TRABAJO PRÁCTICO NRO 1

Respondido eligiendo entre las opciones de Smalltalk

1. La opción 2 es la correcta debido a que los atributos y la funcionalidad están separadas, haciendo esto más fácil la lectura del código como también mejora su representación.
2. La opción 2 es la correcta debido a que por temas de seguridad es conveniente que una clase no posea todos los datos a la mano, es conveniente delegar a otra clase.
3. La opción 4 es la correcta debido a que la clase CuentaBancaria maneja estas clases hijas (CajaDeAhorro y CuentaCorriente) sin distinción a pesar de que sus mensajes de extracción se comporten diferentes.

Actividad de lectura #1

1. ¿Qué significa el acceso directo a las variables? De un ejemplo

El acceso directo a las variables significa que la clase llama a su atributo directamente.

Ejemplo en código java:

Persona {

public String nombre;

public integer edad;

public bool esMayorDeEdad(Persona persona ){

return persona.edad > 18;

}

}

1. ¿Qué significa el acceso indirecto a las variables? De un ejemplo

El acceso indirecto se hace mediante getters and setters donde son mensajes se encarga de traer/modificar un atributo en específico de la clase.

Ejemplo en código java:

Persona {

public String nombre;

public Integer edad;

public Integer getEdad(){

return edad;

}

public void setEdad(Integer edad){

edad = edad;

}

}

Facultad {

public bool puedeInscribirse(Persona persona ){

return persona.getEdad() > 18;

}

}

1. ¿Qué ventajas y desventajas presenta cada estrategia a los getters y setters?

***Ventaja***

Ayudan a que sea más fácil la modificación en el código, por ejemplo:

* A la hora de modificar un mensaje que siempre recibió y guardo un mismo tipo. Y ahora se requiera guardarlo en algun tipo diferente. Para aplicar esa modificación tenemos que implementar los cambios en un único mensaje, será el get/set según nuestra decisión de diseño.

También se pueden modificar los setters para que sean privados por si requerimos de atributos que no sean modificables fuera de la clase. O al contrario modificar los getters para que solo sean modificables.

Además de que evita la repetición de mensajes innecesarios, ejemplo:

* Sin getters/setters podríamos llegar a tener un mensaje getNombre() , obtenerNombre(), elNombre() que cumplan la misma funcionalidad
* Con getters/setters sabemos que con get \*Y el atributo que necesitemos\* de la clase correspondiente obtendremos su valor.

***Desventaja***

Crean una cantidad de métodos que tal vez la clase no llegue a usarlas.

Actividad de lectura #2

En base al siguiente fragmento, responda: ¿en qué situación es conveniente utilizar el "Creation Parameter Method"?

"Creation Parameter Method" es conveniente utilizarlo en el momento en que se inicializa un objeto. Como dice en el texto, dilo toda una sola vez, hace referencia a cuando creamos una instancia de un objeto y le pasamos todos sus parámetros una única vez.

Actividad de lectura #3

En base al siguiente fragmento, responda: ¿cómo se debe proporcionar acceso a variables que referencian a una colección?

Nunca un acceso directo a la colección. El acceso a las colecciones debe ser lo más restringida posible, evitando la modificación total de esta

public void List<Personas> setPersonas( List<Personas> personas) {

this. Personas = personas;

}

No es conveniente que existe un set de este estilo ya que es muy propenso a sufrir errores, es conveniente crear formas de acceso por un solo elemento de la colección.

Actividad de lectura #4

En base al siguiente fragmento, responda: ¿por qué son necesarios dos métodos para asignar el estado a una propiedad booleana?

Son necesarios dos métodos para la asignación de estado en una propiedad booleana, uno que cambie el estado y otro que denote un estado especifico ejemplo: “isOn()”.

Debido a que, si creamos un get y set de este pueden suceder incongruencias y confusión sobre el estado actual. Además de que complica la lectura de código.

Ejemplo si, un tenemos una clase Servicio de internet que se puede de dar de alta y de baja cuando queramos. Tendríamos que estar pasando por parámetros true o false en los casos respectivos, esto dejándonos propensos a equivocaciones, en cambio si aplicamos un isOn y un alternar(), no necesitamos saber qué valor posee.