Trabajo Práctico 2
Programación Orientada a Objetos

Paradigmas de Lenguajes de Programación — 1er cuat. 2020

Fecha de entrega: 25 de junio de 2020

1. Introducción

La reina Elsa[1] está aprendiendo a usar sus poderes para mejorar la vida de la gente de su reino. La gente tiene muchas necesidades y Elsa está necesitando cada vez más amigos para poder encargarse de todo. Por suerte Olaf[2] le recordó que tiene el poder de crear muñecos de nieve vivientes que podrían ayudarla. El problema es que Elsa aún no domina esta habilidad así que necesitará la ayuda de Olaf para hacerlo. En lugar de fabricar muñecos de nieve de la nada, Elsa los modelará a partir de uno anterior. Usando a Olaf como base, Elsa puede crear a un nuevo muñeco a su imagen y semejanza y luego darle instrucciones específicas respecto a la tarea que el nuevo muñeco tiene asignada.

Representación utilizada

Los muñecos de nieve son objetos que saben responder los siguientes mensajes:

1. ***nombre***: que retorna el texto correspondiente al nombre del muñeco
2. ***altura***: que retorna la altura en metros del muñeco
3. ***directiva***: que retorna el texto correspondiente a la misión que tiene asignada el muñeco

2. Ejercicios

Ejercicio 1

Definir el objeto `Olaf`, que representa al muñeco de nieve Olaf. Olaf mide 1,62 metros y su directiva es *abrazar*.

Ejercicio 2

(a) Definir la función `nuevoMuneco` que toma un nombre, una altura y una directiva y retorna un nuevo muñeco basado en Olaf pero con sus respectivos atributos actualizados.

(b) Utilizando la función `nuevoMuneco` crear a Malvavisco. Malvavisco mide 5 metros y su directiva es *cuidar a la reina*.

[1] https://disney.fandom.com/es/wiki/Elsa
[2] https://disney.fandom.com/es/wiki/Olaf

Ejercicio 3

(a) Agregarle a Olaf el método `presentarse` que debe devolver el texto "*Hola, soy Olaf y me encanta abrazar*". Asegurarse que todos los muñecos basados en Olaf puedan responder este mensaje correctamente (es decir, con su propio nombre y directiva).

(b) Agregarle a Olaf el método `abrazar` que no toma argumentos, incrementa en uno la altura de Olaf y retorna el texto "*abrazar*".

Ejercicio 4

Así como Olaf sabe ejecutar el método `abrazar` (porque su directiva es *abrazar*), cada muñeco debería tener definida también una función particular relativa a su directiva.

(a) Definir una **función constructora** `Muneco` que además de inicializar un nuevo muñeco (con los mismos argumentos que `nuevoMuneco`) tome un argumento adicional que será una función (relacionada a su directiva) que el nuevo muñeco deberá ejecutar cuando reciba el mensaje correspondiente a su directiva.

(b) Responder (**justificando adecuadamente**): ¿qué objetos pueden responder al mensaje `abrazar`?

- Olaf
- Malvavisco
- Un objeto creado con `nuevoMuneco` antes de agregarle a Olaf el método `abrazar`.
- Un objeto creado con `Muneco` antes de agregarle a Olaf el método `abrazar`.
- Un objeto creado con `nuevoMuneco` después de agregarle a Olaf el método `abrazar`.
- Un objeto creado con `Muneco` después de agregarle a Olaf el método `abrazar`.

Nota: los modificadores "antes" y "después" se refieren al momento en el que el objeto en cuestión fue creado, no al momento en el que a Olaf se le agregó el método.

Ejercicio 5

Uno de los principales problemas del reino es que el correo siempre llega tarde. Crear al muñeco mensajero `Liam` que se encarga de entregar mensajes (en otras palabras, de *mensajear*). La función asociada a su directiva debe recibir como argumentos un remitente, un destinatario y el mensaje que el remitente le está mandando al destinatario. El sistema de mensajería funciona de la siguiente forma:

1. Si el destinatario sabe responder al mensaje del remitente, el mensajero debe entregar el mensaje al destinatario. En caso contrario, la función retorna el mensaje original.

2. El resultado de haber entregado el mensaje al destinatario también es un mensaje que el mensajero debe entregar como respuesta del destinatario al remitente.

3. Si el remitente sabe responder a la respuesta del destinatario, el mensajero debe entregar la respuesta al remitente. En tal caso, la función debe retornar el resultado de tal ejecución. En caso contrario, la función retorna la respuesta del destinatario.

Ejercicio 6

Elsa se dio cuenta que algunos trabajos requieren más esfuerzo que otros. Los muñecos que tienen las tareas más difíciles necesitan ayuda y los que tienen las tareas más fáciles terminan pronto y no tienen nada que hacer. Para que los muñecos ociosos puedan ayudar a los demás, se necesita que puedan modificar su directiva.

(a) Hacer lo necesario para que todos los muñecos (actuales y futuros) puedan responder al mensaje `cambiarDirectiva` que toma como argumento la nueva directiva a ser asignada y su función asociada. La directiva pasada como argumento debe ser diferente a la que ya tenía. En caso contrario, el muñeco se confunde y su nueva directiva pasa a ser ".".

(b) Hacer lo necesario para que todos los muñecos (actuales y futuros) puedan responder al mensaje `solicitarAyuda` que toma como argumento otro muñeco y se lo asigna como ayudante. Al ayudante se le debe asignar la misma directiva que la del muñeco que lo solicitó (y debe poder ejecutar su misma función asociada). Un muñeco que ya tiene ayudante no puede solicitar un segundo ayudante. Si esto pasara, el muñeco que solicitó ayuda debe indicarle a su ayudante que sea él quien solicite otro ayudante. Por ejemplo:

1. B solicita ayuda de A.
2. A pasa a ser ayudante de B.
3. B solicita ayuda a C pero como B ya tiene un ayudante, le pasa el pedido a A y C pasa a ser ayudante de A.

3. Pautas de Entrega

Se debe entregar el código impreso con la implementación de los ejercicios. Cada función o método asociado a los ejercicios debe contar con ejemplos que muestren que exhibe la funcionalidad solicitada. Además, se debe enviar un e-mail conteniendo el código fuente en JavaScript a la dirección plp-docentes@dc.uba.ar. Dicho mail debe cumplir con el siguiente formato:

- El título debe ser `[PLP;TP-POO]` seguido inmediatamente del nombre del grupo.
- El código JavaScript debe acompañar el e-mail y lo debe hacer en forma de archivo adjunto con nombre `taller.js` (puede adjuntarse un **.zip** o un **.tar.gz**).

El código debe poder ejecutarse en los navegadores de los laboratorios. No es necesario entregar un informe sobre el trabajo, alcanza con que el código esté **adecuadamente** comentado (son comentarios adecuados los que ayudan a entender lo que no es evidente o explican decisiones tomadas; no son adecuadas las traducciones al castellano del código). Los objetivos a evaluar son:

- Corrección.
- Declaratividad.
- Prolijidad: evitar repetir código innecesariamente y usar adecuadamente los métodos previamente definidos.

Importante: se admitirá un único envío, sin excepción alguna. Por favor planifiquen el trabajo para llegar a tiempo con la entrega.

Referencias del lenguaje JavaScript

Como principales referencias del lenguaje de programación JavaScript, mencionaremos:

- **W3Schools JavaScript Reference:** disponible online en: https://www.w3schools.com/jsref/default.asp.
- **MDN Web Docs: Mozilla - Referencia de JavaScript**: disponible online en: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia.