Trabajo Práctico 3
Programación Orientada a Objetos

Paradigmas de Lenguajes de Programación — 2do cuat. 2014

Fecha de entrega: 20/11/2014

Introducción

En este trabajo implementaremos el conocido juego cordobés **Piedra, Papel y Tijera**. Para lograrlo, el proceso será guiado a través de una serie de tests inspirados en la técnica *Test Driven Development*[1]. La manera de resolver el trabajo será entonces implementar el mínimo código razonable para lograr ir pasando los tests de manera progresiva, manteniendo los tests anteriores pasando. Cada ejercicio irá acompañado de un test que tenga la información necesaria para resolverlo y algunas sugerencias que pueden utilizar.

Ejercicios

Ejercicio 1

Implementar lo necesario para que el `test01` pase correctamente. Este test asegura que dos elecciones posibles sean comparables. Por la forma en que están diseñadas las elecciones, deberán modificar la igualdad para poder cumplir con dicho requisito.

Ejercicio 2

"Reglas claras convservan la amistad"

Implementar las reglas del famoso juego. Para ello, deberán completar y crear los métodos necesarios haciendo pasar el `test02`. En este ejercicio NO está permitido el uso de `if` (ni código que lo simule). En su lugar se recomienda utilizar la técnica *double dispatch*.

Sugerencia: observar los métodos dados en la clase `Eleccion`, y aquellos marcados para implementar por las distintas subclases de la misma.

[1] Tomar la metodología presentada como una introducción a la técnica. Para más detalles, recomendamos la materia POO.

Ejercicio 3

Dada una elección cualquiera se querrá obtener una que sea capaz de ganarle. Implementar este mecanismo haciendo pasar el test `test03`. Dado que podrían aparecer nuevas elecciones en el futuro, se espera que no se fije en la implementación de este punto cuáles son el total de elecciones ni qué respuesta se debería devolver para cada tipo de elección.

Sugerencia: utilizar metaprogramación para averiguar las subclases de `Eleccion`

Ejercicio 4

"Los hombres tienen una tendencia a sacar Piedra en la primera elección. Esto tiene que ver con la idea de que Piedra es percibido como 'fuerza' y 'energía' y los hombres tienden a caer en esto. Usa este conocimiento para conseguir una primera victoria fácil eligiendo Papel. Esta táctica es la mejor en partidas puntuales contra alguien que no ha jugado mucho."

Hacer pasar el `test04` para permitir enfrentar en un juego a dos jugadores. Los jugadores serán los objetos encargados en decidir qué elección utilizar al momento de jugar, dependiendo de distintos factores. Notar que el esqueleto provee la base para notificar a los jugadores sobre los resultados de sus juegos.

Ejercicio 5

En este ejercicio se pide implementar un jugador cuya estrategia se adapte a los resultados conseguidos en la jugada anterior. Si el jugador pierde o empata, se adaptará a esta situación suponiendo que el contrincante utilizará la misma elección nuevamente y por lo tanto escoje una elección ganadora. Tests: `test05` y `test06`

Ejercicio 6

"Las tijeras cortan el papel, el papel envuelve a la piedra, la piedra aplasta al lagarto, el lagarto envenena a Spock, Spock rompe la tijera, la tijera decapita al lagarto, el lagarto come papel, el papel impugna a Spock, Spock vaporiza la piedra y la piedra rompe la tijera. "

En este ejercicio se debe incluir la posibilidad de jugar contra Sheldon. Para ello deben agregarse dos elecciones más a las posibles. Ver el test `test07`. Sólo es necesario implementar lo necesario para pasar este test. Debería quedar claro como completarlo si fuese necesario.

Ejercicio 7

Implementar un jugador aleatorio. Este jugador deberá elegir aleatoriamente entre las distintas elecciones que haya. Es importante que el jugador funcione sin importar qué elecciones

existen (ya que, como hemos visto, podrían aparecer más en el futuro).

El test asociado a esta funcionalidad es `test08`, en donde para poder testear determinísticamente se utiliza un objeto polimórfico con el verdadero objeto generador de números pseudo aleatorios. http://xkcd.com/221/

Ejercicio 8

Implementar un jugador interactivo que defina su elección pidiéndole al usuario que decida mediante una interfaz gráfica. Este jugador también deberá informar al usuario el resultado de cada juego.

Sugerencia: utilizar los siguientes mensajes para mostrar el diálogo de elección y la notificación de los resultados

```
UITheme current
    chooseDropListIn: Morph new
    text: '¿Qué jugamos?'
    title: 'Nueva elección'
    list: <acá van las distinas elecciones>.

UITheme current
    messageIn: Morph new
    text: 'Le ganaste a <elección del jugador contrario>'
    title: 'Resultado'.
```

Ejercicio 9

Implementar los mensajes necesarios para permitir iniciar juegos del siguiente modo:

- `Juego mejorDe: 3 entre: unJugador y: otroJugador`
- `Juego mejorDe: 3 contra: unJugador`

El primer mensaje se utilizará para crear juegos entre dos jugadores cualesquiera, mientras que el segundo creará siempre un jugador humano para competir contra uno cualquiera. Alcanza con hacer que los juegos ocurran: no es necesario definir quién es el ganador resultante del total de rondas.

Mas datos útiles (¡y necesarios!):
http://scholar.google.com/scholar?q=rock+paper+scissors+game+theory
https://www.youtube.com/watch?v=Y6GPW1vUaqU
https://www.youtube.com/watch?v=dwj254ofJbk
https://www.youtube.com/watch?v=_tPxrskoK_E
https://www.youtube.com/watch?v=KK7X0yaXR20

Pautas de entrega

El entregable debe contener:

- un archivo `.st` con todas las clases implementadas
- versión impresa (legible) del código, comentado adecuadamente
- **NO** hace falta entregar un informe sobre el trabajo

Se espera que el diseño presentado tenga en cuenta los siguientes factores:

- definición adecuada de clases y subclases, con responsabilidades bien distribuidas
- uso de polimorfismo para evitar exceso de condicionales
- intento de eliminar código repetido utilizando las abstracciones que correspondan

Consulten todo lo que sea necesario.

Consejos y sugerencias generales

- Lean al menos el primer capítulo de *Pharo by example*, en donde se hace una presentación del entorno de desarrollo.
- Explorar la imagen de Pharo suele ser la mejor forma de encontrar lo que uno quiere hacer. En particular tengan en cuenta el buscador (`shift+enter`) para ubicar tanto métodos como clases.
- Pueden agregar tests propios (se recomienda agregar lo que utilicen para hacer sus propias pruebas).

Importación y exportación de paquetes

En Pharo se puede importar un paquete arrastrando el archivo del paquete hacia el intérprete y seleccionando la opción "`FileIn entire file`". Otra forma de hacerlo es desde el "`File Browser`" (botón derecho en el intérprete > `Tools` > `File Browser`, buscar el directorio, botón derecho en el nombre del archivo y elegir "`FileIn entire file`").

Para exportar un paquete, abrir el "`System Browser`", seleccionar el paquete deseado en el primer panel, hacer click con el botón derecho y elegir la opción "`FileOut`". El paquete exportado se guardará en el directorio `Contents/Resources` de la instalación de Pharo (o en donde esté la imágen actualmente en uso).