Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Informatica 2
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Hoja de trabajo #1

Fecha de entrega: 25 de Enero, 2018 - 11:59pm

Instrucciones: Realizar cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Hoja de trabajo 1". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Los ejercicios de programación deben ser colocados en una carpeta llamada "Programas", la cual debe colocarse dentro de la carpeta correspondiente a esta hoja de trabajo.

Ejercicio #1 (10%)

1. Descargar e instalar el framework *.Net Core* desde: `https://www.microsoft.com/net/learn/get-started/windows`. También se recomienda utilizar Visual Studio Code con la extension de C# (`https://code.visualstudio.com/Docs/languages/csharp`), pero el estudiante tiene la libertad de utilizar cualquier editor o ide para programar.

2. Dentro de la carpeta "Programas" de este trabajo, crear una nueva carpeta llamada "QueHaceres". Abrir una terminal en esta carpeta y crear un nuevo proyecto de consola mediante .Net Core:
   ```
   > dotnet new console
   ```

Ejercicio #2: Crear Proyecto (10%)

Dentro de la carpeta del proyecto, definir las siguientes clases en un archivo separado por clase. El archivo debe tener el mismo nombre que la clase:

- QueHacer
- QueHaceres
- Persona

Por el momento no es necesario definir ningun atributo o operación en escritas clases. Se llevara a cabo en las siguientes secciones.

Ejercicio #3: Definir clases (40%)

Propiedad enumerada "Estado"

Defina una propiedad enumerada (un `enum`) llamado "Estado" y coloquelo en el archivo correspondiente a la clase "QueHacer". Esta propiedad debe tener dos campos: "EnProgreso" y "Terminado"/

Clase "QueHacer"

La clase "QueHacer" debe definir al menos 3 propiedades y una operación que haya definido en la hoja anterior. Adicionalmente, debe tener una propiedad llamada "Estado" de tipo "Estado" que puede ser publicamente leida pero solo modificada privadametne. Adicionalmente, debe tener una operación llamada "Completar", que le asigna a la propiedad "Estado" el valor "Terminado". Esta propiedad debe tener el valor "EnProgreso" cuando el "QueHacer" es creado.

El metodo constructor del objeto "QueHacer" debe recibir los valores necesarios para inicializar todas las propiedades de una instancia de "QueHacer".

Clase "Persona"

La clase persona debe definir las siguientes propiedades:

- "Nombre" de tipo `string`
- "Apellido" de tipo `string`
- "Tareas" de tipo `QueHaceres`

Adicionalmente debe definir las siguientes operaciones:

- "EstaDisponible", que no toma ningun argumento y retorna `true` si el numero de "QueHacer" con estado "EnProgreso" en la lista "QueHaceres" de la persona es menor a 1. De lo contrario, false.
- "AgregarQuehacer", que toma como parametro un "QueHacer" y lo agrega a su lista de "QueHaceres".
- "CompletarQuehacer", que no toma ningún parametro y coloca el "QueHacer" más antigüa (la cual se le haya asignado lo más antes mediante su metodo "AgregarQuehacer"), con estado "EnProgreso", en estado "Terminado", mediante su metodo "Finalizar".

Clase "QueHaceres"

Definir la clase "QueHaceres" de tal manera que le permita a la clase "Persona" manejar sus "QueHacer" mediante los metodos descritos anteriormente. **Importante:** la clase persona **no** debe tener variables de tipo "QueHacer". Toda administración de los objetos "QueHacer" debe ser realizada por la clase "QueHaceres".

Ejercicio #4: Presentar resultados (20%)

El comando `dotnet new console` crea automaticametne un archivo llamado "Program.cs". En este archivo existe un metodo llamado "main". Este es el metodo que ejecutara el comando `dotnet run`, el punto de entrada al programa. Este metodo debe realizar lo siguiente:

1. Crear dos instancias de la clase "Persona", me referire a ellos como *Persona1* y *Persona2*
2. Crear seis instancias de la clase "QueHacer"

3. Agregar 3 “QueHacer” a cada instancia de “Persona” mediante su metodo “AgregarQuehacer” no se debe agregar el mismo “QueHacer” a dos personas diferentes.

4. Llamar el metodo “CompletarQuehacer” una vez con “Persona1” y tres veces con “Persona2”.

5. Imprimir en la consola mediante el metodo `Console.WriteLine` el resultado del metodo “EstaDisponible” de ambas peersonas. **Nota:** “Persona1” no debe estar disponible mientras que “Persona2” si, luego de la ejecución del programa.

Ejercicio #5: Confusion (20%)

Listing 1: Confusion

```
class Program{
    public static void main(string[] args){
        Persona turing = new Persona("Alan", "Turing");
        Persona alonzo = new Persona("Alonzo". "Church");

        QueHacer deber1 = new QueHacer("Inventar las ciencias de la computacion.");

        turing.AgregarQuehacer(deber1);
        alonzo.AgregarQuehacer(deber1);

        turing.CompletarQuehacer();

        Console.WriteLine(" Estara disponible Alonzo?");
        Console.WriteLine(alonzo.EstaDisponible() ? "Si" : "No");
    }
}
```

Luego de ejecutar el programa que se muestra en esta sección, el objeto “alonzo” descubre que se encuentra disponible. Sin embargo, el metodo “CompletarQuehacer” nunca fue ejecutado en dicho objeto. A que se debe este extraño resultado? Por colocar su respuesta en un documento Latex y entregarlo como parte de la tarea.

References