abstraction) uyulmamışsa puan verilmez.
miras hiyerarşisi önemli, x, y, z değişkenleri doğru yerde olmalı → 5p
polimorphizm için sanal fonk. tanımlamak önemli → 3p
get ve set fonk. doğru çalışacak, data private veya protected olacak → 3p
doğru data doğru yerde olacak, yapıcı fonksiyonlara parametre aktarımları doğru olacak. → 2p
get ve set fonksiyonları → 2p

```
class point1d {
    public:
        point1d();
        point1d(double xIn);
        virtual ~point1d();
        virtual double
calculateLength();
    protected:
        double x;
};
point1d::point1d(double
xIn=0){
   x = xIn;
}
double
point1d::calculateLength(){
     return sqrt(x*x);
}
point1d::~point1d() { //dtor }

// getX, setX
```

```
class point2d:public point1d {
    public:
        point2d();
        point2d(double xIn,
double yIn);
        virtual ~point2d();
        virtual double
calculateLength();
    protected:
        double y;
};
point2d::point2d(double xIn,
double yIn=0):point1d(xIn){
    y = yIn;
}
double
point2d::calculateLength(){
   return sqrt(x*x + y*y);
}
point2d::~point2d() { //dtor }

// getX, getY, setX, setY
```

```
class point3d:public point2d{
    public:
        point3d();
        point3d(double xIn,
double yIn, double zIn);
        virtual ~point3d();
        virtual double
calculateLength();
    protected:
        double z;
};
point3d::point3d(double xIn,
double yIn, double
zIn=0):point2d(xIn, yIn){
    z=zIn;
}
double
point3d::calculateLength(){
   return(sqrt(x*x + y*y +
z*z));
}
point3d::~point3d() { //dtor }
// get ve set fonksiyonları
```

b
fonksiyonun içi doğru yazılacak, ve doğru çalışmalı → 15p

```
double calculateTotalLength(point1d *pointsVector[], int numOfPoints){
   double totalLength;
   for(int i = 0; i < numOfPoints; i ++)
      totalLength += pointsVector[i]->calculateLength();
   return totalLength;
}
```

3) Sizden **dinamik** olarak boyutu belirlenebilecek bir matris sınıfı gerçeklemeniz istenmektedir. Bu sınıfa ait nesneler aşağıdaki işlemlere tabi tutulabilmelidirler. (***0.5*PÇ2+0.5*PÇ3***)
   a) Parametreli ve parametresiz constructorlar ve gerekli get ve set fonksiyonları gerçeklenmelidir. (5p)
   b) Matris sınıfına ait parametreli constructor **rowCount, colCount ve isUnit** isimli üç parametre almalı. **isUnit** parametresi default olarak sıfır olmalı, eğer setlenmişse birim matris oluşturacak önlemler yapıcıda alınmalıdır (5p).
   c) Matris nesnesinin birim matris olduğunu test eden bir fonksiyonu gerçeklemeniz gerekmektedir. `if(a.isUnit()) cout << "Unit matrix" << endl;` (5p)
   d) Matris toplama: → `a=b+c;` (5p)
   e) Matris çarpma: → `a=b*c;` (10p)
   f) Matrisi bir skalar ile çarpma: → `a=b*5;` (5p)
   g) Bir satırı döndürme: → `double *row=a[0]; //1. satırı geri döndür` (5p)
   h) Bir değer geri döndürme: → `double value = a[0][0];` (5p)

Cevap 3

a →
nesne yönelimli programlama ilkelerine (data abstraction) uyulmamışsa puan verilmez.
matris tanımlama (2p) + fonksiyonların gerçeklenmesi (3p)

```
class Matrix {
    public:
        Matrix();
        Matrix(int rowsIn, int
columnsIn, int isUnit=0);
        double *operator[](int index);
        Matrix operator*(const Matrix&
matrixIn);
        Matrix operator*(const double
scalarIn);
        Matrix operator+(const Matrix&
matrixIn);
        int getRows();
        int getColumns();
        bool isUnit();
        virtual ~Matrix();
    private:
        double **matrix;
        int rows;
        int columns;
};
```

```
Matrix::Matrix() {
   //ctor
   rows = 10;
   columns = 10;
   matrix = new double *[rows];
   for(int i = 0; i < rows; i++)
      matrix[i] = new double[columns];
}

int Matrix::getRows(){
   return rows;
}
int Matrix::getColumns(){
   return columns;
}
// ayrıca set fonksiyonları da yazılmalıdır
```

b → dinamik matris oluşturma (3p) + birim matris (2p)

```
Matrix::Matrix(int rowsIn, int columnsIn, int isUnit){
   if(isUnit)
     rows = columns = rowsIn;
   else
     rows = rowsIn;
   columns = columnsIn;
   matrix = new double *[rows];
   for(int i = 0; i < rows; i++)
      matrix[i] = new double[columns];
   if(isUnit){
     for(int i = 0; i < rows; i++)
       matrix[i][i] = 1;
   }
}
```

c → kodun doğru olması beklenir → 5p

```
bool Matrix::isUnit(){
   for(int i=0; i<rows; i++)
      if(matrix[i][i] != 1)
        return false;
   return true;
}
```

d → operator+ fonksiyonunun prototipi, parametreleri (2p) + kodun doğruluğu (3p)

```
Matrix Matrix::operator+(const Matrix& matrixIn){

    Matrix outMatrix(this->rows, this->columns);
    if(this->columns == matrixIn.columns && this->rows == matrixIn.rows){
       int i,j;
       for(i=0; i<this->columns; i++ ){
         for(j=0; j<this->rows; j++){
            outMatrix[i][j] = matrix[i][j] + matrixIn.matrix[i][j];
         }
       }
    }
    return outMatrix;
}
```

e → operator* fonksiyonunun prototipi, parametreleri (3p) + kodun doğruluğu (7p)

```
Matrix Matrix::operator*(const Matrix& matrixIn){
    Matrix outMatrix(this->rows, matrixIn.columns);
    if(this->columns == matrixIn.rows){
```

```
       int i,j,k;
       for(i=0; i<this->columns; i++ ){
         for(j=0; j<matrixIn.rows; j++){
            int subTotal=0;
            for(k=0; k<matrixIn.rows; k++)
                subTotal += this->matrix[i][k]*matrixIn.matrix[k][j];
            outMatrix[i][j] = subTotal;
         }
       }
    }
    return outMatrix;
}
```

f → operator* fonksiyonunun prototipi, parametreleri (2p) + kodun doğruluğu (3p)

```
Matrix Matrix::operator*(const double scalarIn){
   Matrix outMatrix(this->rows, this->columns);
   int i,j;
   for(i=0; i<this->columns; i++ ){
     for(j=0; j<this->rows; j++){
        outMatrix[i][j] = matrix[i][j] * scalarIn;
     }
   }
   return outMatrix;
}
```

g → operator[] fonksiyonunun prototipi, parametreleri (3p) + kodun doğruluğu (7p)

```
double *Matrix::operator[](int index){
   if(index < rows)
     return matrix[index];
   else
     return matrix[0];
}
```

h → g maddesi yapılınca bu otomatik yapılıyor zaten

```
#include <iostream>
namespace space1{
 class Year{
   public:
     Year(int year=1900){
       this->year = year;
     }
     void setYear(int year){
       this->year = year;
     }
     int getYear(){
       return year;
     }
     bool isLeap(){
       // bu fonksiyonu gerçekleyiniz
       ..................
     }
     int getCentury(){
       // bu fonksiyonu gerçekleyiniz
       ..............................
     }
   private:
     int year;
 };
}
```

```
namespace space2{
 struct Year{
   int year;
 };
 int getCentury(Year y){
   // boş kısımları beklenen çıktıyı
verecek şekilde doldurunuz.
   return (y.year%100!=0)? ............
: ............ ;
 }
}
int main(){
 space1::Year y(1900);
 space2::Year y1;
 y1.year = 2001;
 std::cout<< ........................
<<std::endl; // line 1
 std::cout<< ........................
<<std::endl; // line 2
 return 0;
}
```

Sınav Kuralları

1. Bölüm sayfasında ilan edilen sınav kurallarına uymak zorunludur.
2. Yalnızca 1 adet cevap kağıdı kullanmak yeterlidir.
3. Sorular 2 ve 3 numaralı program çıktıları ile ile ilişkilidir

BAŞARILAR