CASE PROGRAM – ENUM

import case.lang.System

namespace HelloWorld {

//String->Object->Main is a type constructor

String->Object->Main

#public class Program

//during compilation this function is included in the

//codebase whether it is used or not

@required [public Program(String [] args)

System.out.println(@texts:“Hello World”)

Print sizeOf(“Size of args structure” + sizeof(args))

]

}

String->Object🡪Digit

#public class <Digit>

public Digit(double d) { val = d; }

public double val;

// ...other members

// User-defined conversion from Digit to double

[public static implicit operator double(Digit d)

return d.val;

]

// User-defined conversion from double to Digit

[public static implicit operator Digit(double d)

return new Digit(d);

]

#end class

String->Object->Program

#public class <Program>

[static void Main(string[] args)

Digit dig = new Digit(7);

//This call invokes the implicit "double" operator

double num = dig;

//This call invokes the implicit "Digit" operator

Digit dig2 = 12;

Console.WriteLine("num = {0} dig2 = {1}", num, dig2.val);

Console.ReadLine();

]

#end class

By eliminating unnecessary casts, implicit conversions can improve source code readability. However, because implicit conversions do not require programmers to explicitly cast from one type to the other, c

are must be taken to prevent unexpected results. In general, implicit conversion operators should never throw exceptions and never lose information so that they can be used safely without the programmer's awareness. If a conversion operator cannot meet those criteria, it should be marked **explicit**. For more information, see [Using Conversion Operators](https://msdn.microsoft.com/en-us/library/85w54y0a.aspx)..