****

1. ****
2. **Licenciatura en**
3. **Sistemas de Información**

|  |
| --- |
| 1. **Programación Orientada** 2. **a ObjetoS** |

1. **TP N.º 6 - Herencia**
2. **Grupo N°: 1.**
3. **Integrantes:**
4. **Barbera Mateo.**
5. **Colombo Pérez Tomas.**
6. **Gonzales Ignacio.**
7. **Montenegro Ignacio.**
8. **Docente:**

Ing. Adrián Tournour

1. La herencia fomenta la reutilización de software, ahorra tiempo para el desarrollo de programas y ayuda a prevenir posibles errores.

Ya que la herencia es un mecanismo que nos ofrece solución a nuestro problema de duplicación de código. La idea se basa en, si tenemos 2 clases, CD y DVD por ejemplo, podremos definir una clase que contenga todas las cosas que tengan en común ambas clases, a la cual podemos llamar ELEMENTO. Luego declaramos que un CD es un ELEMENTO y que un DVD es un ELEMENTO.

Después de esto, agregaremos en cada clase los detalles adicionales que requieran cada una.

Lo esencial de esta técnica es que solo necesitamos describir las características comunes una sola vez.

1. a) Verdadero

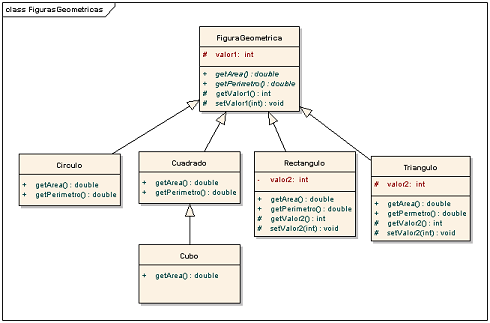
b) Falso. Una relación “es un” se implementa mediante la herencia.

c) Falso Las clases Volante, Dirección y Frenos tienen una relación "es un" con la clase Auto

d) Verdadero

e) Falso La sobrecarga supone que el método tiene el mismo nombre pero los parámetros son distintos (tipo y/o cantidad)

1. Herencia se podría utilizar si se detecta comportamientos similares, además de detectar la relación “es un”



1. Si la jerarquía de clases es demasiado compleja, el programador puede tener problemas para comprender el funcionamiento de un programa. Además, puede volverse más complejo detectar y resolver errores de programación, por ejemplo al modificar una clase padre que afecta el funcionamiento de las subclases.

Otro problema es que las subclases se deben definir en código, por lo que los usuarios del programa no pueden definir subclases nuevas. Otros patrones de diseño permiten que los usuarios puedan definir variantes de una entidad en tiempo de ejecución.

1. Cuando definimos un atributo o método como protected solo podrán tener acceso las clases y objetos que pertenezcan al mismo paquete, sean objetos de la misma clase o a las subclases y sus respectivos objetos

6) A la hora de crear una aplicación la herencia supone:

-Evitar la duplicación de código: evita escribir códigos idénticos o muy parecidos varias veces.

-Reutilizar código: podemos crear una subclase de una clase ya existente para evitar repetir código.

-Facilitar el mantenimiento: facilita el mantenimiento ya que un cambio en una clase se verá reflejado en las subclases.

-Facilitar la extensibilidad: hace más fácil la extensión de una aplicación.