Trabajo Práctico Integrador Nro. 1

Unidad 4: Paradigma Orientado a Objetos

Objetivos:

- Integrar los contenidos prácticos correspondientes a la programación orientada a objetos.
- Reforzar las destrezas necesarias para resolver una situación problemática utilizando la programación orientada a objetos en Smalltalk, utilizando Pharo como implementación.
- Modelar una solución de un problema mediante el paradigma orientado a objetos.

Los temas a evaluar en este trabajo práctico son los que se detallan a continuación: Objetos y mensajes en Smalltalk. Uso de ventanas en Smalltalk. Mensajes unarios, binarios y de palabras claves. Bloques. Composición de clases. Abstracción. Encapsulamiento. Herencia. Polimorfismo. Colecciones.

Caso de Estudio:

La Internet de las cosas **o IOT** (en inglés, **I**nternet **O**f **T**hings) es un concepto que se refiere a una interconexión digital de objetos cotidianos con internet. Los objetos conectados pueden recibir o enviar ciertos datos provenientes de uno o más sensores conectados con él.
Una empresa de la ciudad que ofrece servicios de implementación de sistemas IOT nos solicita desarrollar una aplicación Smalltalk que permita simular un conjunto de objetos inteligentes responsables de la automatización de una vivienda. Cada vivienda tiene un conjunto de objetos inteligentes encargados de gestionar ciertos artefactos o sistemas existentes (como las luces perimetrales o el sistema de filtrado de la piscina) y generar alarmas según su función. Estos objetos pueden ser de dos tipos:

- Objetos de monitoreo: que son capaces de tomar información mediante sus sensores y generar ciertas alarmas cuando ocurren ciertos eventos preconfigurados. Ejemplos: una cámara inteligente o un medidor de consumo de electricidad.
- Objetos actuadores: que permiten activar ciertos sistemas de control a partir de ciertas señales claves y emitir alarmas al respecto. Por ejemplo, un activador de un portón elevadizo o un activador de luces.

En base al relevamiento del dominio, los analistas han encontrado y documentado las siguientes clases en formato ficha, y usted deberá programar:

Clase: Alarma	
Descripción: Entidad que describe una alarma generada por un objeto inteligente.	
Atributos	**Comportamientos**
• **tipo** Hace referencia al tipo de alarma que puede ser 1-Normal o 2-Evento atípico. • **hora** Es un valor entero que indica la hora de la alarma en formato Hora Militar: hhmm. Por ejemplo, las 3 pm se representan como 1500, las 11 am como 1100, etc. • **mensaje** Descripción corta del evento informado.	• Inicialización. • Mensajes de acceso y modificación. • **mostrarDatos** Comportamiento que devuelve una cadena con la descripción completa de la alarma.

Clase: ObjetoInteligente	
Descripción: Entidad que representa a un objeto inteligente que pertenece a una vivienda.	
Atributos	**Comportamientos**
• **id** Valor alfanumérico que identifica unívocamente un objeto inteligente. Ej.: PL001. • **estado** Representa el estado del objeto inteligente. Los únicos valores que puede asumir son: activo o inactivo. • **nombre** Conjunto de caracteres que representa el nombre del objeto inteligente. • **alarmas** Colección de alarmas del objeto inteligente.	• Inicialización. • Mensajes de acceso y modificación. • **tomarComando**:**unValor** Recibe como colaborador externo un valor tomado por uno de los sensores ubicados en la vivienda. Según el valor recibido el objeto genera y registra una alarma con los datos del evento notificado. • **totalAlarmas:tipo** Retorna un valor numérico que representa la cantidad de alarmas del objeto inteligente del tipo que se recibe como colaborador externo. • **mostrarEstado** Devuelve una cadena con los datos completos del objeto inteligente.

Clase: Actuador	
Descripción: Entidad que describe un tipo de objeto inteligente específico. Esta entidad hereda todo el comportamiento y atributos de ObjetoInteligente.	
Atributos	**Comportamientos**
• **valorMinimo** Valor numérico que indica el valor de referencia a partir del cual el objeto **actuador** cuando recibe el mensaje **tomarComando:unValor** genera la alarma correspondiente.	• Inicialización. • Mensajes de acceso y modificación. • **tomarComando:unValor** Permite generar la alarma correspondiente según el valor recibido. Si el valor es inferior o igual al **valor mínimo**, el objeto genera con el mensaje "Valor mínimo alcanzado. Activar" y tipo:2 (evento atípico). En caso contrario se genera una alarma de tipo 1 con el mensaje: "Valor recibido: **unValor**" • **mostrarEstado** Devuelve una cadena con los datos de la entidad Actuador.

Clase: Monitor	
Descripción: Entidad que describe un tipo de objeto inteligente específico. Esta entidad hereda todo el comportamiento y atributos de ObjetoInteligente.	
Atributos	**Comportamientos**
• **limiteInf y limiteSup** Valores numéricos que indican los límites de referencia para los cuales el objeto entiende que los el valor recibido a través del mensaje **tomarComando:unValor** son normales. Es decir, definen el rango de valores normales para el objeto monitor.	• Inicialización. • Mensajes de acceso y modificación. • **tomarComando:unValor** Permite generar la alarma correspondiente según el valor recibido. Si el valor se encuentra fuera del rango [limiteInf:limiteSup], el objeto genera con el mensaje "Valor fuera de rango. Activar" y tipo 2. En caso contrario se genera una alarma de tipo 1 con el mensaje: "Valor de lectura: **unValor**" • **mostrarEstado** Devuelve una cadena con los datos de la entidad Monitor.

<table>
<tr><td colspan="2">Clase: Vivienda</td></tr>
<tr><td colspan="2">Descripción:
Entidad que representa una vivienda.</td></tr>
<tr><td>Atributos</td><td>Comportamientos</td></tr>
<tr><td>

- **direccion**

  Es la dirección de la vivienda. Incluye nombre de la calle y altura. Ej.: San Martin 1212.
- **duenio**

  Es una cadena que contiene el nombre y apellido del dueño de la vivienda.
- **objetos**

  Es una colección que representa al conjunto de objetos inteligente que tiene la vivienda.

</td><td>

- Inicialización.
- Mensajes de acceso y modificación.
- **agregarObjeto:unObjeto**

  Recibe como colaborador externo un objeto inteligente y lo agrega a la colección de objetos de la vivienda. Se debe validar que no pueden registrarse dos objetos con el mismo **id**.
- **quitarObjeto:unId**

  Recibe como colaborador externo el id de un objeto inteligente y lo quita de la colección de objetos de la vivienda.
- **estaActivo:unId**

  Retorna true si el objeto con el id especificado como colaborador externo al método, está en estado activo. Retorna false en cualquier otro caso.
- **tomarComando:unValor para:id**

  Permite tomar un valor recibido de uno de los sensores de la vivienda y enviarlo al objeto inteligente correspondiente según su **id.**
- **listarAlarmaEn:hrDesde y:hrHasta**

  Retorna un listado ordenado cronológicamente con todas las alarmas que se registraron en un determinado lapso, comprendido entre una hora desde (hrDesde) y una hora hasta (hrHasta).
- **calcularPorcentajeInactivos**

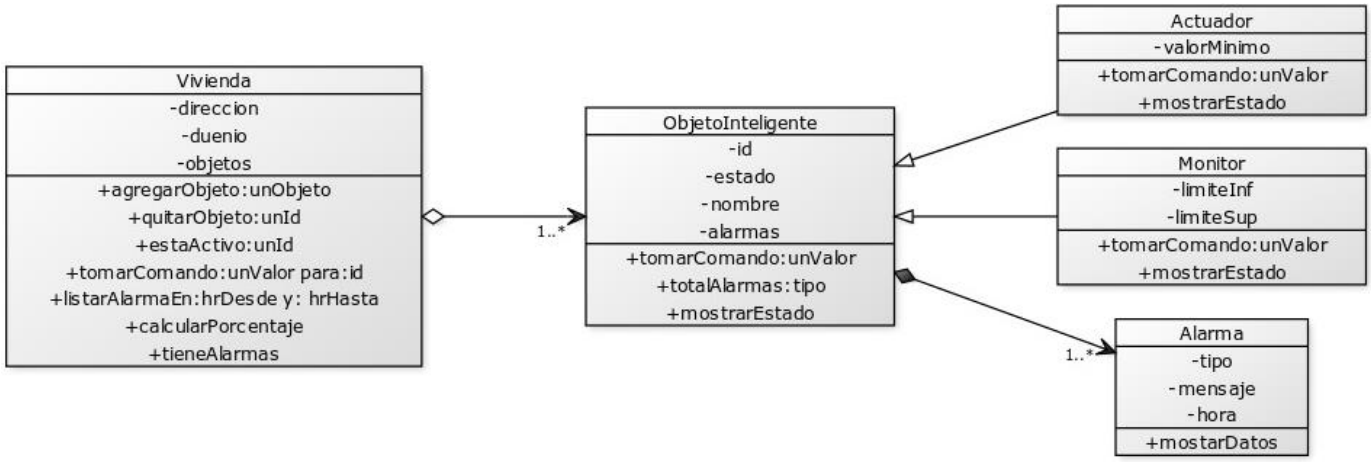
  Retorna un valor numérico que es el porcentaje de objetos actuadores inactivos que posee la vivienda. Este porcentaje se calcula como el cociente entre la cantidad de objetos inteligentes de tipo Actuador inactivos y la cantidad total de objetos inteligentes de la Vivienda.
- **tieneAlarmas**

  Retorna true si la vivienda tiene al menos una alarma y false si la vivienda no tiene alarmas.

</td></tr>
</table>

A continuación, se propone un diagrama de clases que se podrá utilizar como guía para la implementación del programa:

Diagrama de Clases



Código fuente (yuml.me):

[Vivienda|-direccion;-duenio;-objetos|+agregarObjeto:unObjeto;+quitarObjeto:unId;+estaActivo:unId;+tomarComando:unValor para:id;+listarAlarmaEn:hrDesde y: hrHasta; +calcularPorcentaje;+tieneAlarmas]

[ObjetoInteligente|-id; -estado; -nombre;-alarmas|+tomarComando:unValor;+totalAlarmas:tipo;+mostrarEstado]

[Actuador|-valorMinimo|+tomarComando:unValor;+mostrarEstado]

[Monitor|-limiteInf;-limiteSup|+tomarComando:unValor;+mostrarEstado]

[Alarma|-tipo; -mensaje; -hora|+mostarDatos]

[ObjetoInteligente]^[Actuador]

[ObjetoInteligente]^[Monitor]

[Vivienda]<>-1..>[ObjetoInteligente]*

[ObjetoInteligente]++-1..>[Alarma]*

En base a lo expuesto anteriormente usted debe desarrollar un programa orientado a objetos en Smalltalk que satisfaga los siguientes requerimientos:

1. Implemente las clases indicadas en las fichas con todos sus atributos y responsabilidades.
2. En la clase ObjetoInteligente y su jerarquía, implemente los comportamientos polimórficos: mostrarEstado y tomarComando:unValor, definidos anteriormente.
3. Registrar un objeto en una vivienda, garantizando que los objetos son únicos por Id dentro la misma. Se necesita además desarrollar una funcionalidad para quitar un objeto.
4. Determinar si un objeto de la vivienda identificado por **id** se encuentra activo o no.
5. Enviar un valor de lectura obtenido por un sensor de la casa a un objeto inteligente específico.
6. Listar todas las alarmas que se registraron en un determinado rango de horas, ordenadas cronológicamente.
7. Calcular el porcentaje de objetos actuadores inactivos de la vivienda respecto del total de objetos presentes.
8. Validar si la vivienda tiene o no alarmas generadas.

Se sugiere dividir en el trabajo en etapas, de manera que pueda ir resolviendo los requerimientos solicitados a medida que avanzan las clases teórico-prácticas de la materia, tal como se indica a continuación:

- **Etapa 1:** desarrollar la clase Alarma con método: de inicialización, de acceso y mostrarDatos.
- **Etapa 2:** desarrollar jerarquía de clases ObjetoInteligente, Actuador y Monitor sin tener en cuenta la redefinición de los mensajes mostrarEstado y tomarComando.
- **Etapa 3:** Completar la jerarquía de ObjetoInteligente y desarrollar la clase Vivienda con todo el comportamiento requerido.

Solo para instancia de recuperación

En caso de haber desaprobado el trabajo en su primera instancia de entrega el alumno deberá corregirlo y entregarlo nuevamente en la fecha acordada incluyendo el desarrollo de los siguientes requerimientos adicionales:

- Determinar la cantidad promedio de alarmas generadas de tipo 2 tanto por objetos monitores como actuadores.
- Listar todos los objetos de la vivienda con un número de alarmas superior a un valor de referencia recibido en un rango de horas, también enviados como colaboradores externos.

Consignas generales:

- Defina adecuadamente la jerarquía de clases, y asigne en cada clase los atributos y métodos que correspondan según el criterio del grupo.
- Realice la codificación completa en Smalltalk de todas las clases involucradas en el diseño de la solución problemática. Debe haber coherencia total entre el diagrama de clases propuesto y la implementación en Smalltalk. Usted puede reformular el diagrama propuesto, en tal caso deberá presentarlo como parte del trabajo. Siéntase libre de agregar todo el comportamiento extra en las clases que considere necesario.
- Escriba las líneas de código necesarias en la ventana Playground para hacer funcionar el programa y verifique en la ventana Transcript todas las salidas de información generadas.
- Reutilice adecuadamente los comportamientos que sean necesarios.
- Tenga presente la delegación de responsabilidades para cada requerimiento según corresponda.

Criterios de evaluación:

- Identificación de todas las clases involucradas, y asignación correcta de atributos y responsabilidades de cada una.
- Adecuado diseño del diagrama de clases: jerarquía de clases y demás relaciones bien especificadas y representadas, etc.
- Claridad y completitud del diagrama de clases.
- Prolijidad en la codificación en Smalltalk: identificadores, comentarios, envío de mensajes, etc.
- Manejo adecuado de **TODAS** las propiedades esenciales de la programación orientada a objetos en la resolución propuesta.
- Delegación apropiada de responsabilidades en las clases involucradas.
- Reutilización conveniente de los comportamientos implementados.
- Validaciones.
- Correcta utilización de métodos:
  - ✓ unarios;
  - ✓ binarios;
  - ✓ de palabras clave: de una, dos, y más palabras claves.

  Se deberán utilizar todos estos tipos de mensajes en la resolución propuesta en forma adecuada.
- Elección y uso apropiado de las colecciones en cada caso.
- Variedad de mensajes utilizados de las diferentes colecciones utilizadas, y de otros objetos.
- Uso de mensajes específicos tanto de colecciones como de otros objetos, para cada caso.
- Correcta identificación y usos de métodos polimórficos.
- Generación y visualización correcta de todas las salidas de información.
- Cumplimiento de todas las responsabilidades solicitadas.
- Realización de las pruebas solicitadas en el Playground, instanciando adecuadamente los objetos, y realizando las colaboraciones necesarias. Recuerde que los resultados del envío de los mensajes a los objetos se deberán visualizar en la ventana Transcript.

Tabla de valoración de los ítems evaluados

Nro. de ítem	Ítems o requerimiento a evaluar	Puntaje	Observaciones	Obtenido
1	Construcción del modelo de clases, con métodos de acceso y constructores.	10		
2	Definición de métodos polimórficos en la jerarquía objetos inteligentes.	10		
3	Implementación del comportamiento "agregarObjeto:unObjeto" en la clase Vivienda.	5		
4	Implementación del comportamiento "quitarObjeto:unId" en la clase Vivienda.	5		
5	Implementación del comportamiento "estaActivo:unId" en la clase Vivienda.	5		
6	Implementación del comportamiento "listarAlarmaEn:hrDesde y:hrHasta" en la clase Vivienda.	15		
7	Implementación del comportamiento "calcularPorcentajeInactivos" en la clase Vivienda.	10		
8	Implementación del comportamiento "tieneAlarmas" en la clase Vivienda.	5		
9	Implementación del comportamiento "totalAlarmas:tipo" en la clase ObjetoInteligente.	10		
10	Implementación del comportamiento "tomarComando:unValor para:id" en la clase vivienda.	15		
11	Desarrollo de una simulación completa en el Playground donde se haga uso de todos los comportamientos.	10		
	Total	**100**		

Condiciones de entrega:

- Este trabajo práctico integrador **se deberá realizar en forma grupal**.
- Deberán nombrar el **archivo comprimido** con el siguiente formato: TP1_NroLeg1erIntegranteApellido1erIntegrante_NroLeg2doIntegranteApellido2doIntegrente_NroLeg3erIntegranteApellido3erIntegrante. El orden en el cual deberán colocar los legajos y apellidos cada integrante en el nombre del archivo comprimido será de acuerdo al orden alfabético de los apellidos de los integrantes, desde la A hasta la Z.
- Se deberá subir una carpeta comprimida que contenga en su interior el archivo .ST correspondiente a la codificación en Smalltalk; un archivo de texto con el código necesario a colocar en la ventana Playground para hacer funcionar el programa; y el diagrama de clases completo correspondiente a la solución de la situación problemática anteriormente explicada siempre que haya modificado el propuesto por los docentes, o bien haya modelado uno diferente.
- Plazo máximo de entrega de este trabajo: **HASTA el 07/09/21 a las 22:00 hs.**
- Todas las consultas de este TP1 pueden realizarlas a través del foro del aula virtual.
  ¡Éxito a todos!