Prefijos de miembros

Tampoco es necesario añadir `m_` como prefijo a los nombres de variables. Las clases y funciones tienen el tamaño necesario para no tener que hacerlo, y debe usar un entorno de edición que resalte o coloree los miembros para distinguirlos.

```
public class Part {
   private String m_dsc; // La descripción textual
   void setName(String name) {
     m_dsc = name;
   }
}
```

```
public class Part {
   String description;
   void setDescription(String description) {
     this.description = description;
   }
}
```

Además, los usuarios aprenden rápidamente a ignorar el prefijo (o sufijo) y fijarse en la parte con sentido del nombre. Cuanto más código leemos, menos nos fijamos en los prefijos. En última instancia, los prefijos son un indicio de código antiguo.

Interfaces e Implementaciones

Existe un caso especial para usar codificaciones. Imagine por ejemplo que crea una factoría abstracta para crear formas. Esta factoría será una interfaz y se implementará por medio de una clase concreta. ¿Qué nombres debe asignar? `¿IShapeFactory` y `ShapeFactory`? Prefiero las interfaces sin adornos. La I inicial, tan habitual en los archivos de legado actuales es, en el mejor de los casos, una distracción, y en el peor, un exceso de información. No quiero que mis usuarios sepan que se trata de una interfaz, solamente que se trata de `ShapeFactory`. Si tengo que codificar la interfaz o la implementación, opto por ésta última. Es mejor usar `ShapeFactoryImp` o incluso `CShapeFactory`, que codificar la interfaz.