|  |  |  |
| --- | --- | --- |
|  | **TC1014 Fundamentos de programación**  **31 de marzo de 2017**  **Conceptos - Parcial 2** |  |

**Nombre \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Matrícula \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Firma \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. (10%) El error en el llamado al método calculaDivisores y su definición se resuelve:

public static int calculaDivisores (int n){ String numeroDivi, elemento;

.

return numeroDivi;

}

public static void main(String args[]) {

…

String respuesta = calculaDivisores(5);

}

1. Al corregir el nombre del método en la invocación a calculaDIvisores
2. Al corregir el tipo de dato del regreso declarado en la firma de la función int calculaDivisores
3. Al corregir el tipo de dato de la variable de regreso de la función String numeroDivi
4. Al corregir el tipo de dato de la variable que recibe la respuesta en el llamado a la función String respuesta

2. (10%) La instrucción

do { ……

} while (condición);

1. Se ejecuta solamente una vez
2. Se ejecuta cero veces
3. Se ejecuta al menos una vez
4. Se ejecuta cero veces o más

3. (10%) La condición

do {

} while (seguir=="si");

1. Evalúa que la variable “seguir” sea diferente del valor “si”
2. Evalúa que la variable “seguir” sea igual al valor “si”
3. Evalúa que la variable “seguir” sea diferente del valor “si” y continúa en el ciclo
4. Evalúa que la variable “seguir” sea igual al valor “si” y continúa en el ciclo

1. (10%) La condición

while (numero <= n)

* 1. Vuelve a entrar al ciclo si “numero” es menor que n
  2. Vuelve a entrar al ciclo si “numero” es menor o igual que n
  3. Vuelve a entrar al ciclo si “numero” es mayor que n
  4. Vuelve a entrar al ciclo si “numero” es mayor o igual que n

1. (10%) La instrucción en java donde resultado, n y numero son enteros genera

resultado = n % numero;

* 1. El cociente de n entre numero
  2. El divisor de n entre numero
  3. El residuo de n entre numero
  4. El dividendo de n entre numero

1. (10%) La estructura de programa

if ( a > 1)

if (b > 2)

* 1. Es un if complejo
  2. Es un if simple
  3. Es un if anidado
  4. Es un if compuesto

1. (10%) En la definición de un ciclo la variable de control se debe

* 1. Iniciar, modificar y reiniciar
  2. Iniciar, evaluar y modificar
  3. Declarar, evaluar y modificar
  4. Declarar, iniciar y modificar

1. (10%) Las instrucciones siguientes permiten:

double variable; s = new Scanner(System.in);

variable = s.nextDouble();

* 1. Imprimir en pantalla el contenido de variable como valor con decimales
  2. Leer del usuario el valor y guardarlo en variable como valor con decimales
  3. Leer del usuario el valor y guardarlo en variable como valor entero
  4. Imprimir en pantalla el contenido de variable como valor entero

1. (10%) La instrucción “else” en una estructura de condición if

* 1. Es obligatoria
  2. Solamente se puede utilizar en if complejos
  3. Solamente se puede utilizar en if simples
  4. Es opcional

1. (10%) En el proceso de aprendizaje de la programación, el código en java

* 1. Se puede escribir después de hacer el razonamiento y el pseudo-código
  2. Se puede escribir después de hacer el razonamiento
  3. Se puede escribir sin hacer el razonamiento, ni el pseudo-código
  4. Se puede escribir y después desarrollar el razonamiento y el pseudo-código