11 GUIA Resuelta Programacion Orientada a Objetos -Cardacci 2016

Programación Orientada A Objetos (Universidad Abierta Interamericana)

Universidad Abierta Interamericana

PROGRAMACIÓN ORIENTADA A OBJETOS

2016

FACULTAD DE TECNOLOGÍA INFORMÁTICA

UNIVERSIDAD ABIERTA INTERAMERICANA

Universidad Abierta Interamericana

GUÍA DE TRABAJOS REVISIÓN CONCEPTUAL

UNIDAD I

1. Enumere y explique los aspectos más relevantes que hacen que un software de gran magnitud sea complejo.
2. ¿Cuáles son los cinco atributos de un sistema complejo?
3. ¿Cuáles son las dos jerarquías más importantes que consideramos en la orientación a objetos para sistemas complejos?
4. ¿Con qué podemos enfrentar a la complejidad para obtener partes cada vez más pequeñas y simplificadas del dominio del problema?
5. ¿Cuáles son las dos formas de descomposición más conocidas?
6. ¿Explique en qué se diferencia la descomposición algorítmica y la orientada a objetos?
7. ¿Qué rol cumple la abstracción en la orientación a objetos?
8. ¿Qué rol cumple la jerarquía en la orientación a objetos?
9. ¿Consideraría Ud. al diseño orientado a objetos un desarrollo evolutivo o revolucionario? Justifique.
10. ¿Cuántos y cuáles son los modelos básicos que se manejan en el diseño orientado a objetos?
11. ¿Qué es la programación orientada a objetos?
12. ¿Qué es el diseño orientado a objetos?
13. ¿Qué es el análisis orientado a objetos?
14. ¿Cuáles son los elementos fundamentales en el modelo de objetos?
15. ¿Cuáles son los elementos secundarios del modelo de objetos?
16. Explique el significado de la abstracción.
17. Explique el significado del encapsulamiento.
18. Explique el significado de la modularidad.
19. Explique el significado de la jerarquía.

20. Explique el significado de la tipificación.

21. Explique el significado de la concurrencia.

22. Explique el significado de la persistencia.

23. ¿Cómo se denotan las características esenciales de un objeto que lo distinguen de todos los demás tipos de objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador?

24. ¿A qué denominamos un objeto cliente?

25. ¿A qué denominamos un objeto servidor?

26. ¿A qué denomina Meyer el modelo contractual de programación?

27. ¿Qué establece un contrato entre objetos?

28. ¿Cómo se denomina a las formas en que un objeto puede actuar y/o reaccionar, constituyendo estas formas la visión externa completa, estática y dinámica de la abstracción?

29. ¿Cómo se denomina al conjunto completo de operaciones que puede realizar un cliente sobre un objeto, junto con las formas de invocación u órdenes que admite?

30. ¿A qué nos referimos cuando decimos que un concepto central de la idea de abstracción es el de invariancia?

31. ¿Qué se debe definir para cualquier operación asociada a un objeto?

32. ¿Qué es una precondición?

33. ¿Qué es una postcondición?

34. ¿A qué se denomina excepción?

35. ¿A qué se denomina mensaje?

36. ¿El encapsulado es un concepto complementario a la abstracción? Justifique.

37. ¿Cómo se denomina al elemento de un objeto que captura su vista externa?

38. ¿Cómo se denomina al elemento de un objeto que captura su vista interna la cual incluye los mecanismos que consiguen el comportamiento deseado?

39. ¿El concepto de “ocultar los detalles de implementación” lo asociaría a “esconder los detalles de implementación” o a “evitar el uso inapropiado de los detalles de implementación”? Justifique.

40. ¿Cuáles son los dos aspectos que hacen importante considerar a la modularidad?
41. ¿Para qué se utiliza la jerarquía?
42. ¿Cómo denominamos a la caracterización precisa de propiedades estructurales o de comportamiento que comparten una serie de entidades?
43. ¿Las clases implementan tipos?
44. ¿Los tipos implementan clases?
45. ¿Cómo denominamos a los lenguajes que hacen una comprobación de tipos estricta?
46. ¿Cómo denominamos a los lenguajes que no hacen una comprobación de tipos estricta?
47. ¿A qué se denomina ligadura estática (temprana)?
48. ¿A qué se denomina ligadura dinámica (tardía)?
49. ¿Es lo mismo la comprobación estricta de tipos y la ligadura dinámica?
50. ¿Cómo se denomina la característica que permite a diferentes objetos actuar al mismo tiempo?
51. ¿A qué se denomina concurrencia pesada?
52. ¿A qué se denomina concurrencia ligera o liviana?
53. ¿La concurrencia es la propiedad que distingue un objeto activo de uno que no lo está?
54. ¿Cómo se denomina la característica en orientación a objetos que permite conservar el estado de un objeto en el tiempo y el espacio?
55. ¿Qué cosas se persisten?
56. Defina qué es un objeto desde la perspectiva de la cognición humana.
57. ¿Un objeto es real o abstracto? Justifique.
58. ¿Los objetos poseen límites físicos precisos o imprecisos?
59. ¿Cuáles son las tres cosas que debe tener un objeto?
60. ¿Cuál es la palabra que se puede utilizar como sinónimo de objeto?
61. ¿Cuál es la palabra que se puede utilizar como sinónimo de instancia?
62. ¿Cómo definiría el estado de un objeto?

63. ¿A qué definimos propiedad de un objeto?

64. ¿Qué es lo que distingue a “un objeto” de los “valores simples”?

65. ¿Cómo definiría el comportamiento de un objeto?

66. ¿El comportamiento de un objeto se ve afectado por el estado del mismo o bien que el comportamiento del objeto es función de su estado?

67. ¿Algunos comportamientos pueden alterar el estado de un objeto?

68. Se puede afirmar que el estado de un objeto termina siendo los resultados acumulados de su comportamiento.

69. ¿A qué definiría como operación (método/función miembro)?

70. ¿Cuáles son las tres operaciones más comunes?

71. ¿Cuáles son las dos operaciones habituales que se utilizan para crear y destruir instancias de clases?

72. ¿Qué tipo de operación es el modificador?

73. ¿Qué tipo de operación es el selector?

74. ¿Qué tipo de operación es el iterador?

75. ¿Qué tipo de operación es el constructor?

76. ¿Qué tipo de operación es el destructor?

77. ¿Cómo denominamos operaciones fuera de las clases en aquellos programas orientados a objetos que permiten colocarlas (ej. C++)?

78. ¿Todos los métodos son operaciones?

79. ¿Todas las operaciones son métodos?

80. Dado el protocolo de un objeto (todos sus métodos y subprogramas libres asociados al objeto si el lenguaje lo permite) es conveniente dividirlo en grupos lógicos más pequeños de comportamiento?¿Por qué?

81. ¿Cómo denominamos a los grupos lógicos más pequeños de comportamiento del protocolo total de un objeto?

82. ¿Cuáles son las dos responsabilidades más importantes que posee un objeto?

83. ¿Es relevante el orden en que se invocan las operaciones de un objeto?

84. ¿Por qué decimos que los objetos se pueden considerar como máquinas?
85. ¿Qué es la identidad de un objeto?
86. Dadas dos variable X e Y del mismo tipo ¿qué significa que ambas son iguales?
87. Dadas dos variable X e Y del mismo tipo ¿qué significa asignarle Y a X?
88. Dadas dos variable X e Y del mismo tipo ¿qué significa clonar X en Y?
89. ¿Qué significa realizar una clonación superficial?
90. ¿Qué significa realizar una clonación profunda?
91. ¿Qué es el ciclo de vida de un objeto?
92. ¿Cómo se libera el espacio ocupado por un objeto?
93. ¿Qué tipos de relaciones existen entre los objetos?
94. ¿Cómo podemos definir al enlace entre objetos?
95. ¿Cómo pueden ser los mensajes entre dos objetos en una relación de enlace?
96. ¿Qué es un mensaje unidireccional?
97. ¿Qué es un mensaje bidireccional?
98. ¿Quién inicia el paso de un mensaje entre dos objetos en una relación de enlace?
99. ¿Cuáles son los roles o papeles que puede desempeñar un objeto en una relación de enlace?
100. ¿Qué significa que un objeto actúe como “Actor”?
101. ¿Qué significa que un objeto actúe como “Servidor”?
102. ¿Qué significa que un objeto actúe como “Agente”?
103. Dados dos objetos A y B, si A le puede enviar un mensaje a B, estando ambos relacionados por enlace, decimos que B respecto de A está: ……
104. ¿Cuáles son las cuatro formas de visibilidad que puede poseer un objeto servidor respecto de un objeto cliente?
105. En una relación de enlace de dos objetos, cuando uno le pasa un mensaje al otro, además de adoptar roles ambos deben estar:……..

106. ¿Cuáles son las posibles formas de sincronización?
107. ¿Qué significa que dados dos objetos A y B estos están secuencialmente sincronizados?
108. ¿Qué significa que la forma de sincronizarse de un conjunto de objetos es vigilada?
109. ¿Qué significa que la forma de sincronizarse de un conjunto de objetos es síncrona?
110. ¿El enlace es una relación de igual a igual o jerárquica?
111. ¿La agregación es una relación de igual a igual o jerárquica?
112. ¿Qué tipo de jerarquía denota la agregación?
113. ¿Qué otros nombres recibe el “todo” en una relación de agregación?
114. ¿En una relación de agregación las “partes” forman parte del estado del “todo”?
115. ¿Qué tipos de agregación existen?
116. ¿Qué caracteriza a la agregación con contención física?
117. ¿Qué es una clase?
118. ¿La interfaz de la clase proporciona su visión interna?
119. ¿La implementación de la clase proporciona su visión externa?
120. ¿En cuántas partes la podemos dividir una interfaz en términos de la accesibilidad o visibilidad que posee?
121. ¿Qué tipos básicos de relaciones existen entre las clases?
122. ¿Qué relaciones entre clases se desprenden de las tres relaciones básicas?
123. ¿La asociación denota una dependencia semántica y la dirección de esta asociación?
124. ¿Qué significa la cardinalidad en una relación?
125. ¿Qué cardinalidad puede existir entre clases relacionadas por asociación?
126. ¿Qué es la herencia?
127. ¿Cuántos tipos de herencia existen?
128. ¿A qué se denomina herencia simple?
129. ¿A qué se denomina herencia múltiple?

130. ¿Cómo se denomina a la clase que no se espera tener instancias de ella y solo se utilizará para heredar a otras clases?

131. ¿Cómo se denomina a la clase que se espera tener instancias de ella y puede utilizarse para heredar a otras clases o no?

132. ¿Cómo se denomina al método de una clase abstracta que no posee implementación y fue pensado para que sea implementado en las sub clases que lo heredan?

133. ¿Cómo se denomina a la clase más generalizada en una estructura de clases?

134. ¿Qué es el polimorfismo?

135. ¿Cómo se denomina cuando una clase posee métodos que comparten el nombre y se diferencias por su firma?

136. ¿Qué sentencias de código se evitan utilizar cuando se aplica correctamente el polimorfismo?

137. ¿Qué es la agregación cómo relación entre clases?

138. ¿Qué formas de contención física existen en la agregación?

139. ¿Qué características posee la contención física por valor?

140. ¿Qué características posee la contención física por referencia?

141. ¿Qué es una relación de uso?

142. ¿Qué es la instanciación?

143. ¿Todo objeto es una instancia de una clase?

144. ¿Qué es una metaclase?

145. ¿Qué métricas hay que observar para determinar la calidad de una abstracción?

146. ¿Qué es el acoplamiento?

147. ¿Qué es la cohesión?

148. ¿Qué es la suficiencia?

149. ¿Qué es la compleción?

150. ¿Qué significa ser primitivo?

151. ¿Qué se debe observar al momento de decidir si una abstracción debe implementar un determinado comportamiento o no?

152. ¿Qué formas puede adoptar el paso de un mensaje?

153. ¿Qué características posee un mensaje síncrono?

154. ¿Qué características posee un mensaje de abandono inmediato?

155. ¿Qué características posee un mensaje de intervalo?

156. ¿Qué características posee un mensaje Asíncrono?

157. ¿Qué significa que una abstracción está accesible a otra?

158. ¿Qué expresa la ley de Demeter?

159. ¿Cuál es la consecuencia inmediata de aplicar la ley de Demeter?

160. ¿Cuáles son las cuatro formas fundamentales por las cuales un objeto X puede hacerse visible a un objeto Y?

161. ¿Para qué sirve clasificar a los objetos?

162. ¿Por qué es tan difícil la clasificación de objetos?

163. ¿Cómo es el rol del observador en la clasificación de objetos?

164. ¿Cuáles son las aproximaciones generales a la clasificación?

165. ¿Qué es la categorización clásica?

166. ¿Qué es el agrupamiento conceptual?

167. ¿Qué es la teoría de prototipos?

168. ¿Qué es una abstracción clave?

169. ¿Qué son los mecanismos?

UNIDAD II

170. ¿Qué es un Framework?
171. ¿Qué son los frozen-spots en un Framework?
172. ¿Qué son los hot-spots en un Framework?
173. ¿Cómo se puede clasificar un Framework según su extensibilidad?
174. ¿Qué es un Framework de Caja Blanca?
175. ¿Qué es un Framework de Caja Negra?
176. ¿Qué ventajas posee utilizar un Framework?
177. ¿Qué problemas resuelve .NET Framework?
178. ¿Qué es y qué permite hacer el CLR?
179. ¿Qué es el MSIL?
180. ¿Qué es el CTS?
181. ¿Qué es el CLS?

UNIDAD III

182. ¿Qué es un Campo de una clase?

183. ¿Qué es un método de una clase?

184. ¿A qué denominamos sobrecarga?

185. ¿Qué es una propiedad de una clase?

186. ¿Qué tipos de propiedades existen?

187. ¿Qué ámbitos pueden tener los campos, métodos y propiedades de las clases?

188. ¿Qué características posee cada ámbito existente si se lo aplico a un campo, un método y una propiedad de una clase?

189. ¿Para qué se utilizan los constructores?

190. ¿A qué se denomina tiempo de vida de un objeto?

191. ¿Para administrar las instancias de .NET se utiliza un contador de referencias?

192. ¿Qué objeto es el encargado de liberar el espacio ocupado por objetos que ya no se utilizan?

193. ¿Dónde se encuentran las instancias de los objetos administrados por el GC?

194. ¿Cuáles son los dos métodos más notorios que deben implementar las clases para trabajar correctamente con la recolección de elementos no utilizados y matar las instancias administradas y no administradas?

195. ¿De dónde heredan las clases el método Finalize?

196. ¿Cuál es la firma que implementa el método “Finalize”?

197. ¿Qué método se utiliza para que el GC recolecte los elementos no utilizados?

198. ¿Qué método se utiliza para suspender el subproceso actual hasta que el subproceso que se está procesando en la cola de finalizadores vacíe dicha cola?

199. ¿Cuándo se ejecuta el método collect del GC que método se ejecuta en los objetos afectados?

200. ¿Qué método debería exponer una clase bien diseñada teniendo en consideración que no posee destructor?

201. ¿Cómo obtengo el método “Dispose”?

202. ¿Qué se programa en el método “Dispose”?

203. ¿Se pueden combinar el uso de “Dispose” y “Finalize”?

204. ¿A qué se denomina “Resurrección de Objetos”?

205. ¿A qué se denomina “Generación” en el contexto de la recolección de elementos no utilizados?

206. ¿Qué valores puede adoptar la “Generación” de un objeto?

207. ¿Cómo se puede obtener el número de veces que se ha producido la recolección de elementos no utilizados para la generación de objetos especificada?

208. ¿Cómo se obtiene el número de generación actual de un objeto?

209. ¿Cómo se puede recuperar el número de bytes que se considera que están asignados en la actualidad?

210. ¿Qué utiliza para convertir un objeto en “no” válido para la recolección de elementos no utilizados desde el principio de la rutina actual hasta el momento en que se llamó a este método?

211. ¿Cómo se solicita que el sistema no llame al finalizador del objeto especificado?

212. ¿Cómo se solicita que el sistema llame al finalizador del objeto especificado, para el que previamente se ha llamado a “SuppressFinalize”?

213. ¿Cómo se obtiene el número máximo de generaciones que el sistema admite en la actualidad?

214. ¿Qué son los sucesos?

215. ¿Qué se utiliza para declarar un suceso?

216. ¿Cómo se logra que ocurra un suceso?

217. ¿Cómo se pueden atrapar los sucesos?

218. ¿Para qué se utiliza Addhandler en un suceso?

219. ¿Cómo y qué cosas se pueden compartir en una clase?

220. ¿Qué características poseen los campos compartidos?

221. ¿Qué características poseen los métodos compartidos?

222. ¿Qué características poseen los sucesos compartidos?
223. ¿Se pueden atrapar sucesos desde matrices? ¿Cómo?
224. ¿Qué son los miembros compartidos?
225. ¿Qué son y para que se pueden utilizar las clases anidadas?
226. ¿Qué ámbitos existen?
227. ¿Qué característica posee el ámbito público?
228. ¿Qué característica posee el ámbito privado?
229. ¿Qué característica posee el ámbito friend?
230. ¿Qué característica posee el ámbito protected?
231. ¿Qué característica posee el ámbitop protected friend?
232. ¿Qué cosas se pueden heredar?
233. ¿Cómo y para qué se puede aprovechar en la práctica el polimorfismo?
234. ¿Cómo y para qué se utiliza la clase derived?
235. ¿Qué representa la clase me?
236. ¿Qué clase representa a la clase base?
237. ¿Para qué utilizaría MyBase?
238. ¿Qué representa la clase MyClass?
239. ¿Qué diferencia existe entre MyBase y MyClass?
240. ¿Para qué se usa una clase abstracta?
241. ¿Para qué se usa una clase sellada?
242. ¿Qué es la sobreescritura?
243. ¿Qué elementos se pueden sobreescribir?
244. ¿A qué se denomina sombreado de métodos?
245. ¿Qué característica peculiar posee el sombrado vs la sobre-escritura?

UNIDAD IV

246. ¿Qué es una excepción?

247. ¿Qué se coloca en el bloque “Catch”?

248. ¿Cómo construiría un objeto del tipo “Exception” personalizado?

249. ¿Qué ocurre si en el bloque de código donde se produce la excepción el error no está siendo tratado?

250. ¿Cuál es el objeto de mayor jerarquía para el manejo de excepciones?

251. ¿En qué namespace se encuentra la clase Exception?

252. ¿Cuáles son las dos clases genéricas más importantes definidas en el FrameWork además de Exception?

253. ¿Qué instrucción se utiliza para poner en práctica el control e interceptar las excepciones?

254. ¿Dónde se coloca el código protegido contra excepciones si se iniciara una excepción?

255. ¿Qué tipo de excepción se utiliza para interceptar un error de división por cero?

256. ¿Qué tipo de excepción se utiliza para interceptar una DLL que tiene problemas al ser cargada?

257. ¿Qué colocaría dentro de una clausula “Catch” para especificar una condición adicional que el bloque “Catch” deberá evaluar como verdadero para que sea seleccionada?

258. ¿Si se desea colocar código de limpieza y liberación de recursos para que se ejecute cuando una excepción se produzca, dónde lo colocaría?

259. ¿Qué instrucción se utiliza para provocar un error y que el mismo se adapte al mecanismo de control de excepciones?

260. ¿Escriba el código que permitiría provocar una excepción del tipo “ArgumentException”?

261. ¿Cómo construiría un objeto del tipo “Exception” personalizado?

262. ¿Cómo armaría un “Catch” personalizado para que se ejecute cuando se de la excepción “ClienteNoExisteException”?

UNIDAD V

263. ¿Para qué se utilizan las interfaces?
264. ¿Cómo se implementa una interfaz?
265. ¿Se pueden heredar las interfaces?
266. ¿Se puede implementar un tipo de polimorfismo peculiar por medio de interfaces?
267. ¿Para qué se utiliza la interfaz IComparable?
268. ¿Para qué se utiliza la interfaz IComparer?
269. ¿Para qué se utiliza la interfaz ICloneable?
270. ¿Para qué se utiliza la interfaz IEnumerable?
271. ¿Para qué se utiliza la interfaz IEnumerator?
272. ¿Qué es un delegado?
273. ¿A qué elementos se les puede delegar?
274. ¿Qué se puede delegar?
275. ¿Cómo construiría un delegado?
276. ¿Cómo implementaría un procedimiento de devolución de llamadas?
277. ¿Para qué sirve la multidifusión de delegados?
278. ¿Qué es un atributo?
279. ¿Cómo es la sintaxis de un atributo?
280. ¿Para qué se utiliza el atributo StructLayout?
281. ¿Para qué se utiliza el atributo FieldOffset?
282. ¿Para qué se utiliza el atributo Conditional?
283. ¿Para qué se utiliza el atributo Obsolete?
284. ¿Para qué se utiliza el atributo DebuggerStepThrough?

UNIDAD VI

285. ¿Qué características posee un esquema cliente - servidor?
286. ¿Qué significa pasar información batch?
287. ¿Qué significa pasar información on line?
288. ¿Qué es un protocolo?
289. ¿Qué protocolo usa una red de área local?
290. ¿Qué protocolo usa internet?
291. ¿Qué hace el protocolo IP?
292. ¿Qué ventajas tiene distribuir procesos?
293. ¿Qué ventajas tiene distribuir almacenamientos?
294. ¿Qué es un socket?
295. ¿Qué característica posee un socket sincrónico?
296. ¿Qué característica posee un socket asincrónico?
297. ¿Qué objeto se puede utilizar para construir un navegador?
298. ¿Para qué se utilizan los puertos de la pc?
299. ¿Qué puertos posee una PC?
300. ¿Cómo funciona el puerto paralelo?
301. ¿Cómo funciona el puerto serie?
302. ¿Cómo funciona el puerto USB?
303. ¿Qué es la domótica?

GUÍA DE TRABAJOS PRÁCTICOS

UNIDAD I

1. Desarrollar una jerarquía de clases relacionadas por herencia y agregación que represente la estructura necesaria para identificar claramente los elementos encontrados en un sistema de calificaciones de una universidad. En cada clase detallar métodos y propiedades.

2. Defina una situación donde se desee llegar desde un estado inicia a uno final, por ejemplo, una cuenta bancaria que posee un saldo inicial y se la somete a una operación bancaria. Desarrollar la clase cuenta con las propiedades y métodos que considere pertinentes y demuestre como cambia el estado del objeto en la medida que se le realizan depósitos, extracciones y transferencias.

3. Genere una estructura de clases donde se pueda observar la herencia simple y se justifique utilizar el polimorfismo.

4. Genere una estructura de clases donde se pueda observar la herencia múltiple y se justifique utilizar el polimorfismo.

5. Desarrollar conjuntos de objetos reales donde para su agrupación y clasificación sea necesario aplicar la categorización conceptual, la clásica y el agrupamiento prototípico. Justifique.

UNIDAD II

6. Desarrollar un programa que genera varias instancias (una cantidad importante), verifique la memoria utilizada, pase el GC y vuelva a verificar el espacio de memoria. ¿Qué se observa?

7. Desarrollar un programa que genere una instancia, pierda la referencia a la misma y aplicando la técnica de “resurrección de objetos” logre obtener la referencia a ese mismo objeto.

UNIDAD III

1. Desarrollar un programa que posea una estructura de clases donde se puedan observar dos métodos y el constructor sobrecargados.

2. Desarrollar un programa que posea una estructura de clases donde se pueda observar una propiedad de solo lectura, una propiedad de solo escritura, una propiedad de escritura-lectura, una propiedad de predeterminada y una propiedad con argumentos.

3. Desarrollar un programa que posea una clase que aplique el Finalize y el Dispose, explique para qué utilizó cada uno y qué debió suceder para que funcione.

4. Desarrollar un programa que posea una clase que posea propiedades, métodos y sucesos. Los sucesos deben ser atrapados por el método tradicional (withevents) y por delegados (AddHandler).

5. Desarrollar un programa que posea una clase que contenga al menos dos campos compartidos, tres métodos compartidos, un constructor compartido y dos sucesos compartidos.

6. Desarrollar un programa donde se observe claramente el uso del MyBase. Enfatice las particularidades al usarlo en los constructores y finalizadores. Demuestre en el mismo programa el uso de me. Establezca las diferencias y en qué caso se justifica utilizar cada uno.

7. Desarrollar un programa que posea una clase que implemente dos propiedades. La primera de ellas deberá permitir calcular la edad de una persona y la segunda retornar un valor boolean que sea verdadero si la persona puede votar. Una persona puede votar cuando posee 18 o más años. La propiedad que determina calcula la edad es sobre-escribible. Una segunda clase hereda de esta y sobre-escribe la propiedad colocando una implementación errónea (p.e. un cálculo erróneo de la edad). En la implementación de la propiedad booleana utilice primero MyBase y observe que ocurre, tome nota, luego haga lo mismo con MyClass, tome nota. Finalmente compare los resultados.

8. Desarrollar un programa que posea al menos una clase abstracta, un método virtual, una clase sellada y una clase anidada.

UNIDAD IV

9. Desarrollar un programa que aplique el concepto de manejo de errores. La estructura propuesta debe tener al menos 5 Catch y el finally.

10. Desarrollar un programa que aplique el concepto de manejo de errores. Generar un error personalizado por medio de una clase que herede de Exception y disparar el error con Throw.

UNIDAD V

11. Desarrollar un programa que implemente la interfaz IComparable. Genere un arrary de objetos y ordénelos.

12. Desarrollar un programa que implemente la interfaz IComparer. La clase que lo implementa deberá tener al menos cinco criterios de ordenamiento. Genere un arrary de objetos y ordénelos por cada criterio.

13. Desarrollar un programa que implemente la interfaz ICloneable. Clone el objeto que posee la interfaz y demuestre que la clonación funcionó.

14. Desarrollar un programa que implemente las interfaces IEnumerable e IEnumerator. Lo adaptado recórralo con el for each y muestre los resultados.

15. Desarrollar un programa que posea una clase por cada atributo visto en clase. Cada clase deberá implementar un atributo. Demostrar que el atributo funciona.

UNIDAD VI

16. Desarrollar un programa que utilizando socket permita escribir en una aplicación que se esté ejecutando en una PC de la red y lo escrito se pueda visualizar en la aplicación que está corriendo en otra PC de la red.

17. Desarrollar un programa que utilizando socket que emule un servidor de chat. También desarrollar el software cliente que se ejecutará en las Pc's que se conecten al servidor. Cada mensaje enviado por un cliente lo podrán recibir todos los demás. En el caso que un cliente solicite comunicación privada el mensaje se verá solo por los que participan de esta comunicación. Un cliente puede solicitar armar un grupo y el resto unirse a él, para que esto suceda el propietario del grupo deberá aceptarlos y tiene la potestad de desconectarlos. El servidor es quien modera todo y por allí pasan todos los mensajes.

18. Desarrollar un programa que haciendo uso del puerto paralelo y un emulador que permita enviar señales al mismo y recolectar las entradas que se producen en él.

RESPUESTAS

UNIDAD I

1. Enumere y explique los aspectos más relevantes que hacen que un software de gran magnitud sea complejo.

1-Un software es complejo por varias razones:

-La complejidad del dominio del problema; por que los usuarios no explican con precisión lo que desean. Hay muchos requisitos y van cambiando.

-La dificultad de gestionar el proceso de desarrollo; porque en esta época, hay millones de líneas de código y se quiere comprender, se necesita un gran número de desarrolladores y a la vez, mientras más sean, es más compleja la comunicación.

-La flexibilidad que se puede alcanzar a través del SW; porque no hay límites, se puede hacer de todo, no así en 1 empresa de construcción, que por ejemplo no tiene gracia construir una fábrica de madera en el lugar donde se va a hacer la construcción.

-Los problemas que plantea la categorización del comportamiento de sistemas discretos; hay muchas variantes de sistemas discretos, hay muchas variables, los eventos externos pueden afectar a cualquier parte del estado interno.

2. ¿Cuáles son los cinco atributos de un sistema complejo?

-Atributos de un sistema complejo:

A) Frecuentemente, la complejidad toma la forma de una jerarquía, por lo cual un sistema complejo se compone de subsistemas, y así sucesivamente, hasta que se alcanza algún nivel ínfimo de componentes elementales.

B) La elección de qué componentes de un sistema son primitivos es relativamente arbitraria y queda en gran medida a decisión del observador.

C) Los enlaces internos de los componentes suelen ser más fuertes que los enlaces entre componentes. Este hecho tiene el efecto de separar la dinámica de la estructura interna de los componentes de la dinámica que involucra la interacción entre los componentes.

D) Los sistemas jerárquicos están compuestos usualmente de solo unas pocas clases diferentes de subsistemas en varias combinaciones y disposiciones.

E) Se encontrará invariablemente que un sistema complejo que funciona ha evolucionado de un sistema simple que funcionaba. Un sistema complejo diseñado desde cero nunca funciona y no puede parchearse para conseguir que lo haga. Hay que volver a empezar, partiendo de un sistema simple que funcione.

3. ¿Cuáles son las dos jerarquías más importantes que consideramos en la orientación a objetos para sistemas complejos?

Jerarquías:

- "Parte - de" (part of) Estructura de clases

- "Es - un" (is a) Estructura de objetos

4. ¿Con qué podemos enfrentar a la complejidad para obtener partes cada vez más pequeñas y simplificadas del dominio del problema?

Para enfrentar la complejidad podemos usar la Descomposición.

5. ¿Cuáles son las dos formas de descomposición más conocidas?

Las dos formas de descomposición son la Algorítmica y la Orientada a objetos.

6. ¿Explique en qué se diferencia la descomposición algorítmica y la orientada a objetos?

Algorítmica: cada modulo del sistema representa a un paso importante de algún proceso global.

Orientada a objetos: se descompone el sistema desde el punto de vista de objetos y clases con una notación para el modelo lógico físico estático y dinámico.

7. ¿Qué rol cumple la abstracción en la orientación a objetos?

Una abstracción denota las características esenciales de un objeto que lo distinguen de todos los demás tipos objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador.

8. ¿Qué rol cumple la jerarquía en la orientación a objetos?

La jerarquía es una clasificación u ordenación de abstracciones.

9. ¿Consideraría Ud. al diseño orientado a objetos un desarrollo evolutivo o revolucionario? Justifique.

Es un desarrollo revolucionario porque el sistema va evolucionando desde un sistema simple hasta un sistema complejo donde posee partes q han sido reutilizadas y modificadas para el mismo, descomponiéndolo en módulos que iran evolucionando en el tiempo.

10. ¿Cuántos y cuáles son los modelos básicos que se manejan en el diseño orientado a objetos?

Son 4 los modelos básicos:

-Modelo estático.

-Modelo dinámico.

-Modelo lógico.

-Modelo físico.

11. ¿Qué es la programación orientada a objetos?

Es un método de implementación en que los programas se organizan como colecciones cooperativa de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia.

12. ¿Qué es el diseño orientado a objetos?

Es un método de diseño que abarca el proceso de descomposición orientada a objetos y una notación para describir los modelos lógico y físico, así como los modelos estático y dinámico.

13. ¿Qué es el análisis orientado a objetos?

Es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema.

14. ¿Cuáles son los elementos fundamentales en el modelo de objetos?

Abstracción.

Encapsulamiento.

Modularidad.

Jerarquía.

15. ¿Cuáles son los elementos secundarios del modelo de objetos?

Tipos.

Concurrencia.

Persistencia.

16. Explique el significado de la abstracción.

Una abstracción denota las características esenciales de un objeto que lo distinguen de todos los demás tipos objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador.

17. Explique el significado del encapsulamiento.

Oculta los detalles de implementación de un objeto. Es el proceso de almacenar en un compartimiento los elementos de una abstracción que constituyen su estructura y su comportamiento; sirve para separar la interfaz contractual de una abstracción y su implantación.

18. Explique el significado de la modularidad.

La modularidad es la propiedad que posee un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados.

19. Explique el significado de la jerarquía.

La jerarquía es una clasificación u ordenación de abstracciones.

20. Explique el significado de la tipificación.

Es la puesta en vigor de la clase de objetos, de modo que los objetos distintos no pueden intercambiarse o, como mucho, pueden pero solo de formas muy restringidas.

21. Explique el significado de la concurrencia.

La concurrencia permite a diferentes objetos actuar al mismo tiempo. La concurrencia es la propiedad que distingue un objeto activo de uno que no está activo.

22. Explique el significado de la persistencia.

La concurrencia permite a diferentes objetos actuar al mismo tiempo. La concurrencia es la propiedad que distingue un objeto activo de uno que no está activo. La persistencia es la propiedad de un objeto por la que su existencia trasciende el tiempo (es decir, el objeto continúa existiendo después de que su creador deja de existir) y/o el espacio (es decir, la posición del objeto varía con respecto a espacio de direcciones en el que fue creado).

23. ¿Cómo se denotan las características esenciales de un objeto que lo distinguen de todos los demás tipos de objetos y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador?

A través de la Abstracción, concentrándose en la visión externa de un objeto.

24. ¿A qué denominamos un objeto cliente?

Un objeto puede operar sobre otros objetos pero otros objetos no pueden operar sobre él. (Operar=mandar mensajes)

25. ¿A qué denominamos un objeto servidor?

Un objeto que nunca va a operar sobre los demás, da 1 servicio.

26. ¿A qué denomina Meyer el modelo contractual de programación?

La vista externa de cada objeto define un contrato del que pueden depender otros objetos, y que a su vez debe ser llevado a cabo por la vista interior del propio objeto.

27. ¿Qué establece un contrato entre objetos?

Establece las responsabilidades de un objeto.

28. ¿Cómo se denomina a las formas en que un objeto puede actuar y/o reaccionar, constituyendo estas formas la visión externa completa, estática y dinámica de la abstracción?

Protocolo

29. ¿Cómo se denomina al conjunto completo de operaciones que puede realizar un cliente sobre un objeto, junto con las formas de invocación u órdenes que admite?

Protocolo.

30. ¿A qué nos referimos cuando decimos que un concepto central de la idea de abstracción es el de invariancia?

Porque ese valor de verdad debe mantenerse y si se rompe esa invariancia, se rompe el contrato asociado con 1 abstracción.

31. ¿Qué se debe definir para cualquier operación asociada a un objeto?

Se deben definir pre y post condiciones.

32. ¿Qué es una precondición?

Invariantes asumidos por la operación. (Invariante: es una condición booleana (V o F))

33. ¿Qué es una postcondición?

Invariantes satisfechos por la operación.

34. ¿A qué se denomina excepción?

Es una indicación de que no se ha satisfecho algún invariante.

35. ¿A qué se denomina mensaje?

Es lo que le envía un objeto a otro para que ese otro objeto haga una alguna operación.

36. ¿El encapsulado es un concepto complementario a la abstracción? Justifique.

Si, por que el encapsulado se centra en la implementación que da a lugar al comportamiento del objeto.

37. ¿Cómo se denomina al elemento de un objeto que captura su vista externa?

Interfaz.

38. ¿Cómo se denomina al elemento de un objeto que captura su vista interna la cual incluye los mecanismos que consiguen el comportamiento deseado?

Implementación.

39. ¿El concepto de "ocultar los detalles de implementación" lo asociaría a "esconder los detalles de implementación" o a "evitar el uso inapropiado de los detalles de implementación"? Justifique.

Evitar el uso inapropiado de los detalles de implementación porque si no se ocultaría se podrían obtener problemas.

40. ¿Cuáles son los dos aspectos que hacen importante considerar a la modularidad?

Crear fronteras bien definidas (interfaces) y reducir la complejidad.

41. ¿Para qué se utiliza la jerarquía?

Para ordenar o clasificar las abstracciones.

42. ¿Cómo denominamos a la caracterización precisa de propiedades estructurales o de comportamiento que comparten una serie de entidades?

Tipo

43. ¿Las clases implementan tipos?

Si

44. ¿Los tipos implementan clases?

No

45. ¿Cómo denominamos a los lenguajes que hacen una comprobación de tipos estricta?

Lenguajes tipados.

46. ¿Cómo denominamos a los lenguajes que no hacen una comprobación de tipos estricta?

Débilmente tipados.

47. ¿A qué se denomina ligadura estática (temprana)?

Ligadura denota la asociación de un nombre (como 1 declaración de variable) con 1 clase; lig. est. es 1 ligadura en la que la asociación nombre/clase se realiza cuando se declara el nombre (en tiempo de compilación) pero antes de la creación del objeto que designe el nombre.

48. ¿A qué se denomina ligadura dinámica (tardía)?

Ligadura denota la asociación de un nombre (como 1 declaración de variable) con 1 clase; lig. din. es 1 ligadura en la que la asociación nombre/clase no se realiza hasta que el objeto designado por el nombre se crea en tiempo de ejecución.

49. ¿Es lo mismo la comprobación estricta de tipos y la ligadura dinámica?

No. Estática o temprana. Dinâmica o tardia.

50. ¿Cómo se denomina la característica que permite a diferentes objetos actuar al mismo tiempo?

Concurrencia

51. ¿A qué se denomina concurrencia pesada?

Es el típicamente manejado de forma independiente por el sistema operativo de destino y abarca su propio espacio de direcciones. Es un programa más, un proceso más.

52. ¿A qué se denomina concurrencia ligera o liviana?

Es el que suele existir dentro de un solo proceso del SO en compañía de otro procesos ligeros, que comparten el mismo espacio de direcciones.

53. ¿La concurrencia es la propiedad que distingue un objeto activo de uno que no lo está?

Si.

54. ¿Cómo se denomina la característica en orientación a objetos que permite conservar el estado de un objeto en el tiempo y el espacio?

Persistencia

55. ¿Qué cosas se persisten?

Estados del objeto y clases.

56. Defina qué es un objeto desde la perspectiva de la cognición humana.

Una cosa tangible y/o visible. Algo que puede comprenderse intelectualmente. Algo hacia lo que se dirige un pensamiento o acción.

57. ¿Un objeto es real o abstracto? Justifique.

Es cualquier cosa real o abstracta que posee una estructura que lo define y acciones que lo controlan.

58. ¿Los objetos poseen límites físicos precisos o imprecisos?

Puede poseer las dos cosas.

59. ¿Cuáles son las tres cosas que debe tener un objeto?

Estado, comportamiento e identidad.

60. ¿Cuál es la palabra que se puede utilizar como sinónimo de objeto?

Instancia

61. ¿Cuál es la palabra que se puede utilizar como sinónimo de instancia?

Objeto

62. ¿Cómo definiría el estado de un objeto?

Es el conjunto de todas las propiedades estáticas y los valores dinámicos que adoptan en un momento dado.

63. ¿A qué definimos propiedad de un objeto?

A los valores que este posee, tanto los estáticos como los dinámicos. Es un rasgo que contribuye a identificarlo de otros

64. ¿Qué es lo que distingue a "un objeto" de los "valores simples"?

Un valor simple puede ser un número, como por ej 3, en cambio un objeto puede ser cualquier cosa.

65. ¿Cómo definiría el comportamiento de un objeto?

Es todo aquello que el objeto puede hacer.

66. ¿El comportamiento de un objeto se ve afectado por el estado del mismo o bien que el comportamiento del objeto es función de su estado?

El comportamiento de un objeto se ve afectado por el estado del mismo.

67. ¿Algunos comportamientos pueden alterar el estado de un objeto?

Sí.

68. Se puede afirmar que el estado de un objeto termina siendo los resultados acumulados de su comportamiento.

Si

69. ¿A qué definiría como operación (método/función miembro)?

Algún trabajo que un objeto realiza sobre otro con el fin de provocar una reacción. Es una funcionalidad que la clase expone para que otras la puedan usar.

70. ¿Cuáles son las tres operaciones más comunes?

Modificador, Selector e Iterador.

71. ¿Cuáles son las dos operaciones habituales que se utilizan para crear y destruir instancias de clases?

Constructor y Destructor.

72. ¿Qué tipo de operación es el modificador?

Una operación que altera el estado de un objeto.

73. ¿Qué tipo de operación es el selector?

Una operación que accede al estado de un objeto, pero no altera ese estado.

74. ¿Qué tipo de operación es el iterador?

Una operación que permite acceder a todas las partes de un objeto en algún orden perfectamente establecido.

75. ¿Qué tipo de operación es el constructor?

Una operación que crea un objeto y/o inicializa (o instancia) su estado.

76. ¿Qué tipo de operación es el destructor?

Una operación que libera el estado de un objeto y/o destruye el propio objeto.

77. ¿Cómo denominamos operaciones fuera de las clases en aquellos programas orientados a objetos que permiten colocarlas (Ej. C++)?

Subprogramas libres.

78. ¿Todos los métodos son operaciones?

Si

79. ¿Todas las operaciones son métodos?

No porque pueden ser subprogramas libres.

80. Dado el protocolo de un objeto (todos sus métodos y subprogramas libres asociados al objeto si el lenguaje lo permite) es conveniente dividirlo en grupos lógicos más pequeños de comportamiento?¿Por qué?

Sí, porque así se dividen en papeles y se definen los contratos entre las abstracciones y sus clientes.

81. ¿Cómo denominamos a los grupos lógicos más pequeños de comportamiento del protocolo total de un objeto?

Papeles o roles.

82. ¿Cuáles son las dos responsabilidades más importantes que posee un objeto?

El conocimiento que un objeto mantiene y las acciones que pueden llevar a cabo.

83. ¿Es relevante el orden en que se invocan las operaciones de un objeto?

Si por qué puede ocurrir que la maq exp busque indefinidamente una lata de la cual ya no hay stock, antes de buscarla se tiene que fijar si hay o no stock.

84. ¿Por qué decimos que los objetos se pueden considerar como máquinas?

Porque para algunos objetos es tan importante el orden de las operaciones que se pueden caracterizar mejor formalmente el comportamiento de los objetos en términos de una máquina de estados finitos equivalentes.

85. ¿Qué es la identidad de un objeto?

Es aquella propiedad o conjunto de propiedades que lo distingue de otros objetos.

86. Dadas dos variable X e Y del mismo tipo ¿qué significa que ambas son iguales?

2 variables que apunten al mismo objeto. 2 variables que apunten a 2 instancias pero con el mismo estado.

87. Dadas dos variable X e Y del mismo tipo ¿qué significa asignarle Y a X?

Y apunta a 1 instancia en memoria o nada y X apunta al mismo lugar.

88. Dadas dos variable X e Y del mismo tipo ¿qué significa clonar X en Y?

(Pueden tener el mismo estado o no) Ambas van a apuntar al mismo objeto en memoria.

89. ¿Qué significa realizar una clonación superficial?

Significa que copia el objeto pero comparte su estado.

90. ¿Qué significa realizar una clonación profunda?

Copia el objeto así como su estado y así recursivamente.

91. ¿Qué es el ciclo de vida de un objeto?

Es desde el momento en que se crea por 1º vez, y consume así un espacio por 1º vez, hasta que ese espacio se recupera.

92. ¿Cómo se libera el espacio ocupado por un objeto?

Cuando se ejecute el Garbage Collector. (o las referencias al objeto se pierdan)

93. ¿Qué tipos de relaciones existen entre los objetos?

Agregación y Enlace.

94. ¿Cómo podemos definir al enlace entre objetos?

Es una conexión física o conceptual entre objetos.

95. ¿Cómo pueden ser los mensajes entre dos objetos en una relación de enlace?

Uni y bi direccionales.

96. ¿Qué es un mensaje unidireccional?

Cuando un objeto activo invoca a 1 objeto servidor o agente (que actúe como pasivo)

97. ¿Qué es un mensaje bidireccional?

Cuando 1 objeto activo invoca al pasivo y el pasivo le puede devolver datos al objeto activo

98. ¿Quién inicia el paso de un mensaje entre dos objetos en una relación de enlace?

Lo inicia el cliente u objeto activo

99. ¿Cuáles son los roles o papeles que puede desempeñar un objeto en una relación de enlace?

El de Actor, servidor y agente.

100. ¿Qué significa que un objeto actúe como "Actor"?

Cuando el objeto puede operar sobre otros objetos pero los demás no pueden operar sobre él.

101. ¿Qué significa que un objeto actúe como "Servidor"?

Cuando el objeto nunca opera sobre otros objetos pero los demás si pueden operar sobre él.

102. ¿Qué significa que un objeto actúe como "Agente"?

Cuando el objeto puede operar sobre otros objetos y los demás otros objetos pueden operar sobre él.

103. Dados dos objetos A y B, si A le puede enviar un mensaje a B, estando ambos relacionados por enlace, decimos que B respecto de A está: visible.

104. ¿Cuáles son las cuatro formas de visibilidad que puede poseer un objeto servidor respecto de un objeto cliente?

El objeto servidor es global para el cliente.

El objeto servidor es un parámetro de alguna operación del cliente.

El objeto servidor es parte del objeto cliente.

El objeto servidor es un objeto declarado localmente en alguna operación del cliente.

105. En una relación de enlace de dos objetos, cuando uno le pasa un mensaje al otro, además de adoptar roles ambos deben estar: sincronizados.

106. ¿Cuáles son las posibles formas de sincronización?

Secuencial, vigilado y síncrono.

107. ¿Qué significa que dados dos objetos A y B estos están secuencialmente sincronizados?

El funcionamiento del objeto pasivo está garantizado por el accionar de un único objeto activo simultáneamente.

108. ¿Qué significa que la forma de sincronizarse de un conjunto de objetos es vigilada?

El funcionamiento del objeto pasivo está garantizado por la utilización de múltiples hilos de control. Los clientes activos deben colaborar para asegurar la exclusión mutua.

109. ¿Qué significa que la forma de sincronizarse de un conjunto de objetos es síncrona?

El funcionamiento del objeto pasivo está garantizado por la utilización de múltiples hilos de control. El servidor garantiza la exclusión mutua.

110. ¿El enlace es una relación de igual a igual o jerárquica?

De igual a igual.

111. ¿La agregación es una relación de igual a igual o jerárquica?

Jerárquica.

112. ¿Qué tipo de jerarquía denota la agregación?

Todo/parte.

113. ¿Qué otros nombres recibe el "todo" en una relación de agregación?

Agregado (todo) o contenedor.

114. ¿En una relación de agregación las "partes" forman parte del estado del "todo"?

Si.

115. ¿Qué tipos de agregación existen?

Con y sin contención física.

116. ¿Qué caracteriza a la agregación con contención física?

Cuando un objeto no existe sin el otro. Sus ciclos de vida están íntimamente relacionados.

117. ¿Qué es una clase?

Una clase es un conjunto de objetos que comparten una estructura común y un comportamiento común. Es la representación abstracta resultante de observar un conjunto de objetos reales (obj reales: a aquellos que no son resultantes de la instanciación de las clases) que comparten 1 estructura y 1 comportamiento común.

118. ¿La interfaz de la clase proporciona su visión interna?

No.

119. ¿La implementación de la clase proporciona su visión externa?

No.

120. ¿En cuántas partes la podemos dividir una interfaz en términos de la accesibilidad o visibilidad que posee?

En 3: pública, privada y protegida.

121. ¿Qué tipos básicos de relaciones existen entre las clases?

Es-un, Todo-parte y Asociación.

122. ¿Qué relaciones entre clases se desprenden de las tres relaciones básicas?

Herencia, asociación (si no la pongo como básica), agregación, uso, instanciación y metaclase.

123. ¿La asociación denota una dependencia semántica y la dirección de esta asociación?

La asociación es una relación bidireccional. Dada una instancia de cliente podríamos encontrar el objeto que denota sus compras. Posee CARDINALIDAD.

124. ¿Qué significa la cardinalidad en una relación?

Es la multiplicidad con que se relaciona una clase con otra.

125. ¿Qué cardinalidad puede existir entre clases relacionadas por asociación?

UNO A UNO - UNO A MUCHOS - MUCHOS A MUCHOS

126. ¿Qué es la herencia?

Es una relación jerárquica de clases. Capacidad por la cual una clase de orden inferior puede recibir estructura o acciones de una o más clases de orden superior. La subclase posee la capacidad de incorporar parte estructural y acciones propias.

127. ¿Cuántos tipos de herencia existen?

Dos tipos: Simple y múltiple.

128. ¿A qué se denomina herencia simple?

Es cuando 1 o + clases heredan de 1 sola clase.

129. ¿A qué se denomina herencia múltiple?

Es cuando 1 o + clases heredan de más de 1 clase.

130. ¿Cómo se denomina a la clase que no se espera tener instancias de ella y solo se utilizará para heredar a otras clases?

Clase abstracta.

131. ¿Cómo se denomina a la clase que se espera tener instancias de ella y puede utilizarse para heredar a otras clases o no?

Clase concreta.

132. ¿Cómo se denomina al método de una clase abstracta que no posee implementación y fue pensado para que sea implementado en las sub clases que lo heredan?

Método o función virtual.

133. ¿Cómo se denomina a la clase más generalizada en una estructura de clases?

Superclase o clase base.

134. ¿Qué es el polimorfismo?

Capacidad por la cual una acción puede responder de distinta forma de acuerdo a la subclase que la implementa.

135. ¿Cómo se denomina cuando una clase posee métodos que comparten el nombre y se diferencias por su firma?

Sobrecarga.

136. ¿Qué sentencias de código se evitan utilizar cuando se aplica correctamente el polimorfismo?

Las sentencias de Switch o Case, la cadenas de If...{if...{if...

137. ¿Qué es la agregación cómo relación entre clases?

Relación Jerárquica del tipo Todo - Parte

138. ¿Qué formas de contención física existen en la agregación?

Existe: Con y Sin contención física. Dentro de la Con esta la por valor y por referencia.

139. ¿Qué características posee la contención física por valor?

Sus ciclos de vida están íntimamente relacionados.

140. ¿Qué características posee la contención física por referencia?

Sus ciclos de vida no están íntimamente relacionados. Se pueden crear y destruir instancias de cada clase independientemente.

141. ¿Qué es una relación de uso?

Es una asociación refinada, donde se establece que abstracción es cliente y cual servidor. El cliente hace USO del servidor.

142. ¿Qué es la instanciación?

Acción por la cual se crean instancias de una clase. Los objetos creados corresponden al tipo de la clase que los origina. Es la acciono por la cual sometemos a una clase para obtener un objeto.

143. ¿Todo objeto es una instancia de una clase?

Sí.

144. ¿Qué es una metaclase?

Es la clase de una clase. Es una clase donde las instancias son ella misma.

145. ¿Qué métricas hay que observar para determinar la calidad de una abstracción?

Acoplamiento, cohesión, ser primitivo, complesión y suficiencia.

146. ¿Qué es el acoplamiento?

Medida de la fuerza de la asociación establecida por una conexión entre dos objetos.

147. ¿Qué es la cohesión?

Establece el grado de conectividad interna. Indica que tan específico es un objeto.

148. ¿Qué es la suficiencia?

Indica si se poseen las características necesarias de la abstracción como para permitir una interacción significativa y eficiente.

149. ¿Qué es la compleción?

Indica si la interfaz de la clase captura todas las características de la abstracción.

150. ¿Qué significa ser primitivo?

Denotan operaciones sencillas que se pueden acceder solo por medio de la representación interna básica de la abstracción.

151. ¿Qué se debe observar al momento de decidir si una abstracción debe implementar un determinado comportamiento o no?

4 elementos: La Reutilización, Complejidad, Aplicabilidad y el Conocimiento de la implementación.

152. ¿Qué formas puede adoptar el paso de un mensaje?

Síncrono, asíncrono, de intervalo y de abandono inmediato.

153. ¿Qué características posee un mensaje síncrono?

1 operación comienza solo cuando el emisor ha iniciado la acción y el receptor está preparado para aceptar el mensaje; el emisor y el receptor esperaran indefinidamente hasta que ambas partes estén preparadas para continuar.

154. ¿Qué características posee un mensaje de abandono inmediato?

Igual que el síncrono, excepto en que el emisor abandonara la operación si el receptor no está preparado inmediatamente.

155. ¿Qué características posee un mensaje de intervalo?

Igual que el síncrono, excepto en que el emisor esperara a que el receptor esté listo solo durante un intervalo de tiempo especificado.

156. ¿Qué características posee un mensaje Asíncrono?

Un emisor puede iniciar una acción independientemente de si el receptor está esperando o no el mensaje.

157. ¿Qué significa que una abstracción está accesible a otra?

Es la capacidad de 1 abstracción para ver a otra y hacer referencia a recursos en su vista externa.

158. ¿Qué expresa la ley de Demeter?

Expresa que los métodos de una clase no deberían depender de ninguna manera de la estructura de ninguna clase, salvo de la estructura inmediata, de nivel superior, de su propia clase. Además, cada método debería enviar mensajes solo a objetos pertenecientes a un conjunto muy limitado de clases.

159. ¿Cuál es la consecuencia inmediata de aplicar la ley de Demeter?

Es la creación de clases débilmente acopladas, cuyos secretos de implantación están encapsulados.

160. ¿Cuáles son las cuatro formas fundamentales por las cuales un objeto X puede hacerse visible a un objeto Y?

El objeto proveedor es global al cliente.

El objeto proveedor es parámetro de alguna operación del cliente.

El objeto proveedor es parte del objeto cliente.

El objeto proveedor es un objeto declarado localmente en el ámbito del diagrama de objetos.

161. ¿Para qué sirve clasificar a los objetos?

Para poder agruparlos de manera correcta. Para ordenar el conocimiento.

162. ¿Por qué es tan difícil la clasificación de objetos?

Porque cada uno tiene una visión en particular de los objetos, la clasificación inteligente requiere una tremenda cantidad de perspicacia creativa y no existe la clasificación perfecta.

163. ¿Cómo es el rol del observador en la clasificación de objetos?

Trascendente, muy importante.

164. ¿Cuáles son las aproximaciones generales a la clasificación?

Son 3:

-categorización clásica

-agrupamiento conceptual

-teoría de prototipos

165. ¿Qué es la categorización clásica?

Se agrupan todas aquellas que posean una propiedad o conjunto de propiedades en común.

166. ¿Qué es el agrupamiento conceptual?

Se definen pautas descriptivas. Se desarrolla una estructura conceptual. Se agrupan todas aquellas que respondan a la descripción establecida.

167. ¿Qué es la teoría de prototipos?

Se crean clases prototípicas. A todas aquellas que se le aproximan en forma significativa se las considera pertenecientes a ese tipo.

168. ¿Qué es una abstracción clave?

Es una clase u objeto que forma parte del vocabulario del dominio del problema.

169. ¿Qué son los mecanismos?

Son medios por los cuales los objetos colaboran para proporcionar algún comportamiento de nivel superior.

UNIDAD II

170. ¿Qué es un Framework?

Un framework es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

171. ¿Qué son los frozen-spots en un Framework?

Algunas de las características del framework no son mutables ni tampoco pueden ser alteradas fácilmente. Estos puntos inmutables constituyen el núcleo o kernel de un

framework, también llamados como los puntos congelados o frozen-spots del framework.

172. ¿Qué son los hot-spots en un Framework?

Los puntos flexibles de un framework se llaman los puntos calientes (hot-spots). Los puntos calientes o Hot-spots son las clases o los métodos abstractos que deben ser implementados o puestos en ejecución.

173. ¿Cómo se puede clasificar un Framework según su extensibilidad?

Un framework se puede también clasificar según su extensibilidad; puede ser utilizado como una caja blanca o caja negra.

174. ¿Qué es un Framework de Caja Blanca?

Los frameworks de caja blanca también llamados frameworks con arquitectura de configuración-conducidos, la instanciación es solamente posible a través de la creación de nuevas clases. Estas clases y el código se pueden introducir en el marco por herencia o composición.

175. ¿Qué es un Framework de Caja Negra?

Los frameworks de caja negra producen instancias usando escrituras o scripts de configuración. Después de la configuración, una herramienta automática de instanciación crea las clases y el código de fuente (Source Code). La caja negra del framework no requiere que el usuario sepa los detalles internos del framework. Por lo tanto, estos frameworks también se llaman frameworks dato-conducidos [4]y son generalmente más fáciles de usar.

176. ¿Qué ventajas posee utilizar un Framework?

Las principales ventajas de la utilización de un framework son:

1. El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.

2. La reutilización de componentes software al por mayor. Los frameworks son los paradigmas de la reutilización.

3. El uso y la programación de componentes que siguen una política de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

177. ¿Qué problemas resuelve .NET Framework?

El objetivo de .NET es eliminar varios de los problemas que se le presentan a los desarrolladores. Antes de .NET pocas aplicaciones se ejecutaban en más de una plataforma de hardware y de software por lo tanto debían rescribir el software para

adaptarlo a los distintos entornos. Otro de los problemas era la comunicación y el intercambio de datos entre distintas aplicaciones.

La plataforma .NET resuelve estos problemas utilizando el Common Language Runtime que es independiente de hardware y software y por medio de la utilización de XML como lenguaje intercambio de datos universal entre aplicaciones. Ahora los desarrolladores pueden escribir aplicaciones en cualquier lenguaje .NET y estar seguros que pueden ser ejecutadas en todas las plataformas de hardware y software compatibles con .NET. El Common Language Runtime también se ocupa de problemas de infraestructura como por ejemplo el manejo de la seguridad y la memoria permitiéndole al desarrollador concentrarse en la lógica de su aplicación.

178. ¿Qué es y qué permite hacer el CLR?

El Common Languaje Runtima es la primera capa que pertenece a .NET Framework. Esta capa es la responsable de los servicios básicos de .Net, tales como la administración de memoria, la recolección de los elementos no utilizados y el control estructurado de excepciones y de subprocesamiento múltiples.

Es el motor de todo .net, este CLR vuelve a compilar esta vez para generar código nativo, es decir optimizado para el sistema operativo y el hardware actual Esta compilación la realiza el compilador llamado JIT (Just In Time)

179. ¿Qué es el MSIL?

MSIL significa Microsoft Inmediate Language

Cuando compilamos un assembly (que es un fichero de .net más o menos) da igual que el lenguaje que escojamos sea c# , Vb.net o pyton el código generado es en lenguaje MSIL (similar al Bytecode de Java)

Este MSIL es un lenguaje intermedio común a todos los sistemas operativos que soporten .net framework

180. ¿Qué es el CTS?

CTS significa Common Type System (CTS)

Es una especificación que define como el CRL utiliza y administra los tipos de datos. Básicamente es una relación entre los distintos tipos de datos que proporciona cada lenguaje. La consecuencia inmediata del CTS es que dentro de un mismo proyeto .NET podemos utilizar varios lenguajes .NET.

181. ¿Qué es el CLS?

La especificación común de lenguajes CLS hace posible que cualquier fabricante de software genere herramientas compatibles con .NET. De esta forma existen versiones .NET de Delphi y otros lenguajes ajenos a Microsoft.

UNIDAD III

182. ¿Qué es un Campo de una clase?

Un campo de una clase son variables internas que se encargaran de mantener los valores que las instancias de estas clases (Objetos) adoptaran en distintos momentos de su ciclo de vida (Estado).

183. ¿Qué es un método de una clase?

Un método es una acción dentro de una clase, estos se construyen por medio de las funciones y los procedimientos que poseen internamente.

184. ¿A qué denominamos sobrecarga?

Es cuando poseen un método con el mismo nombre y se diferencia por su firma.

185. ¿Qué es una propiedad de una clase?

Son campos de una clase que representan características de la misma y necesitan ser accedidos con el objeto de ser leídos o escritos.

186. ¿Qué tipos de propiedades existen?

Existen los siguientes tipos de propiedades:

- ✓ De escritura y lectura
- ✓ De solo lectura
- ✓ De solo escritura
- ✓ Con parámetros
- ✓ Por defecto

187. ¿Qué ámbitos pueden tener los campos, métodos y propiedades de las clases?

Pueden tener los ámbitos públicos o privados.

188. ¿Qué características posee cada ámbito existente si se lo aplico a un campo, un método y una propiedad de una clase?

Ámbito privado: Si posee un ámbito privado no se observara en la interface de los objetos de la clase que la implementa.

Ámbito público: Se puede acceder desde cualquier lado.

189. ¿Para qué se utilizan los constructores?

Se utilizan cuando se desea inicializar el estado de un objeto ni bien se crea

190. ¿A qué se denomina tiempo de vida de un objeto?

El tiempo de vida de un objeto es desde el momento en que se crea (Consume espacio en memoria) hasta que se recupera el espacio.

191. ¿Para administrar las instancias de .NET se utiliza un contador de referencias?

SI

192. ¿Qué objeto es el encargado de liberar el espacio ocupado por objetos que ya no se utilizan?

El objeto Destructor.

193. ¿Dónde se encuentran las instancias de los objetos administrados por el GC?

Se encuentran en memoria.

194. ¿Cuáles son los dos métodos más notorios que deben implementar las clases para trabajar correctamente con la recolección de elementos no utilizados y matar las instancias administradas y no administradas?

Los métodos son Finalize y Dispose.

195. ¿De dónde heredan las clases el método Finalize?

Lo Heredan de OBJECTS

196. ¿Cuál es la firma que implementa el método "Finalize"?

Protected Overridable Sub Finalize ()

197. ¿Qué método se utiliza para que el GC recolecte los elementos no utilizados?

Se utiliza el método collect.

198. ¿Qué método se utiliza para suspender el subproceso actual hasta que el subproceso que se está procesando en la cola de finalizadores vacíe dicha cola?

Se utiliza el método: GC.WaitForPendingFinalizers

199. ¿Cuándo se ejecuta el método collect del GC que método se ejecuta en los objetos Afectados?

Se ejecuta el Destructor.

200. ¿Qué método debería exponer una clase bien diseñada teniendo en consideración que no posee destructor?

Se deberá exponer el método Dispose

201. ¿Cómo obtengo el método "Dispose"?

Se obtiene según el programador mediante el siguiente código Implements IDispose.

202. ¿Qué se programa en el método "Dispose"?

Se programa mediante la interfaz IDispose que implementa el procedimiento Dispose.

203. ¿Se pueden combinar el uso de "Dispose" y "Finalize"?

SI

204. ¿A qué se denomina "Resurrección de Objetos"?

Cada vez que un objeto es eliminado del grafo de referencias de una aplicación, este pasa a una cola llamada la cola de finalización (comando de WinDBG !finlizaqueue) en la que se le da una última oportunidad de ejecutar el código que tenga en el destructor o en el método Dispose (si implementa IDisposable). En ese instante en el que el objeto está en la cola de finalización un objeto está fuera del grafo de objetos, pero si durante la ejecución de ese código se referencia a si mismo de otro objeto que está en el grafo de objetos de la aplicación a través de un objeto estático, diremos que el objeto ha sufrido una resurrección puesto que el recolector de basura sacará a ese objeto de la cola de finalización y el objeto será de nuevo referenciable y volverá a la vida (metafóricamente hablando).

205. ¿A qué se denomina "Generación" en el contexto de la recolección de elementos no utilizados?

Las generaciones son agrupaciones de las edades de los objetos en memoria

206. ¿Qué valores puede adoptar la "Generación" de un objeto?

Pueden optar entre 0 y 2.

207. ¿Cómo se puede obtener el número de veces que se ha producido la recolección de elementos no utilizados para la generación de objetos especificada?

La cantidad de veces está dada en el número de Generación ya que siempre que se realiza una recolección, a los objetos que sobreviven se los asciende.

208. ¿Cómo se obtiene el número de generación actual de un objeto?

Se obtiene mediante el método GetGeneration(Object)

209. ¿Cómo se puede recuperar el número de bytes que se considera que están asignados en la Actualidad?

Se recupera mediante el método GetTotalMemory

210. ¿Qué utiliza para convertir un objeto en "no" válido para la recolección de elementos no utilizados desde el principio de la rutina actual hasta el momento en que se llamó a este método?

Se utiliza el Método "KeepAlive"

211. ¿Cómo se solicita que el sistema no llame al finalizador del objeto especificado?

Se solicita mediante el método "SuppressFinalize"

212. ¿Cómo se solicita que el sistema llame al finalizador del objeto especificado, para el que previamente se ha llamado a "SuppressFinalize"?

Se solicita mediante el método "ReRegisterForFinalize"

213. ¿Cómo se obtiene el número máximo de generaciones que el sistema admite en la Actualidad?

Se obtiene mediante la propiedad "MaxGeneration"

214. ¿Qué son los sucesos?

Son reacciones de los objetos a estímulos externos. No sé cuando pasan.

215. ¿Qué se utiliza para declarar un suceso?

Se declaran mediante la palabra clave EVENT. Ejemplo: " AnEvent(ByVal EventNumber As Integer)"

216. ¿Cómo se logra que ocurra un suceso?

Para que ocurra un evento se utiliza la instrucción RaiseEvent.

Ejemplo: RaiseEvent AnEvent(EventNumber)

217. ¿Cómo se pueden atrapar los sucesos?

Los sucesos se atrapan mediante la instrucción WithEvents. Ejemplo; Dim WithEvents P10 As persona = New Persona.

218. ¿para qué se utiliza Addhandler en un suceso?

Asocia un evento a un controlador de eventos en tiempo de ejecución.

219. ¿Cómo y qué cosas se pueden compartir en una clase?

En una clase se puede compartir una estructura y comportamiento en común.

220. ¿Qué características poseen los campos compartidos?

Un campo Shared se crea cuando empieza a ejecutarse un programa, y deja de existir cuando el programa termina. Una variable Shared se inicializa con el valor predeterminado de su tipo.

221. ¿Qué características poseen los métodos compartidos?

El modificador Shared indica que el método no opera en una instancia específica de un tipo y que se puede invocar directamente desde un tipo, en vez de desde la instancia concreta del tipo

222. ¿Qué características poseen los sucesos compartidos?

Los Eventos compartidos no están asociados a instancias de clases.

223. ¿Para qué se utiliza AddHandler?

Se utiliza para asociar un evento a un controlador de eventos en tiempo de ejecución.

224. ¿se pueden atrapar sucesos desde matrices? ¿Cómo?

Sí, pero se deberá recorrer la matriz y llamar al evento en cada uno de sus ítem.

225. ¿Qué son los miembros compartidos?

Los miembros compartidos son propiedades, procedimientos y campos que comparten todas las instancias de una clase.

226. ¿Qué característica posee un campo compartido?

Los campos compartidos son un concepto de aprovisionamiento al que hacen referencia los tipos de contenido, de forma que si proporciona una colección de sitios con dos tipos de contenido independientes que usan el mismo campo compartido, el

script de aprovisionamiento para esos tipos de contenido crea una instancia del mismo campo, y cada instancia del campo comparte el mismo identificador único.

227. ¿Qué característica posee un método compartido?

En un método Shared no es válido hacer referencia a Me. Los métodos Shared no pueden ser Overridable, NotOverridable o MustOverride, ni pueden reemplazar otros métodos.

228. ¿Qué característica posee un constructor compartido?

A diferencia de los constructores de instancia, los constructores Shared tienen acceso Public implícito, no tienen parámetros y no pueden llamar a otros constructores.

229. ¿Qué característica posee un suceso compartido?

Los Eventos compartidos no están asociados a instancias de clases.

230. ¿Qué son y para que se pueden utilizar las clases anidadas?

Las clases anidadas son clases cuya declaración está contenida por completo en la declaración de clase de otra clase y que resultan útiles para programar modelos de objetos en los componentes.

231. ¿Qué ámbitos existen?

Public, protected, friend, protected friend, prívate.

232. ¿Qué característica posee el ámbito público?

Se utiliza para declarar una clase que sea accesible desde cualquier lugar del proyecto actual o desde un proyecto que referencie al proyecto actual.

233. ¿Qué característica posee el ámbito privado?

Se logra que la clase sea accesible solo desde dentro de quien la contiene.

234. ¿Qué característica posee el ámbito friend?

Permite que la clase posea visibilidad desde el ensamblado donde se encuentra.

235. ¿Qué característica posee el ámbito protected?

Se logra que la clase sea accesible desde dentro de sí misma o desde una subclase de ella.

236. ¿Qué característica posee el ámbito protected friend?

La visibilidad alcanza a la propia clase, sus clases derivadas y el ensamblado en el cual esta se encuentre.

237. ¿Qué es la herencia?

La herencia es una de las relaciones que pueden darse entre clases. Permite que una sublcase herede de una superclase de orden superior.

238. ¿Qué cosas se pueden heredar?

Hereda propiedades, métodos, eventos etc.

239. ¿Cómo y para qué se puede aprovechar en la práctica el polimorfismo?

El polimorfismo se aprovecha para poder dar mayor flexibilidad al modelo y poder abstraerse de que sublcase ejecutara el método, ya que se denomina igual en cada una de ellas.

240. ¿Cómo y para qué se utiliza la clase derived?

Se utiliza de la siguiente forma “Class derived : public clasebase”, se utiliza para poner en práctica la herencia antes indicada.

241. ¿Qué representa la clase me?

La palabra clave Me siempre hace referencia a la instancia específica de una clase o estructura donde se está ejecutando el código

242. ¿Qué clase representa a la clase base?

La Palabra clave MyBase.

243. ¿Para qué utilizaría MyBase?

La palabra clave MyBase se comporta como una variable de objeto que hace referencia a la clase base de la instancia actual de una clase. MyBase suele usarse para obtener acceso a los miembros de la clase base que se reemplazan o se sombrean en una clase derivada.

244. ¿Qué representa la clase MyClass?

La palabra clave MyClass se comporta como una variable de objeto que hace referencia a la instancia actual de una clase tal y como se implementó originalmente. MyClass similar a Me, pero todas las llamadas de método realizadas a través de ella se tratan como si el método fuese NotOverridable.

245. ¿Qué diferencia existe entre MyBase y MyClass?

Mybase hace referencia a la clase base y MyClass a una instancia.

246. ¿Para qué se usa una clase abstracta?

Las clases abstractas le permiten crear definiciones de comportamiento al mismo tiempo que proporcionan implementación común para las clases heredadas.

247. ¿Para qué se usa una clase sellada?

El modificador sealed se utiliza principalmente para impedir la derivación no intencionada, pero también permite algunas optimizaciones en tiempo de ejecución. En particular, como una clase sealed no puede tener clases derivadas, es posible transformar las llamadas virtuales a miembros de función en instancias de clase sealed en llamadas no virtuales.

248. ¿Qué es la sobreescritura?

Permite trabajar sobre un elemento heredado cuando no resulta útil la implementación que posee.

249. ¿Qué elementos se pueden sobreescribir?

Sólo puede utilizarse Overrides en una propiedad o instrucción de declaración de procedimiento.

250. ¿A qué se denomina sombreado de métodos?

Si dos elementos de programación comparten el mismo nombre, uno de ellos puede ocultar o sombrear al otro. En esta situación, el elemento sombreado no está disponible como referencia; en vez de esto, cuando el código utiliza el nombre del elemento, el compilador de Visual Basic resuelve en favor del elemento que sombrea.

251. ¿Qué característica peculiar posee el sombreado vs la sobreescritura?

Sombreado, Protege frente a una modificación posterior de clase base que introduce un miembro ya definido en la clase derivad. Sobrescribir, logra el polimorfismo mediante la definición de una implementación distinta de un procedimiento o propiedad con la misma secuencia de llamada.

UNIDAD IV

252. ¿Qué es una excepción?

Es una situación inesperada.

253. ¿Qué se coloca en el bloque "Catch"?

Se coloca el tipo de excepción que queremos capturar. Para eso utilizamos una variable de tipo exception.

254. ¿Cómo construiría un objeto del tipo "Exception" personalizado?

Ejemplo: catch ex as system.io.IOException

255. ¿Qué es una excepción?

Es una situación inesperada.

256. ¿Qué ocurre si en el bloque de código donde se produce la excepción el error no está siendo tratado?

Estas serán controladas por el runtime de .NET.

257. ¿Cuál es el objeto de mayor jerarquía para el manejo de excepciones?

La mayor Jerarquía la tiene Exception.

258. ¿En qué namespace se encuentra la clase Exception?

Se encuentra en el namespace System.

259. ¿Cuáles son las dos clases genéricas más importantes definidas en el FrameWork además de Exception?

SystemException y AplicationException.

260. ¿Qué instrucción se utiliza para poner en práctica el control e interceptar las excepciones?

Las instrucciones Try, catch y finally.

261. ¿Dónde se coloca el código protegido contra excepciones si se iniciara una excepción?

Se coloca entre la palabra clave Try y Catch.

262. ¿Qué se coloca en el bloque "Catch"?

Se coloca el tipo de excepción que queremos capturar. Para eso utilizamos una variable de tipo exception.

263. ¿Qué tipo de excepción se utiliza para interceptar un error de división por cero?

Se utiliza la excepción DivideByZeroException.

264. ¿Qué tipo de excepción se utiliza para interceptar una DLL que tiene problemas al ser cargada?

Se utiliza una AplicationException.

265. ¿Qué colocaría dentro de una clausula "Catch" para especificar una condición adicional que el bloque "Catch" deberá evaluar como verdadero para que sea seleccionada?

Se colocaría la palabra clave When.

266. ¿Si se desea colocar código de limpieza y liberación de recursos para que se ejecute cuando una excepción se produzca, dónde lo colocaría?

Se colocaría en bloque Finaly.

267. ¿Qué instrucción se utiliza para provocar un error y que el mismo se adapte al mecanismo de control de excepciones?

Se utiliza la instrucción throw.

268. ¿Escriba el código que permitiría provocar una excepción del tipo "ArgumentException"?

Imhertit ArgumentException (la clase hereda de ArgumentException)

269. ¿Cómo construiría un objeto del tipo "Exception" personalizado? Lo construiría con el siguiente código:

```
Class cargarArchivo
    Inherits system.applicationEsception
    Overrides ReadOnly Property message() As String
        Get
            Return "No se puede cargar el archivo"
        End Get
    End Property

End Class
```

270. ¿Cómo armaría un "Catch" personalizado para que se ejecute cuando se de la excepción "ClienteNoExisteException"?

```
Try
Catch ex as clienteNoExisteException
End try
```

271. ¿Para qué se utilizan las interfaces?

Se utilizan para poder definir firmas de un conjunto de miembros, luego estas interfaces podrán ser implementadas en clases.

272. ¿Cómo se implementa una interfaz?

Se implementan con la palabra reservada Implements + la Inteface Ejemplo: Implements IFigura

273. ¿Se pueden heredar las interfaces?

SI

274. ¿Se puede implementar un tipo de polimorfismo peculiar por medio de interfaces?

Si, se implementa el polimorfismo por interface.

275. ¿Para qué se utiliza la interfaz IComparable?

Se utiliza para ordenar un Array.

276. ¿Para qué se utiliza la interfaz IComparer?

Se utiliza para ordenar elementos.

277. ¿Para qué se utiliza la interfaz IClonable?

Se utiliza para clonar objetos.

278. ¿Para qué se utiliza la interfaz IEnumerable?

Se utiliza para relevar los objetos que tengo.

279. ¿Para qué se utiliza la interfaz IEnumerator?

Se utiliza para mostrar mis objetos y ocultar los objetos que yo quiera.

280. ¿Qué es un delegado?

Un delegado es una referencia a una función, lo que también se conoce como un puntero a una función, es decir un delegado permite delegado permite acceder a una función de forma casi anónima.

281. ¿A qué elementos se les puede delegar?

Se pueden delegar métodos o funciones.

282. ¿Qué se puede delegar?

Métodos y Funciones.

283. ¿Cómo construiría un delegado?

Pasos:

a) Declaro el tipo.

b) Declaro la variable.

c) Instanciaciòn.

d) Le indico el destino

e) Invoco.

284. ¿Cómo implementaría un procedimiento de devolución de llamadas?

Muestra todos los nombres de las carpetas en un árbol de directorios:

```
Sub DisplayDirectoryTree(ByVal path As String)
    For Each dirName As String In System.IO.Directory.GetDirectories(path)
        Console.WriteLine(dirName)
        DisplayDirectoryTree(dirName)
    Next

End Sub
```

285. ¿Para qué sirve la multidifusión de delegados?

Sirve para enviar una llamada a más de un procedimiento.

286. ¿Qué es un atributo?

El atributo es un nuevo concepto en la programación y se utiliza prácticamente en todas partes en . NET Framework. La idea subyacente es que-sin importar el idioma que está utilizando- algunos aspectos de la aplicación no se pueden expresar con instrucciones ejecutables.

287. ¿Cómo es la sintaxis de un atributo?

Se encierra un atributo entre corchetes angulares (<>) y la inserta inmediatamente antes del elemento al que se refiere. Por ejemplo, puede aplicar System.ComponentModel.DescriptionAttribute a una clase de la siguiente manera:

```
<System.ComponentModel.DescriptionAttribute("Person")>
Clase Persona
§

End Class
```

288. ¿Para qué se utiliza el atributo StructLayout?

El Atributo StructLayout de Visual Basic. NET define atributos que permiten controlar los elementos de Estructura de un bloque que están en la memoria y la forma en que el tiempo de ejecución se va a calcular cuando son pasados a una función en una DLL externa.

289. ¿Para qué se utiliza el atributo FieldOffset?

Se utiliza para Especificar en Byte la distancia desde el principio de la estructura. Este atributo debe estar en la definición de todos los campos de la estructura.

290. ¿Para qué se utiliza el atributo Conditional?

Se utiliza para incluir o excluir partes del código en el compilador.

291. ¿Para qué se utiliza el atributo Obsolete?

Se utiliza para reemplazar una función obsoleta por otra nueva, con el fin de optimizar mi programa.

292. ¿Para qué se utiliza el atributo DebuggerStepThrough?

Se utiliza para marcar una rutina que debe ser saltado por el Visual Studio. NET debugger porque debe ejecutar en su conjunto o sólo porque ya se ha probado y depurado .

UNIDAD VI

293. ¿Qué características posee un esquema cliente / servidor?

Permite distribuir de manera más eficiente la carga de procesos adaptándolos a cada situación.

294. ¿Qué significa pasar información batch?

Esta información antes de ser transmitida se almacena en un repositorio y son recibidos de a lotes por el otro extremo de forma asíncrona.

295. ¿Qué significa pasar información on line?

Los datos son recibidos y procesados en tiempo real. Generando mayor fluidez de información en el sistema.

296. ¿Qué es un protocolo?

Es un conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos para transmitir información .

297. ¿Qué protocolo usa una red de área local?

Se utiliza el protocolo TCP (Transfer Control Protocol)

298. ¿Qué protocolo usa Internet?

Usa el protocolo IP.

299. ¿Qué hace el protocolo IP?

Proporciona información de ruteo en redes basadas en IP (Permite que el datagrama salte entre los routers para definitivamente llegar a destino)

300. ¿Qué ventajas tiene distribuir procesos?

Tiene la ventaja de no consumir tanto recurso de un mismo servidor y optimizar el tiempo de respuesta.

301. ¿Qué ventajas tiene distribuir almacenamientos?

Tiene a disposición la información tanto en el Servidor como en el mismo cliente sin generar dependencia directa.

302. ¿Qué es un socket?

Es un conjunto de dos partes de IP y puertos que forman dos extremos en una conexión.

303. ¿Qué característica posee un socket sincrónico?

a) El socket se bloquea hasta que recibe toda la información.

b) Cuando se hace una llamada para recibir, la llamada a la funcion no termina hasta que recibe los datos, esto puede congelar la aplicación.

304. ¿Qué característica posee un socket asincrónico?

El socket asincrónico no se bloquea mientras el servidor devuelve una respuesta.

305. ¿Qué objeto se puede utilizar para construir un navegador?

El objeto Webbrowser con sus siguientes variantes:
Webbrowser1.Gohome
- WebBrowser1.Stop
- Webbrowser1.Goback
- WebBrowser1.GoForward
- WebBrowser1.Gosearch

306. ¿Para qué se utilizan los puertos de la pc?

Se utilizan para conectar diversos dispositivos independientemente de sus funciones (impresoras, reproductores MP3, bocinas, pantallas LCD, ratones (Mouse), PDA, etc.)
Puertos físicos de la computadora: son conectores integrados en tarjetas de expansión ó en la tarjeta principal "Motherboard" de la computadora; diseñados con formas y características electrónicas especiales, utilizados para interconectar una gran gama de dispositivos externos con la computadora, es decir, los periféricos.
Usualmente el conector hembra estará montado en la computadora y el conector macho estará integrado en los dispositivos ó cables.
Varía la velocidad de transmisión de datos y la forma física del puerto acorde al estándar y al momento tecnológico.

307. ¿Qué puertos posee una PC?

1) Puertos de uso general:
Puerto eSATA
Puerto USB
Puerto FireWire ó IEEE1394
Puerto SCSI
Puerto paralelo / LPTx
Puerto serial / COMx
2) Puertos para impresoras: soportan solamente la conexión de impresoras y algunos Plotter.
Puerto Centronics para impresora
3) Puertos para teclado y ratón: su diseño es exclusivo para la conexión de teclados y ratones (Mouse).
Puerto miniDIN - PS/2
Puerto DIN - PS/1
4) Puertos para dispositivos de juegos: permiten la conexión de palancas, almohadillas y volantes de juego.
Puerto de juegos Gameport (DB15)
5) Puertos de video: permiten la transmisión de señales procedentes de la tarjeta de video hacia una pantalla ó proyector.
Puerto DisplayPort (transmite video, sonido y datos de manera simultánea)
Puerto HDMI (transmite video, sonido y datos de manera simultánea)
Puerto DVI
Puerto S-Video
Puerto VGA
Puerto RCA
Puerto CGA
Puerto EGA

6) Puertos de red: permiten la interconexión de computadoras por medio de cables.
Puerto RJ45 (para red local LAN)
Puerto RJ11 (para red telefónica)
Puerto de red BNC
Puerto de red DB15
7) Puertos de sonido: permiten la conexión de sistemas de sonido como bocinas, amplificadores, etc.
Puerto Jack 3.5"

308. ¿Cómo funciona el puerto paralelo?

La mayoría de los puertos paralelos de la parte posterior de las computadoras compatibles poseen un conector DB-25. Las señales que ocupan esas terminales se pueden dividir en cuatro grupos básicos: tierras, salidas de datos, entradas de dialogo y salidas de dialogo. En la imagen las tierras se indican con círculos, las entradas de dialogo se indican con flechas que apuntan al conector y las salidas (tanto de datos como de dialogo) tienen flechas que apuntan hacia afuera del conector. Envía varios bits simultáneamente.

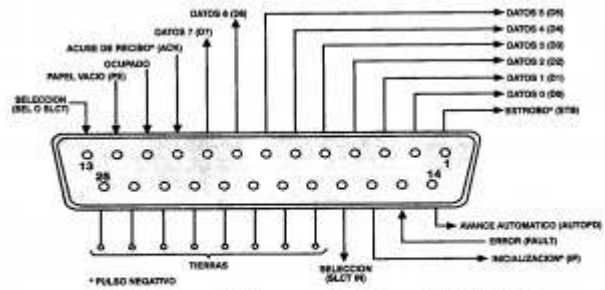

FIGURA 1: En la mayoría de las computadoras IBM compatibles (y algunas no compatibles) el puerto paralelo es como éste. Las flechas que apuntan hacia afuera del conector DB-25 son salidas, las que apuntan hacia adentro, entradas. Los terminales restantes, señalados con círculos, son tierras.

309. ¿Cómo funciona el puerto serie?

La transmisión en serie reduce la complejidad y parte del coste del sistema, pero obteniendo a cambio una menor eficacia: es necesario un intervalo de tiempo ocho veces mayor para transmitir ocho bits individuales que para transmitirlos simultáneamente.

310. ¿Cómo funciona el puerto USB?

El "Universal Serial Bus", mejor conocido como USB, es un método para conectar dispositivos periféricos a una computadora para que esta pueda hacer uso de ellos. El USB se considera que es superior a los métodos anteriores de conexión en un número de maneras, incluyendo la facilidad de la conexión en sí y las velocidades a las que los datos pueden ser transferidos entre el dispositivo periférico y la computadora.

311. ¿Qué es la domótica?

El término proviene del latín domus añadiéndole el final de la palabra "informática" y, según explica la propia Real Academia Española de la Lengua, es el "conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda". El principal objetivo de estas tecnologías es la mejora de la calidad de vida incrementando la comodidad de los inquilinos, sin embargo, últimamente se está imponiendo como una tendencia en el mundo de la ecología...

Ya sea por el ahorro económico, la obligatoriedad gubernamental o por la mentalidad ecológica la verdad es que cada vez más empresas y particulares optan por los sistemas automáticos de control de los edificios. Como explican en el manual difundido por el Instituto para la Diversificación y el Ahorro Energético (IDAE) titulado Cómo ahorrar energía instalando domótica en su vivienda, "aprovechando mejor los recursos naturales se puede reducir la factura energética mientras se gana en confort y seguridad".