Orientación a Objetos 1 - 2017
Práctica 2

IMPORTANTE:
actualice el ambiente de trabajo ejecutando nuevamente:

```
ConfigurationOfBotArena loadDevelopment
```

Introducción:

En esta práctica deberá utilizar el ambiente del robot en una versión más avanzada, para ello copie, pegue en el Playground y ejecute el siguiente código:

```
BGSArenaWindow expertViewOn: (OnTheFlyConfigurableSimulation batteryWalkingBrush).
```

Ejercicio 1

El mundo de robots de esta práctica ya no tiene un robot al comenzar. El mismo debe agregarse explícitamente por medio del botón etiquetado “Agregar Robot” (Add Robot). Al agregar un robot debe indicar el nombre con el que se lo referenciará (en este caso sugerimos utilizar la variable `robotech`). También es posible agregar un robot haciendo click en cualquier posición de la arena; en este caso el robot se ubicará en la posición clickeada. Extienda el comportamiento del robot para que sea capaz de entender los siguientes mensajes. Compruebe que el robot `robotech` reacciona correctamente. Recuerde incluir el comentario del método en la definición de cada uno.

`0) #position:aPoint`

`“Posiciona al robot en la celda aPoint de la arena. Ayuda: ver def de #position”`

`1) #squareOfSize: aSize`

`“Realiza un cuadrado con una esquina en su posición actual y de lado aSize.”`

`2) #squareOfSize: aSize at: aPoint`

`“Realiza un cuadrado con una esquina en el punto aPoint y de lado aSize.”`

`3) #squareAtHomeOfSize: aSize`

`“Realiza un cuadrado con una esquina en 25@25 y de lado aSize.”`

`4) #rotatedSquareOfSize: aSize`

`“Realiza un cuadrado con una esquina en su posición actual, de lado aSize, rotado 45 grados.”`

`5) #rotatedSquareOfSize: aSize at:aPoint`

`“Realiza un cuadrado con una esquina en aPoint, de lado aSize, rotado 45 grados.”`

Nota: implemente los métodos re-utilizando los métodos previamente definidos.

Orientación a Objetos 1 - 2017
Práctica 2

Ejercicio 2

Agregue otro robot al que conocerá por medio de la variable `afrodita`. Compruebe que la robot afrodita entiende los mensajes definidos en el Ejercicio 1 y reacciona de igual forma que el robot `robotech`. ¿Qué sucede si modifica uno de los métodos? ¿Siguen comportándose ambos robots de igual manera? ¿Por qué? Discuta el por qué con un ayudante.

Ejercicio 3: PatrolCouple

Una PatrolCouple (pareja de guardia) está formada por dos robots: uno conocido como *patrol* (patrulla) y el otro conocido como *sniper* (francotirador).

1. Defina la clase PatrolCouple y cree una instancia en el Playground. PatrolCouple debe implementar el mensaje *#patrol: sniper:* que toma dos robots como parámetros y los guarda en las variables de instancia que correspondan.
2. Cree dos robos diferentes para que sean *patrol* y *sniper* respectivamente. Cree una instancia de PatrolCouple. Por último, envíe el mensaje *#patrol: sniper:* con los robots como parámetros de instancia de PatrolCouple.
3. Según lo visto en la teoría, implemente el o los métodos necesarios para que se puedan inicializar las instancias de PatrolCouple con los robots que lo conforman por medio de un método de clase que se encargue de la “creación e inicialización”.

Ejercicio 4: PatrolCouple con comportamiento

En este ejercicio agregaremos comportamiento a `PatrolCouple` con la siguiente definición de mensajes:

`#reset`

`"Ambos robots se posicionan enfrentados a distancia 5 uno del otro (uno mira al south y el otro al north, uno de ellos posicionado en 20@20"`

`#regularPatrol`

`"patrol hace un cuadrado de lado 10 rotado 45 grados alrededor de sniper, sniper en el centro gira en sentido de las agujas del reloj."`

`#regularPatrolTrace`

`"Similar a regularPatrol pero patrol realiza un trazo con el brush"`

`#doTheRegularPatrol`

`"Los guardias repiten regularPatrol 5 veces, pero luego de cada una se corren 5 hacia el este. Considere usar bateria con suficiente carga"`

`#doTheRegularPatrolTrace`

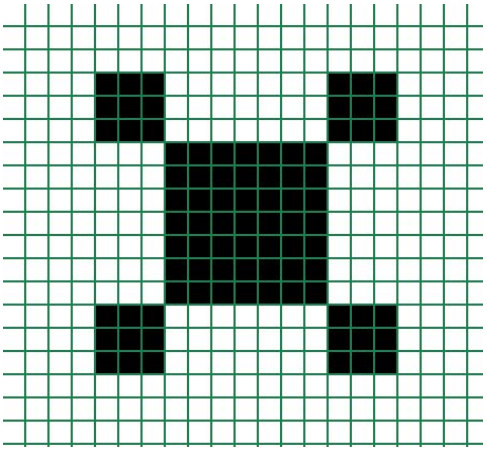
`"Similar a` **`doTheRegularPatrol`** `pero cada robot deja un trazo en la arena con el brush"`

1. Realice el diagrama UML de clases.

2. Implemente los métodos pedidos (no olvide agregar los comentarios a cada método). Y pruebe valiéndose del Playground y el Inspector que funcionen correctamente.

Ejercicio 5: Castle Watch

Castle's Watch es un antigua orden de robots que protegen un castillo.



Para lanzar el ambiente con el castillo antes debe:

1) Cerrar su simulación y Playground arena.
2) Actualizar el ambiente del robot evaluando la siguiente sentencia:

```
ConfigurationOfBotArena loadDevelopment
```

3) Lanzar el ambiente con la simulación de castillo:

```
BGSArenaWindow expertViewOn: (CastleSimulation
batteryWalkingBrush).
```

Defina la clase CastleWatch, la cual debe coordinar el accionar de los 4 guardianes conocidos como northWatch, southWatch, eastWatch y westWatch.
Cada guardián debe cubrir el correspondiente flanco del castillo.

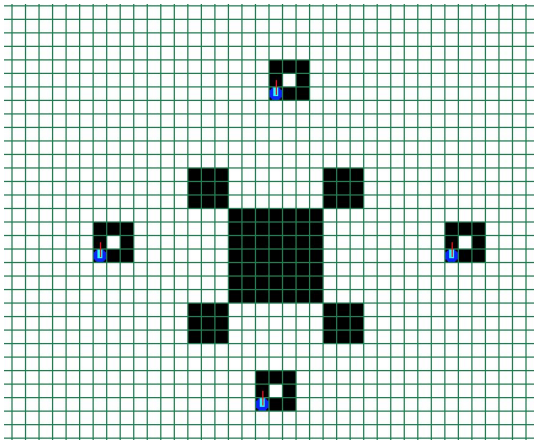
Implemente los siguientes métodos:

```
#regularWatch
```

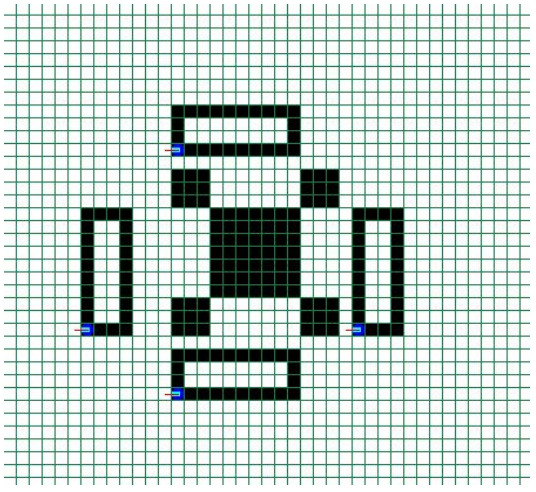
" los robots realizan un cuadrado de lado 3 en su correspondiente flanco."

#paranoicWatch
"cada robot realiza un recorrido del flanco completo, dibujando un rectangulo de 4x10"

Note que siempre los robots deben dejar el rastro de su patrullaje. Por ej. el resultado de regularWatch podria ser:



Mientras que el resultado del `paranoicWatch` podría ser:



Ejercicio 6: Wallpost

En este ejercicio abandonamos el mundo del robot y nos manejamos con el Browser del Sistema.

Orientación a Objetos 1 - 2017
Práctica 2

Debe desarrollar un post en un muro, al estilo de Facebook. Definimos un objeto Wallpost con los siguientes atributos: un texto que se desea publicar, cantidad de likes (“me gusta”) y una marca que indica si es destacado o no.

Defina la clase `Wallpost` en Smalltalk, con los siguientes mensajes:

```
#text
"Retorna el texto descriptivo de la publicación"

#text: aString
"Setea el texto descriptivo de la publicación"

#likes
"Retorna la cantidad de “me gusta”"

#like
"Incrementa la cantidad de likes en uno"

#dislike
"Decrementa la cantidad de likes en uno. Si ya es 0, no hace nada"

#isFeatured
"Retorna true si el post esta marcado como destacado, false en caso
contrario"

#toggleFeatured
"Cambia el post del estado destacado a no destacado y viceversa"

#initialize
"Inicializa el estado de las variables de instancia del Wallpost. Luego de la
invocación el Wallpost debe tener como texto: “Undefined post”, no debe estar
marcado como destacado y la cantidad de “Me gusta” deben ser 0."
```

Utilice el test provisto por la cátedra para comprobar que su implementación de `Wallpost` es correcta.

Ejercicio 7: Ventana del Wallpost

Una vez que su implementación pasa los tests del ej anterior puede utilizar la ventana que se muestra a continuación, la cual permite inspeccionar y manipular el post (definir su texto, hacer like y dislike, marcarlo como destacado).

Para abrir la ventana puede evaluar la siguiente expresión en el Playground:

```
WallpostUI on: (Wallpost new)
```

Orientación a Objetos 1 - 2017
Práctica 2

En la expresión que evaluó para abrir la ventana `WallpostUI on: (Wallpost new)` se instancian 2 objetos, el wallpost y la ventana. Discuta con un ayudante

1. ¿En qué difieren las instanciaciones?
2. En el ej. anterior ud.implementó el método #initialize, pero, ¿quien lo invoca? Ayuda: coloque un breakpoint en el método initialize para ver quién lo invoca. Ayuda de la ayuda: para poner un breakpoint agregue la sentencia `self halt.` al código del método `#initialize`.

Ejercicio 8: Leyendo código

Leer código ayuda a conocer convenciones sobre cómo se escribe el código en Smalltalk. Utilizando el browser y mirando cualquier clase o conjunto de clases y sus métodos responda:

1. ¿Los nombres de clase comienzan con minúscula o mayúscula?
2. ¿Cómo se escriben los nombres de métodos?
3. ¿Para qué se usan los protocolos?
4. ¿Qué pasa si un método no tiene un protocolo asignado?
5. Busque la clase DateAndTime y:
    a. Mencione al menos 3 clases a las que se haga referencia desde el código de la clase DateAndTime.
    b. Busque el método más largo de la clase. ¿Qué pasa en el browser cuando aparece un método largo? ¿Qué conclusión puede sacar al respecto?
    c. Busque un método que haga uso de variables temporales, y dos métodos que usen de instancia. Discuta con el ayudante sobre el uso que se les da a las variables temporales.

Ejercicio 9: Evaluación de expresiones

1. ¿Qué devuelve Smalltalk cuando se evalúan las siguientes expresiones? Realice el ejercicio en papel, tenga en cuenta el estado del ambiente producido por las evaluaciones previas.

```
x := 2 * 5 factorial.
y := x + 1.
|n| n := n+1.
|n m| n := 4. m := 1. ^(n+m+x+y).
5 timesRepeat: [x := x + y]. x
('objeto' at:2) isVowel.
```

Luego de hacer las evaluaciones en papel, compruebe los resultados copiando las expresiones en un Playground y evaluándolas con Pharo. Recomendamos usar el debugger.

2. Dado un triángulo rectángulo representado por las variables temporales *base* y *altura*, escriba las expresiones en Smalltalk para calcular:
    a. La superficie
    b. La hipotenusa
    c. El perímetro
    d. *true* si el perímetro es mayor a *unPerimetro* o *false* en caso contrario
    e. Implemente en Smalltalk y verifique su correcto funcionamiento.

3. Dada la clase Wallpost creada anteriormente, escriba las siguientes expresiones en Smalltalk:
    a. Cree una instancia de Wallpost, *unWallpost.*
    b. Incremente los likes de unWallPost hasta llegar a 20.
    c. Cree otra instancia de Wallpost, *otroWallpost*.
    d. Obtener el texto del Wallpost con más likes.
    e. Si el post *unWallpost* tiene más de 100 likes, márquelo como featured.
    f. Evalúe a true si ambos tienen más de 20 likes
    g. Cree una nueva instancia de Wallpost que sea la “concatenación” de ambos. Esto es que el texto debe ser la concatenación de los textos de ambos, sus likes deben ser la suma de ambos likes, y debe estar marcado como featured si al menos alguno de ellos lo está.

4. Dada una variable *aNumber*, escriba la expresión para calcular la suma de los primeros *aNumber* números naturales.

5. Dada la siguiente expresión:

```
3 + 5 > 6 ifTrue: [ 4 ] ifFalse: [ 5 ]
```

¿Qué valor se obtiene al ser evaluada? ¿Cómo la modificaría para obtener el valor 8?

Ejercicio 10:

Orientación a Objetos 1 - 2017
Práctica 2

Considerando la clase `DateAndTime`, busque ejemplos de métodos donde se utilizan paréntesis para forzar cierto orden particular en la evaluación de mensajes. Discuta con un ayudante, mirando el código, qué pasaría si los paréntesis no estuvieran allí.

En particular analise el código de estos métodos (tomados de Pharo) y determine si los paréntesis son necesarios o no:

```
asLocal

	^ (self offset = self class localOffset)
		ifTrue: [self]
		ifFalse: [self offset: self class localOffset]


julianDayOffset
	"Return the offset in julian days possibly introduced by the timezone offset"

	^ ((seconds + self offset asSeconds) / SecondsInDay) floor
```

Ejercicio 11:

Implemente un nuevo tipo batería que se llama EnergyRecoveryCell. Este tipo de batería tiene el mismo comportamiento que la clase Battery pero además se recarga mientras el robot se mueve. Una instancia de EnergyRecoveryCell recarga 1 unidad de energía por cada 10 unidades de energía consumidas. Tenga en cuenta que es indistinto si las 10 unidades se consumen en una sola movida del robot o en varias.

Verifique su implementación utilizando el test case provisto por la cátedra.