**Trabajo Práctico Nº 10**

1. La expresión se refiere a que nuestro propio lenguaje es una abstracción. Los lenguajes de programación fueron evolucionando de lenguajes de nivel mas bajo para darle ordenes a la computadora, fueron variando de solo 1 y 0 a comandos y después algo similar a nuestro lenguaje con el que nos comunicamos, es una abstracción de las cosas que estan pasando cuando nosotros escribimos una función o una variable. Podra evolucionar de muchas formas pero lo que esta pasando por debajo que nosotros no vemos, seguirá igual.
2. La programación orientada objetos es una forma de representar los programas y sus partes de forma similar a como estan construidos y como se comportan en la vida real, esto lleva a que la programación cada dia se aleja mas de esa imagen de una ciencia estructurada y super rigida que solo maneja 1 y 0. Lo lleva a algo mas parecido a nuestra imaginación o a otra forma de expresión artística como puede ser la música o la escultura.
3. Una clase es la descripción que comparten varios objetos para poder agruparlos. Una clase esta conformada por todos nuestros objetos que comparten cosas como por ejemplo varias formas, podemos tener objetos cuadrados, triángulos y círculos pero todos ellos van a pertenecer a la clase FORMAS, y comparten cosas en común y por eso pueden agruparse.
4. Un objeto podría decirse que es una variable que almacena datos y también puede trabajarlos en si mismo, todos los objetos de una misma clase se comportan de formas similares y se supone que nuestro programa debería poder trabajar cualquiera de estos objetos, por mas que sea diferentes, de la misma forma sin ningún problema.
5. El método imperativo esta armado de una forma similar a una receta, en la que la computadora guarda estados y nuestro programa los va modificando para ir consiguiendo distintos resultados y resolviendo diferentes problemas.  
   El POO define a los programas como comunidades de objetos conformados por pro su estado(datos) y su comportamiento(métodos) ósea que son y que hacen, los objetos similares entre si son agrupados en clases. El POO trabaja con los objetos interactuando entre si para resolver el problema.
6. Una abstracción es una representación de algo, de cualquier cosa, solo mostrando sus partes principales y mas visibles. Nosotros la usamos para representar un objeto real en un objeto dentro de un programa solo usando sus partes principales y dejando ver solo lo mas básico. Esto acerca mucho mas el lenguaje del programa al lenguaje del problema que estamos tratando, si un arquitecto esta viendo un programa para ayudarlo a construir algo, va a poder interactuar directamente con objetos que reconoce y puede trabajar usando el lenguaje propio del problema.
7. Las clases y los objetos son una abstracción de un objeto real, un modelo virtual que es una representación de la realidad y como en la vida los objetos estan separados según sus atributos o características en diferentes clases, como son por ejemplo todos los diferentes tipos de mesas, todas las diferentes sillas, son objetos diferentes pero forman parte de la misma clase.
8. Algunos lenguajes orientados a objetos son Smalltalk, C++, C#.  
   Alan Key resumio 5 puntos básicos de un lenguaje orientado a objetos. Todo es un objeto, hay que ver los objetos como variables muy elaboradas que contienen muchos datos y es una representación de un problema que se esta intentando resolver.  
   Un programa es un conjunto de objetos enviándose mensajes entre si, para solucionar un problema.  
   Cada objeto puede estar constituido por diferentes objetos y podemos crear objetos mas elaborados agrupando dentro de el otros objetos.  
   Cada objeto tiene un tipo(lo que llamaríamos una clase a la que pertenece), Lo que diferencia a un objeto de una clase es que tipo de mensajes puedes enviarle?.  
   Todos los objetos de un mismo tipo pueden recibir los mismos mensajes, se refiere a que si yo le envió una petición a un objeto silla, independientemente de que silla sea de todas las que conforman la clase, puedo enviarles el mismo mensaje y todas van a poder responderme.
9. La interfaz es la forma que tenemos para interactuar con los objetos que tenemos, son todas las acciones que pueden realizar nuestros objetos.
10. Si nosotros separamos la interfaz de la implementación evitamos que se cree código basado en la implementación, lo único que tenemos que dejar visible es la interfaz y que el resto del código se base en eso, debemos declarar la implementación en otro archivo.
11. Por que todo objeto tiene una utilidad y una forma de usarse. Cuando nosotros creamos un objeto es para hacer algo no solo para que este ahí sin hacer nada, no nos aporta nada eso. La interfaz establece que peticiones le podemos hacer a ese objeto por lo tanto todos los objetos tienen una interfaz.
12. El encapsulamiento es solo mostrar lo que hace falta mostrar. Si solo tiene acceso al uso del objeto no generara una dependencia a todo su contenido y se pueden hacer cambios a futuro en los objetos en la parte oculta sin afectar al resto del programa
13. Es útil hacer una distinción entre creadores de clases y programadores clientes. Los primeros son aquellos que crean nuevos tipos de datos; y los últimos usan estos tipos de datos predefinidos en sus aplicaciones. Esto se hace a fin que el creador de clases pueda en un futuro realizar cambios en dichas clases sin preocuparse de las consecuencias. La parte oculta representa una sólida codificación en términos de optimización, pero resulta frágil en otro sentido, ya que puede fácilmente corromperse por un programador cliente descuidado o desinformado. Ocultando la implementación se reducen estos errores de programación.
14. El ámbito de visibilidad es que tanto y que partes del código damos acceso nosotros.  
    Se relaciona directamente con lo hablado anteriormente, usando estas herramientas el creador de clases pone las reglas de que tanto puede ver y manipular el programador cliente.
15. Public quiere decir que esta disponible para cualquiera. Private quiere decir que solo el creador del tipo puede acceder, ósea solo puede ser manipulado por los métodos internos de la clase y no por el programador, si este intenta utilizarlos va a recibir un error de compilación. Protected es similar al private pero las clases derivadas tienen acceso a los miembros protegidos, mas no a los privados.
16. Si escribimos bien no debemos escribir dos veces, se refiere a que si tenemos un código funcional y eficiente, no debemos repetirlo muchas veces en el código, tenemos que utilizarlo y llamarlo cuando haga falta. Lo mismo pasa con los objetos y las clases, un objeto puede estar conformado por otros objetos, como una casa puede ser un objeto que dentro de si tenga otros objetos como puerta, ventanas, etc.
17. Cuando nosotros creamos una clase con sus diferentes atributos y funciona perfecto, es una pena tener que tomarse las molestias de escribir otro de nuevo para algo que es muy similar, entonces que hacemos? creamos una clase hija. La clase hija hereda todo lo que tiene la clase madre de la que deriva pero agregando sus diferencias y detalles.
18. El principio de sustitución hace referencia a que si B es un subtipo de A, entonces todos los objetos de A pueden ser sustituidos por los objetos de B sin ningún tipo de alteración en el programa.  
    Cuando comparamos usando el es-un lo hacemos para saber si es correcta la sustitución que estamos, como decir “La silla ES UN mueble”  
    En cambio si usamos el es-como-un queremos decir que si bien son similares y comparten muchas cosas, necesita agregar cosas a su interfaz que su clase padre no las tenia.
19. Por que se usan de la misma forma. Si nosotros tenemos varios objetos figura(circulo, triangulo, cuadrado) en cualquier parte del código que nosotros llamamos al circulo también puede ser llamado al triangulo sin ningún tipo de problema.
20. El ciclo de vida de un objeto es que tanto tiempo vamos a estar necesitándolo en nuestro programa, todo ocupa espacio en memoria o disco y estos recursos no son infinitos, tenemos que manejarlos con cuidado y esta tarea por lo general se deja a criterio del programador para decidir el tiempo que tiene de utilidad el objeto.
21. Cuando creamos nuestro programa no sabemos cuanto espacio de memoria vamos a necesitar y reservarlo todo antes. Si necesitamos espacio nosotros lo reservamos con la palabra new en el que literalmente estamos creando este objeto hasta que cumpla su tiempo de uso, cuando este cumple su función lo descartamos usando la palabra delete para destruirlo y liberar el espacio en memoria.