SISTEM ERROR CHECKING

Sistem error checking yang digunakan di dalam program kami ada beberapa.

Berikut salah satu sample code yang kami buat untuk melakukan error checking :

```
do
{
    printMenu();
    Console.Write("Masukan pilihan anda : ");
    pilihan = Console.ReadLine();
    kondisi = int.TryParse(pilihan, out pilih);
    if (kondisi == true && pilih > 0 && pilih < 5)
    {
        continue;
    }
    Console.WriteLine("\nPilihan yang anda masukan salah!");
    Console.WriteLine("Tekan sembarang untuk memilih kembali...");
    Console.ReadLine();
} while (pilih < 1 || pilih > 4);
```

Error checking yang dilakukan diatas ialah melakukan prevensi terhadap karakter selain angka. Jadi di baris code ini hanya bisa menerima angka. Dibuatnya variable Boolean yang dinamakan kondisi digunakan untuk hasil parsing dari input yang dimasukan. Kami menggunakan sistem tryparse dimana ketika file input yang awalnya kami deklarasikan sebagai string akan dilakukan parsing menuju integer. Tryparse ini juga menghindarkan program untuk crash atau terjadinya error sehingga program tidak dapat berjalan dengan baik. Hingga versi debug paling akhir, sistem code seperti ini belum kami temukan bug atau kejanggalan yang menghambat program ini berjalan dengan baik.

Tak hanya menggunakan TryParse, program kali juga melengkapinya lagi dengan menggunakan switch case.

```
switch (pilih)
{
    case 1:
        // tampilin shape
        Shape.tampilshape();
        break;
    case 2:
        // tambah shape
        Shape.tambahshape();
        break;
    case 3:
        //delete shape
        Shape.hapusshape();
        break;
    case 4:
```

Created by : Alvin Julian, David Antoni, Kelvin Kristianto

```
            //exit
            Environment.Exit(0);
            break;
        default:
            break;
    }
```

Code mengenai switch case diatas melakukan prevensi untuk kedua kalinya. Hal ini mencegah apabila ada kesalahan parsing yang dilakukan program kali, maka program ini akan melakukan prevensi untuk kedua kalinya.

Untuk melakukan file append atau penambahan baik itu berupa karakter atau angka ke dalam file, program kami menggunakan sistem try dan catch.

```
    try
    {
        using (FileStream fs = new FileStream(file, FileMode.Append,
           FileAccess.Write))
        using (StreamWriter sw = new StreamWriter(fs))
        {
            sw.WriteLine(size);
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
    }
```

Proses try catch yang ada diatas merupakan proses untuk melakukan append ke dalam file. Untuk melakukan append ke dalam file, tentu file tersebut harus dibuka terlebih dahulu. Setelah file di buka, barulah file tersebut dimasukan sejumlah data yang diinginkan. Dalam proses membuka dan menuliskan ke dalam file, biasanya ada error. Baik itu filenya tidak ada maupun filenya tidak bisa diakses. Untuk itu, program kami tambahkan barisan try catch untuk mencegah adanya error yang tidak diduga sebelumnya.

Proses try catch kami tambahkan bukan hanya untuk melakukan append kepada file, melainkan juga untuk menghapus dan beberapa perintah lainnya.

KONSEP OBJECT-ORIENTED PROGRAMMING

Polymorphism

Pada code dibawah ini, akan menunjukan sifat OOP yang dipakai yakni polymorphism

```
public static double rumus(double ja,string hit)
{
    if (hit == "Luas")
        return ja * ja * 3.14;
    else
        return 2*ja * 3.14;
}
public static int rumus(int ja, string hit)
{
    if (hit == "Luas")
        return ja * ja;
    else
        return 4 * ja;
}

public static int rumus(int panjang,int lebar, string hit)
{
    if (hit == "Luas")
        return panjang *lebar;
    else
        return ((2*panjang)+(2*lebar));
}
```

Mengapa kami membuat bagian ini menjadi polymorphism? Hal ini dikarenakan ketiganya memiliki kesamaan yakni melakukan kalkulasi pada luas dan keliling. Misalkan saja bagian yang paling pertama. Code yang paling atas itu digunakan ketika ingin dilakukan kalkulasi pada luas dan keliling lingkaran. Baian kedua digunakan untuk kalkulasi luas dan keliling persegi. Sedangkan bagian paling akhir digunakan untuk kalkulasi luas dan keliling persegi panjang. Disini juga terdapat function overloading dimana yang digunakan untuk memisahkan penghitungan pada object tertentu.

Inheritance

Pada code dibawah ini, akan menunjukan sifat OOP yang dipakai yakni inheritance,

```
public class Shape
    {
       ....
       ....
    }
```

```
public class Square : Shape
{

.....

....

}

public class Circle : Shape
{

.....

....

}

public class Rectangle : Shape
{

.....

....

}
```

Dari baris program ini dapat dilihat bahwa kelas Square, Circle, dan Rectangle merupakan turunan dari kelas Shape. Sehingga ke tiga kelas tersebut dapat mengakses method dan variable di dalam kelas Shape yang berupa Public atau Protected.

Dalam kelas Square, Circle, dan Rectangle mereka dapat mengakses method yang ada didalam Shape untuk digunakan dalam menghapus, menambah data dari file, seperti berikut ini

```
        protected static void tambahkefile(int size,string filein)
        {
            string sizeS = size.ToString();
            string dir = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
            string file = dir + @"\"+filein+".txt";
            //StreamReader sr = new StreamReader(file);
            try
            {
                using (FileStream fs = new FileStream(file, FileMode.Append,
FileAccess.Write))
                using (StreamWriter sw = new StreamWriter(fs))
                {
                    sw.WriteLine(size);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
        }
```

```
        protected static void tambahkefile(int panjang,int lebar)
        {
            string panjangS = panjang.ToString();
            string lebarS = lebar.ToString();
            string dir = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
            string file = dir + @"\rectangle.txt";
            //StreamReader sr = new StreamReader(file);
            try
            {
                using (FileStream fs = new FileStream(file, FileMode.Append,
FileAccess.Write))
                using (StreamWriter sw = new StreamWriter(fs))
                {
                    sw.WriteLine(panjang+"\t"+lebar);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
        }
```

Atau untuk menghapus data, setiap anak kelas dapat mengakses code dari Shape yaitu,

```
        protected static void hapuskefile(int size, string filein)
        {
            string sizes = size.ToString();
            int counter = 0;
            string line;
            string pattern = @"\t+";
            Regex rgx = new Regex(pattern);
            string dir = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
            string file = dir + @"\"+filein+".txt";
            string filecp = dir + @"\" + filein + "cp.txt";
            StreamReader sr = new StreamReader(file);
            while ((line = sr.ReadLine()) != null)
            {
                string[] result = rgx.Split(line);
                //mengecek apakah nama, kalau ada gak melakukan apa2 kalau ada nulis file
dengan mengabaikan nama
                if (result[0] != sizes)
                {
                    if (!File.Exists(filecp))
                    {
                        // Create a file to write to. kalau belom ada filenya
                        try
                        {
                            using (StreamWriter swnew = File.CreateText(filecp))
                            {
                                swnew.WriteLine(result[0]);
                            }
                        }
                        catch (Exception e)
                        {
                            Console.WriteLine(e);
                        }

                    }
```

```
                    //kalau ud ada file yang mau ditulis
                    else
                    {


                        try
                        {
                            using (FileStream fs = new FileStream(filecp,
FileMode.Append, FileAccess.Write))
                            using (StreamWriter sw = new StreamWriter(fs))
                            {
                                sw.WriteLine(result[0]);
                            }
                        }
                        catch (Exception e)
                        {
                            Console.WriteLine(e);
                        }
                    }
                }
                else
                {
                    counter++;
                }
            }
            sr.Close();
            string nilai;
            if (filein == "circle")
            {
                nilai = "jari-jari";
            }
            else
            {
                nilai = "sisi";
            }
            if (counter > 0)
            {
                Console.WriteLine("Penghapusan " + filein + " dengan {0} {1} Sejumlah {2}
Berhasil!\n",nilai,sizes,counter);
            }
            else
            {
                Console.WriteLine(filein + " dengan {0} {1} Tidak ada!\n", nilai,sizes);
            }
        }
```

Untuk menghapus dan menambah file, kami menggunakan tipe protected. Sehingga method tersebut hanya dapat diakses oleh Shape sendiri atau oleh turunananya saja.

Dalam melihat luas dan keliling setiap shape (Square, Circle, dan Rectangle) hanya dapat diakses oleh masing-masing class seperti berikut

```
public class Square : Shape
{
        …
        …
        …
        public static void sorting(string hit)
         {
             Console.Clear();
             Console.WriteLine("\t\t\t\t\t\t\t\tLihat Square");
             Console.WriteLine("\t\t\t\t\t\t\t\t==================");
             string dir = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
             string file = dir + @"\square.txt";
             string[] scores = File.ReadAllLines(file);
             var orderedScores = scores.OrderBy(x => int.Parse(x.Split('\t')[0]));
             int counter = 0;
             Console.WriteLine("No.\t" + hit + "\t\tSisi");
             foreach (var score in orderedScores)
             {
                 counter++;
                 /// kode buat nampilin file dan jumlah gitu...
                 int jari = Convert.ToInt16(score);
                 int hasil = rumus(jari, hit);
                 Console.WriteLine(counter + ".\t" + hasil + "\t\t\t" + jari);
             }
             Console.WriteLine("\nTekan sembarang untuk kembali ke menu lihat square");
             Console.ReadKey();
             lihat();
         }

}
```

Masing-masing kelas bentuk ini hanya dapat mengakses kedalam file yang mereka punya saja, sehingga tidak menganggu file yang lain.

Untuk melihat semua bentuk, kami memberikan kuasa tersebut pada kelas Shape, yang dapat mengakses kedalam semua file sehingga dapat menghasilkan tampilan lihat semua shape.

```
public class Shape
    {
        ....
        ....
        public static void sorting(string hit)
         {

             string dir = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
             string file = dir + @"\rectangle.txt";
             string filecp = dir + @"\lihat.txt";
             bacafile(file, filecp, hit);
             file = dir + @"\circle.txt";

             bacafile1(file, filecp, hit,"Circle");
             file = dir + @"\square.txt";

             bacafile1(file, filecp, hit,"Square");
             Console.Clear();
             Console.WriteLine("\t\t\t\t\t\t\t\tLihat Semua Shape");
             Console.WriteLine("\t\t\t\t\t\t\t\t==================");
```

```
            file = dir + @"\lihat.txt";
            string[] scores = File.ReadAllLines(file);
            var orderedScores = scores.OrderBy(x => double.Parse(x.Split('\t')[0]));

            int counter = 0;
            if (hit == "Luas")
                Console.WriteLine("No.\t" + hit + "\t\tPanjang/Sisi/Jari-
Jari\t\tLebar\tShape");
            else
                Console.WriteLine("No.\t" + hit + "\tPanjang/Sisi/Jari-
Jari\t\tLebar\tShape");

            foreach (var score in orderedScores)
            {
                counter++;
                Console.WriteLine(counter+".\t" + score);
            }
            File.Delete(file);
            Console.WriteLine("\nTekan sembarang untuk kembali ke menu lihat semua
shape");
            Console.ReadKey();
            lihat();
        }

    }
```

Dari kode diatas dapat dilihat bahwa kelas Shape yang memegang control untuk dapat mengakses ke seluruh file yang ada.