UNViMe

Trabajo Práctico 0

Alumno: Tula E. Fernando

Carrera: Ing. de Sistemas de Información

El presente documento es un informe de trabajo que si bien no fue requerido en la consigna, me sirve a mi para cerrar el tema, y le puede servir a alguno de los evaluadores para poder ver qué es lo que he realizado a lo largo del trabajo:
Primero que nada clasificare las funciones que realice y las clasificare en acciones y calculos, teniendo en cuenta que los cálculos o funciones puras son aquellos que cumplen con:

- Transparencia referencial.
- No generan efectos secundarios o cambios en el entorno.
- No mutan los objetos.
- Están aisladas.

Acciones son todas aquellas funciones que por contrapartida:

- Generan efectos secundarios.
- Mutan los objetos.
- Modifican el estado global del sistema: generan efectos secundarios.

En total mi programa cuenta con 36 funciones de las cuales:

- 29 son puras: 75%
- 10 son impuras: 25%

De modo que no llegue al objetivo del 80% de funciones puras por cuestiones de tiempo y que debo estudiar para otras materias. A continuación detallo porque las funciones que use son puras, que hacen algunas de ellas, y en comparación cuales son impuras y porque:

Funciones Puras de mi programa:

- insertarTarea

```
constructorListaTareas.prototype.insertarTarea = function(lista : interfazTarea[],
tarea: interfazTarea) : interfazTarea[] {
  return [...lista, tarea];
}
```

Es función pura porque:recibe por parámetro la lista de tareas y la tarea a insertar y devuelve una nueva lista que copia con "..." todos los objetos de la lista anterior y le agrega el nuevo objeto a agregar.

- crearTarea

```
constructorListaTareas.prototype.crearTarea = function (id : string, titulo : string,
estado: Estado, descripcion : string, dificultad : Dificultad, vencimiento:
Vencimiento, fechaCreacion: Date, ultimaModificacion: Date ): interfazTarea {
    const tarea : interfazTarea = new (constructorTarea as any) ( id, titulo,
descripcion, dificultad, vencimiento, fechaCreacion, ultimaModificacion, estado);
    Object.freeze(tarea);
    return tarea;
}
```

Esta función pura obtiene los datos por parametro, cumple con la transparencia referencial, ya que siempre devuelve el mismo objeto cuando se le pasan los mismos parametros y para sumar congela el objeto para evitar futuros cambios, asegurando la inmutabildiad del mismo. No genera efectos secundarios.

- filtrarTareas

```
constructorListaTareas.prototype.filtrarTareas = function (clave : string) :
interfazTarea[] {
  const listaFiltrada = this.listaTareas.filter(t =>
    t.titulo.toLowerCase().includes(clave) ||
t.descripcion.toLowerCase().includes(clave)
  )
  return listaFiltrada;
}
```

Es funcion pura porque recibe por parametro la lista y devuelve una copia de la lista con los objetos que cumplen con la condicion de la clave, de modo que cumple con la inmutabilidad, no genera efectos secundarios.

Funciones Impuras de mi programa:

- AgregarTarea

```
constructorListaTareas.prototype.agregarTarea = function () : void {
  const id = uuidv4();
  const titulo = this.agregarTitulo();
  const descripcion = this.agregarDescripcion();
  const estado: Estado = "pendiente"; // Estado inicial siempre es "pendiente"
  const dificultad: Dificultad= this.agregarDificultad();
  const vencimiento : Vencimiento= this.agregarVencimiento();
  const ultimaModificacion = new Date();
  const fechaCreacion = new Date();
  const tarea : interfazTarea = this.crearTarea
```

```
  (id, titulo, estado, descripcion, dificultad, vencimiento, fechaCreacion,
ultimaModificacion);
  this.listaTareas = this.insertarTarea(this.listaTareas, tarea);
}
```

Esta función impura gestiona acciones que interactúan con el entorno como obtención de id mediante uuid y el tiempo mediante date y generan efectos secundarios y funciones puras. No cumple con la integridad referencial porque ante las mismas entradas no devuelve siempre el mismo resultado. En este caso no tiene ninguna entrada, sino que ejecuta un procedimiento y no devuelve ningún calculo u objeto, de modo que no es función pura. Cuando hago la reasignación de la lista de tareas al invocar mi insertarTarea no rompo la pureza, porque hago apuntar la variable a un nuevo valor (lista = ordenar(lista)): no cambia el viejo arreglo; crea otro y reemplaza la referencia.
Devolver copias y luego reasignar el estado "actual" no viola inmutabilidad de los datos. No modificás el arreglo viejo. Solo haces una transición de estado en la capa de orquestación

- agregarTitulo

```
constructorListaTareas.prototype.agregarTitulo = function (): string {
  let titulo : string = prompt("Título: ") ?? "";
  if (titulo.length < 100 && titulo.length > 0) {
    return titulo;
  } else {
    console.log("El título debe tener entre 1 y 100 caracteres.");
    return this.agregarTitulo();
  }
}
```

Es una función impura porque hace uso de console.log de modo que genera efectos secundarios. Es importante notar que uso recursividad para no entrar en bucles for o while

- encontrarPosicion

```
constructorListaTareas.prototype.encontrarPosicion = function (idTarea: string):
number {
  return this.listaTareas.findIndex(t => t.getId() === idTarea);
};
```

Es función pura porque no hace iteraciones, para la misma entrada devuelve siempre el mismo resultado, no genera efectos secundarios, etc.

Conclusiones y/o inconvenientes:

Ya que es inevitable en un programa de este tipo evitar las acciones, o funciones impuras, me costo hacer un balance entre: cumplimiento de la consigna y mantener buenas practicas de programación, ya que podría haber acortado aún más las funciones impuras al generar un bloque de código dentro del menú principal que hace acciones e invoca los cálculos para la

creacion/modificacion de objetos pero que dificulta la legibilidad y el mantenimiento del programa.