|  |  |  |
| --- | --- | --- |
|  | **FICHA TÉCNICA DE CONSTRUCCIÓN DEL ÍTEM** | |
|  | **No. Ítem**: **1** |
|
|
| **DATOS DEL ÍTEM** | | **DATOS DEL AUTOR** |
| **Programa académico**: Ingeniería de Sistemas | |  |
| **Prueba**: Programación Avanzada | |  |
|  |
|  |
| **ÍTEM: COMPETENCIA ESPECÍFICA, CONTEXTO, ENUNCIADO Y OPCIONES DE RESPUESTA** | | |
| **Competencia específica señalada en el syllabus, que evalúa este ítem:**  Conocimientos básicos de los fundamentos de los lenguajes de programación empleados para el desarrollo de aplicaciones, que deben brindar al estudiante el fundamento para la creación tanto de programas básicos como de algunos más avanzados | | |
| **CONTEXTO - Caso - situación problémica**:  Observe la siguiente gráfica: | | |
| **ENUNCIADO**:  Especifique qué método se encarga de generar la salida de la aplicación que se muestra: | | |
| **Opciones de respuesta**  a. public void imprimirTabla(int n)  {  int i = 0;    System.out.print("N");  System.out.print("\t");  System.out.print("10\*N");  System.out.print("\t");  System.out.print("100\*N");  System.out.print("\t");  System.out.print("1000\*N");  System.out.println("\t");    for (i = 0; i >= n; i++)  {  System.out.print(i+1);  System.out.print("\t");  System.out.print((i+1)\*10);  System.out.print("\t");  System.out.print((i+1)\*100);  System.out.print("\t");  System.out.println((i+1)\*1000);  }  }  b. public void imprimirTabla(int n)  {    System.out.println("N");  System.out.println("\t");  System.out.println("10\*N");  System.out.println("\t");  System.out.println("100\*N");  System.out.println("\t");  System.out.println("1000\*N");  System.out.println("\t");    for (i = 0; i <= n; i++)  {  System.out.print(i);  System.out.print("\t");  System.out.print(i\*10);  System.out.print("\t");  System.out.print(i\*100);  System.out.print("\t");  System.out.println(i\*1000);  }  }  c. public void imprimirTabla(int n)  {  int i = 0;    System.out.print("N");  System.out.print("\t");  System.out.print("10\*N");  System.out.print("\t");  System.out.print("100\*N");  System.out.print("\t");  System.out.print("1000\*N");  System.out.println("\t");    for (i = 0; i < n; i++)  {  System.out.print(i+1);  System.out.print("\t");  System.out.print((i+1)\*10);  System.out.print("\t");  System.out.print((i+1)\*100);  System.out.print("\t");  System.out.println((i+1)\*1000);  }  }  d. public void imprimirTabla(int n)  {  int i = 0;    System.out.print("N");  System.out.print("\t");  System.out.print("10\*N");  System.out.print("\t");  System.out.print("100\*N");  System.out.print("\t");  System.out.print("1000\*N");  System.out.println("\t");    for (i = 1; i < n; i++)  {  System.out.print(i);  System.out.print("\t");  System.out.print(i+1\*10);  System.out.print("\t");  System.out.print(i+1\*100);  System.out.print("\t");  System.out.println(i+1\*1000);  }  } | | |
|
|
| **JUSTIFICACIÓN DE OPCIONES DE RESPUESTA** | | |
| Por qué NO es a: porque como se observa en la implementación del bucle repetitivo, este se está condicionando con una estructura que de acuerdo al esquema lógico (for (i = 0; i >= n; i++)). Esta instrucción hace que el ciclo jamás se ejecute, y, por ende, ninguna de las instrucciones, a pesar de ser correctas, van a ejecutarse dentro del cuerpo del bucle repeticional. | | |
| Por qué NO es b: porque se presentan dos errores importantes. El primero está relacionado con la estructura de la condición lógica que implementa el ciclo. Este está bien estructurado, pero si se observa el condicionante especificado este haría que se ejecutara una instrucción adicional, puesto que, si se envía como parámetro un valor de 5, por ejemplo, lo hará 6 veces. Por otro lado, se observa otro error en las instrucciones de impresión, ya que, en la salida, esta empieza desde uno y con la forma como están escritas, empezaría no desde 1, sino desde cero. | | |
| Por qué NO es d: porque al no contar con el uso de paréntesis en las instrucciones de impresión, el operador de multiplicación tiene mayor nivel de jerarquía, por lo que los resultados de salida no van a corresponder a los mostrados en la consola de impresión de la aplicación. | | |
| **CLAVE Y JUSTIFICACIÓN.**  La clave es c porque en la estructura que se muestra en la construcción del método, el parámetro permite definir el valor límite hasta el cual se imprime (es decir, el número de líneas sobre las cuales se aplican todas las condiciones). Por otro lado, el manejo de los paréntesis en las impresiones, permite realizar correctamente el incremento y posteriormente la multiplicación por el factor de cómputo de cada columna de la salida impresa (10, 100, 1000, etc), y la incrementación desde 1 en adelante (i+1) permite que la primera línea inicie desde 1 en adelante. | | |
| **ESPECIFICACIONES DE DISEÑO: DIBUJOS, ECUACIONES Y / O GRÁFICOS**: | | |

|  |  |  |
| --- | --- | --- |
|  | **FICHA TÉCNICA DE CONSTRUCCIÓN DEL ÍTEM** | |
|  | **No. Ítem**: **2** |
|
|
| **DATOS DEL ÍTEM** | | **DATOS DEL AUTOR** |
| **Programa académico**: Ingeniería de Sistemas | |  |
| **Prueba**: Programación Avanzada | |  |
|  |
|  |
| **ÍTEM: COMPETENCIA ESPECÍFICA, CONTEXTO, ENUNCIADO Y OPCIONES DE RESPUESTA** | | |
| **Competencia específica señalada en el syllabus, que evalúa este ítem:**  Conocimientos específicos en la temática de Herencia, Polimorfismo, Encapsulamiento y Abstracción en Java, su estructura, características, y ejemplos que permitan que el estudiante comprenda el mecanismo como forma de optimización en la creación de aplicaciones en el modelo de abstracción orientado a objetos. | | |
| **CONTEXTO - Caso - situación problémica**:  Observe con atención la siguiente definición: "Es la propiedad que define el ocultamiento del estado, es decir, de los datos miembro, de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.” | | |
| **ENUNCIADO**:  ¿A cuál característica de la programación orientada a objetos corresponde? | | |
| **Opciones de respuesta**  a. Polimorfismo.  b. Herencia.  c. Encapsulamiento.  d. Ocultación. | | |
|
|
| **JUSTIFICACIÓN DE OPCIONES DE RESPUESTA** | | |
| Por qué NO es a: porque el concepto de polimorfismo está asociado directamente con la forma como los métodos son implementados y la forma como dichos métodos son ejecutados en el contexto de las diferentes aplicaciones. El polimorfismo no está directamente asociado con el estado de los datos sino con la forma como se moldea el comportamiento de los objetos, por lo cual es una respuesta distractora. | | |
| Por qué NO es b: porque la herencia es un concepto asociado a la definición estructural de la transmisión del comportamiento y los datos en una jerarquía de clases, la cual no afecta el cambio de estado de un atributo a través de las operaciones, dado que cada instancia en la jerarquía maneja sus propios datos en su propio espacio de memoria. | | |
| Por qué NO es d: porque, aunque el concepto semánticamente es similar a encapsulamiento, los datos no están siendo escondidos, sino transformados por las operaciones. Es una respuesta similar a la correcta, pero la definición no es la apropiada. | | |
| **CLAVE Y JUSTIFICACIÓN.**  La clave es c porque el encapsulamiento como su nombre lo indica, representa almacenar los datos fuera del alcance o injerencia de otros objetos, por lo que la definición del modificador de visibilidad o acceso (private) establece que los datos pertenecen únicamente al objeto que los implementa, por lo que los demás objetos solo pueden acceder a su modificación, si contienen métodos públicos que desencapsulen la información y permitan la modificación del estado contenido en ellos, independientemente de su dominio (tipo de dato). | | |
| **ESPECIFICACIONES DE DISEÑO: DIBUJOS, ECUACIONES Y / O GRÁFICOS**: | | |

|  |  |  |
| --- | --- | --- |
|  | **FICHA TÉCNICA DE CONSTRUCCIÓN DEL ÍTEM** | |
|  | **No. Ítem**: **3** |
|
|
| **DATOS DEL ÍTEM** | | **DATOS DEL AUTOR** |
| **Programa académico**: Ingeniería de Sistemas | |  |
| **Prueba**: Programación Avanzada | |  |
|  |
|  |
| **ÍTEM: COMPETENCIA ESPECÍFICA, CONTEXTO, ENUNCIADO Y OPCIONES DE RESPUESTA** | | |
| **Competencia específica señalada en el syllabus, que evalúa este ítem:**  Conocimientos de lógica algorítmica y mecanismos de definición de procesos para la resolución de problemas de organización y carácter lógico matemático. | | |
| **CONTEXTO - Caso - situación problémica**:  Observe la siguiente clase:  public class Ejercicio2  {  public Ejercicio2()  {  }  public static void main(String[] args)  {  double resultado = 0.0;  resultado = Ejercicio2.calcularRetencion(10000,20);  System.out.println("La retención para el precio de  10000 pesos fue de " + resultado);  }    public double calcularRetencion (double precio, double impuesto)  {  double retencion = 0.0;    if (precio >= 10000 && precio < 20000)  {  retencion = precio \* impuesto + 100;  }  else  if (precio >= 20000 && precio < 50000)  {  retencion = (precio + 1500) \* (2\*impuesto);  }  else  {  retencion = precio + (3 \* impuesto);  }  return (retencion);  }  } | | |
| **ENUNCIADO**:  ¿Qué resultado considera usted que se generaría con la implementación de la clase anterior? | | |
| **Opciones de respuesta**  a. 200100.00.  b. 105450.00.  c. Un error.  d. Un warning. | | |
|
|
| **JUSTIFICACIÓN DE OPCIONES DE RESPUESTA** | | |
| Por qué NO es a: porque, aunque matemática y algorítmicamente el resultado si es el que se debe generar a partir del procesamiento de los datos en el algoritmo mostrado por el método, el estudiante debe identificar que el método está siendo llamado en el método principal main() de forma estática, y el método como tal en su signatura no está siendo declarado de dicha forma, por lo que el procedimiento de invocación es incorrecto, independiente de que el resultado lógico o semántico esté bien expresado. Es una respuesta altamente distractora. | | |
| Por qué NO es b: porque el cálculo del algoritmo con los datos dados no corresponde al valor proporcionado en la respuesta. Adicionalmente, se debe tener en cuenta que el concepto explicado en el ítem a) es aplicable igualmente a esta respuesta, ya que el estudiante debe analizar que falta instanciar el objeto para poder invocar el método en cuestión de cálculo lógico. | | |
| Por qué NO es d: porque al compilar el programa, no se genera un warning (una advertencia) sino un error directo, por lo cual la aplicación no corre. | | |
| **CLAVE Y JUSTIFICACIÓN.**  La clave es c porque al intentar compilar la aplicación, el compilador detecta que se está intentando ejecutar un método de forma estática, y dicho método no tiene una signatura de este tipo, por lo cual la aplicación no puede enlazarse y por ende ejecutarse en el contexto de la máquina virtual de Java. | | |
| **ESPECIFICACIONES DE DISEÑO: DIBUJOS, ECUACIONES Y / O GRÁFICOS**: | | |

|  |  |  |
| --- | --- | --- |
|  | **FICHA TÉCNICA DE CONSTRUCCIÓN DEL ÍTEM** | |
|  | **No. Ítem**: **4** |
|
|
| **DATOS DEL ÍTEM** | | **DATOS DEL AUTOR** |
| **Programa académico**: Ingeniería de Sistemas | |  |
| **Prueba**: Programación Avanzada | |  |
|  |
|  |
| **ÍTEM: COMPETENCIA ESPECÍFICA, CONTEXTO, ENUNCIADO Y OPCIONES DE RESPUESTA** | | |
| **Competencia específica señalada en el syllabus, que evalúa este ítem:**  Conocimientos de lógica algorítmica y mecanismos de definición de procesos para la resolución de problemas de organización y carácter lógico matemático. | | |
| **CONTEXTO - Caso - situación problémica**:  Teniendo la siguiente implementación hecha en Java:  public class Ejercicio3  {  public Ejercicio3()  {  }  public static void main(String[] args)  {  String cadena1 = "esternocleidomastoideo";  String cadena2 = "imperio austrohungaro";  String cadena3 = "diseño de sistemas de informacion";    Ejercicio3 manejador = new Ejercicio3();  manejador.extrarSubcadena(cadena1);  manejador.extrarSubcadena(cadena2);  manejador.extrarSubcadena(cadena3);  }    public void extrarSubcadena (String original)  {  String subcadena = "";  subcadena = original.substring(  original.length()/4,original.length()-2);  System.out.println("CADENA OBTENIDA = " + subcadena);  }  } | | |
| **ENUNCIADO**:  Especifique las cadenas de caracteres que podrían obtenerse. | | |
| **Opciones de respuesta**  a. Esternocleido, imperio austrohun, diseño de sistemas de inf.  b. Cleidomastoideo, austrohungaro, istemas de informacion.  c. Nocleidomastoid, io austrohunga, e sistemas de informaci.  d. Esternoclei, mperio austrohu, ño de sistemas de inform. | | |
|
|
| **JUSTIFICACIÓN DE OPCIONES DE RESPUESTA** | | |
| Por qué NO es a: porque el método define que la cadena se origina en la posición indicada por la operación matemática resultante de dividir la longitud de la cadena entre 4, por lo que si se observa la primera cadena de longitud (22), se genera un valor de 5 (división entera), por lo que la cadena iniciaría en nocleidomastoideo, y no desde el inicio de la cadena (posición 0) que se observa en la respuesta del ítem a). | | |
| Por qué NO es b: porque el método define que la cadena se origina en la posición indicada por la operación matemática resultante de dividir la longitud de la cadena entre 4, por lo que si se observa la primera cadena de longitud (22), se genera un valor de 5 (división entera), por lo que la cadena iniciaría en nocleidomastoideo, y no desde la posición 7 (c), que se observa en la respuesta del ítem b). | | |
| Por qué NO es d: porque el método define que la cadena se origina en la posición indicada por la operación matemática resultante de dividir la longitud de la cadena entre 4, por lo que si se observa la primera cadena de longitud (22), se genera un valor de 5 (división entera), por lo que la cadena iniciaría en nocleidomastoideo, y no desde el inicio de la cadena (posición 0) que se observa en la respuesta del ítem d). | | |
| **CLAVE Y JUSTIFICACIÓN.**  La clave es c porque como se observa en la estructura de las cadenas de la respuesta, cada una inicia en la posición obtenida de la división de la longitud de la cadena entre 4 y finaliza en la longitud de la cadena menos 2 posiciones. La siguiente tabla ilustra el resultado del procesamiento del algoritmo del método. | | |
| **ESPECIFICACIONES DE DISEÑO: DIBUJOS, ECUACIONES Y / O GRÁFICOS**: | | |

|  |  |  |
| --- | --- | --- |
|  | **FICHA TÉCNICA DE CONSTRUCCIÓN DEL ÍTEM** | |
|  | **No. Ítem**: **5** |
|
|
| **DATOS DEL ÍTEM** | | **DATOS DEL AUTOR** |
| **Programa académico**: Ingeniería de Sistemas | |  |
| **Prueba**: Programación Avanzada | |  |
|  |
|  |
| **ÍTEM: COMPETENCIA ESPECÍFICA, CONTEXTO, ENUNCIADO Y OPCIONES DE RESPUESTA** | | |
| **Competencia específica señalada en el syllabus, que evalúa este ítem:**  Conocimientos de lógica algorítmica y mecanismos de definición de procesos para la resolución de problemas de organización y carácter lógico matemático. Igualmente se requiere de conocimientos de la API Java IO con el fin de comprender los principales elementos de la esencia de los flujos de entrada y salida y su procesamiento a nivel de memoria y acceso a disco. | | |
| **CONTEXTO - Caso - situación problémica**:  Se tiene una aplicación Java que gestiona un proceso de lectura y escritura de archivos planos en disco. Indique. Se intenta ejecutar esta aplicación, y el archivo que es referenciado no está presente en la unidad de disco local.  import java.io.FileInputStream;  import java.io.IOException;  public class Excepciones  {  public static void main( String[] args )  {  // Para leer un fichero  Lanzadora lanza = new Lanzadora();  FileInputStream entrada = null;  int leo;  try  {  entrada = new FileInputStream( "fich.txt" );  while ( ( leo = entrada.read() ) != -1 )  lanza.lanzaSiNegativo( leo );  entrada.close();  System.out.println( "Todo fue bien" );  }  catch ( MiExcepcion e )  {  // Personalizada  System.out.println( "Excepcion: " + e.getMessage() );  }  catch ( IOException e )  { // Estándar  System.out.println( "Excepcion: " + e.getMessage());  }  finally  {  if ( entrada != null )  try  {  System.out.println("Se cierre al flujo");  entrada.close();  // Siempre queda cerrado  }  catch ( Exception e )  {  System.out.println( "Excepcion: " + e.getMessage() );  }  System.out.println( "Fichero cerrado." );  }  }  } | | |
| **ENUNCIADO**:  Indique de las respuestas que se presentan a continuación, que mensajes de depuración del aplicativo son impresos en la salida de la consola estándar de Entrada y Salida (I/O) de Java. | | |
| **Opciones de respuesta**  a. Excepción: fich.txt (El sistema no puede encontrar el archivo especificado)  Se cierra el flujo  b. Todo fue bien  Excepción: .java.io.FileNotFoundException:(The system cannot find the file specified)  c. Todo fue bien  Excepcion: .java.io.FileNotFoundException:(The system cannot find the file specified)  Se cierra el flujo  d. Excepción: fich.txt (El sistema no puede encontrar el archivo especificado)  Fichero cerrado. | | |
|
|
| **JUSTIFICACIÓN DE OPCIONES DE RESPUESTA** | | |
| Por qué NO es a: porque al no existir el archivo, aunque el error está controlado por la excepción personalizada definida en el código del catch, el objeto entrada nunca se puede instanciar, por lo que el mensaje de depuración "Se cierra el flujo" jamás puede imprimirse, pero si el mensaje general que está después del control de dicha excepción ("Fichero cerrado"), el cual en la respuesta no está presente. Es por lo tanto, una respuesta distractora. | | |
| Por qué NO es b: porque el no poder apresurarse el flujo sobre el recurso fich.txt, la excepción controlada se dispara y el mensaje "Todo fue bien", no alcanza a ejecutarse, por lo que no tiene sentido que esta sea la salida de depuración al ejecutarse esta aplicación bajo estas condiciones. | | |
| Por qué NO es c: porque la traza del objeto de la excepción no está siendo impreso (y dado que al lanzar el método printStackTrace () del objeto de excepción se genera el mensaje en inglés y la implementación no está llamando dicho método del objeto en cuestión. Igualmente, no se imprime el mensaje "Se cierra el flujo", ya que como se explicó arriba, esto solo se haría si la instancia del objeto entrada fuera diferente de null, y dado que el objeto no se instancia, no tiene sentido que esta operación de depuración se llegue a ejecutar. | | |
| **CLAVE Y JUSTIFICACIÓN.**  La clave es d porque Excepcion: fich.txt (El sistema no puede encontrar el archivo especificado). Fichero cerrado., ya que como lo muestra la lógica del algoritmo, en el contexto de las circunstancias de ejecución del método (ausencia del archivo fich.txt), la excepción personalizada se dispara, y se genera la impresión del mensaje "Excepción: " + e.getMessage(), que retorna la traza de error devuelta por el Sistema Operativo a la máquina virtual. Luego, dado que el bloque finally, se ejecuta, independientemente de que el try lo haga correctamente o no, el único mensaje que se imprime es "Se cerro el fichero", el cual no depende de ninguna condición lógica, y que como se observa, parte de la respuesta de este ítem. | | |
| **ESPECIFICACIONES DE DISEÑO: DIBUJOS, ECUACIONES Y / O GRÁFICOS**: | | |