|  |  |  |  |
| --- | --- | --- | --- |
| ***PROGRAMACIÓN ORIENTADA A OBJETOS*** | ***CARACTERÍSTICAS*** | ***VENTAJAS*** | ***DESVENTAJAS*** |
| ***ABSTRACCIÓN.-*** Es la generalización conceptual de un determinado conjunto de objetos y de sus atributos y propiedades. | ***REUSABILIDAD.-*** Se puede usar las clases adecuadamente en distintas partes del programa y en numerosos proyectos. | Los programas no pueden ser modelados enteramente por la programación orientada a objetos. |
| ***ENCAPSULACIÓN.-*** Es reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, permitiendo a todos los elementos ser coherentes en el sistema. | ***MANTENIBILIDAD.-*** Resalta los detalles más relevantes haciendo el programa más sencillo de leer y comprender. | Para leer, modificar o cambiar algo, en algunos programas se debe realizar un paso extra para ejecutarlo, el uso de bibliotecas de las clases obliga a su aprendizaje y entrenamiento. |
| ***HERENCIA.-*** Realiza una estructura jerarquizada de las clases, promueve que las relaciones entre objetos, es un estrato básico del paradigma orientado a objetos. | ***MODIFICABILIDAD.-*** Es fácil de añadir, suprimir y modificar nuevos objetos de formar sencilla. | Si se exige el lenguaje puede perder algunos objetos y características, haciéndola más lenta. |
| ***POLIMORFISMO.-*** Es acceder a un variado rango de funciones distintas a través del mismo interfaz, como comportamientos diferentes asociados a objetos distintos que pueden compartir el mismo nombre. | ***FIABILIDAD.-***  Al dividir el programa en partes es mucho más sencillo, realizar pruebas y aislar los posibles errores que puedan surgir. | Los objetos requieren una extensa documentación y al ser abstractos pueden coincidir o no con la visión de un programador y otro. |
| ***PROGRAMACIÓN ORIENTADA A COMPONENTES*** | ***CARACTERÍSTICAS*** | ***VENTAJAS*** | ***DESVENTAJAS*** |
| Es una extensión de la programación orientada a objetos. Permite construir sistemas abiertos y distribuidos. | ***FACILIDAD DE INSTALACIÓN.-*** Permite reemplazar sin perjudicar la funcionalidad de cualquier otro componente. | Clarividencia |
| Está basada en la noción de componente, entendiendo a componente como la unidad de composición de aplicaciones software que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio. | ***COSTOS REDUCIDOS.-*** Se podría considerar la reducción de costos de desarrollo y mantenimiento. | Evolución de los componentes. |
|  | ***FACILIDAD DE DESARROLLO.-*** Al realizarlo por separado no interfiere en el desarrollo de otros componentes. | Percepción del entorno y particularización. |
|  | ***REUSABLE.-***  La reusabilidad es una importante característica de un componente de software de alta calidad. Los programadores deben diseñar e implementar componentes de software de una manera tal que diversos programas puedan reutilizarlos.  . | Falta de soporte formal. |
|  |  | Interoperabilidad. |