Estructuras de control en Java

Ejercicios Extras

[Guía3]

# Ejercicios extras

Estos ejercicios son para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado con los ejercicios prácticos del día y aún no haya finalizado el encuentro. Recuerda que no es necesario que termines estos ejercicios extra para continuar con lo siguiente.

1. Dado un tiempo en minutos, calcular su equivalente en días y horas. Por ejemplo, si el usuario ingresa 1600 minutos, el sistema debe calcular su equivalente: 1 día, 2 horas.
2. Declarar cuatro variables de tipo entero A, B, C y D y asignarle un valor diferente a cada una. A continuación, realizar las instrucciones necesarias para que: B tome el valor de C, C tome el valor de A, A tome el valor de D y D tome el valor de B. Mostrar los valores iniciales y los valores finales de cada variable. Utilizar sólo una variable auxiliar.
3. Elaborar un algoritmo en el cuál se ingrese una letra y se detecte si se trata de una vocal. Caso contrario mostrar un mensaje. Nota: investigar la función equals() de la clase String.
4. Elaborar un algoritmo en el cuál se ingrese un número entre 1 y 10 y se muestre su equivalente en romano.
5. Una obra social tiene tres clases de socios:

* Los socios tipo ‘A’ abonan una cuota mayor, pero tienen un 50% de descuento en todos los tipos de tratamientos.
* Los socios tipo ‘B’ abonan una cuota moderada y tienen un 35% de descuento para los mismos tratamientos que los socios del tipo A.
* Los socios que menos aportan, los de tipo ‘C’, no reciben descuentos sobre dichos tratamientos.
* Solicite una letra (carácter) que representa la clase de un socio, y luego un valor real que represente el costo del tratamiento (previo al descuento) y determine el importe en efectivo a pagar por dicho socio.

1. Leer la altura de N personas y determinar el promedio de estaturas que se encuentran por debajo de 1.60 mts. y el promedio de estaturas en general.
2. Realice un programa que calcule y visualice el valor máximo, el valor mínimo y el promedio de n números (n>0). El valor de n se solicitará al principio del programa y los números serán introducidos por el usuario. Realice dos versiones del programa, una usando el bucle “while” y otra con el bucle “do - while”.
3. Escriba un programa que lea números enteros. Si el número es múltiplo de cinco debe detener la lectura y mostrar la cantidad de números leídos, la cantidad de números pares y la cantidad de números impares. Al igual que en el ejercicio anterior los números negativos no deben sumarse. Nota: recordar el uso de la sentencia break.
4. Simular la división usando solamente restas. Dados dos números enteros mayores que uno, realizar un algoritmo que calcule el cociente y el residuo usando sólo restas. Método: Restar el dividendo del divisor hasta obtener un resultado menor que el divisor, este resultado es el residuo, y el número de restas realizadas es el cociente.

Por ejemplo: 50 / 13:

50 – 13 = 37 una resta realizada

37 – 13 = 24 dos restas realizadas

24 – 13 = 11 tres restas realizadas

dado que 11 es menor que 13, entonces: el residuo es 11 y el cociente es 3.

¿Aún no lo entendiste? Recomendamos googlear división con restas sucesivas.

1. Realice un programa para que el usuario adivine el resultado de una multiplicación entre dos números generados aleatoriamente entre 0 y 10. El programa debe indicar al usuario si su respuesta es o no correcta. En caso que la respuesta sea incorrecta se debe permitir al usuario ingresar su respuesta nuevamente. Para realizar este ejercicio investigue como utilizar la función Math.random() de Java.
2. Escribir un programa que lea un número entero y devuelva el número de dígitos que componen ese número. Por ejemplo, si introducimos el número 12345, el programa deberá devolver 5. Calcular la cantidad de dígitos matemáticamente utilizando el operador de división. Nota: recordar que las variables de tipo entero truncan los números o resultados.
3. Necesitamos mostrar un contador con 3 dígitos (X-X-X), que muestre los números del 0-0-0 al 9-9-9, con la particularidad que cada vez que aparezca un 3 lo sustituya por una E. Ejemplo:

0-0-0

0-0-1

0-0-2

0-0-E

0-0-4

0-1-2

0-1-E

**Nota:** investigar función equals() y como convertir números a String.

1. Crear un programa que dibuje una escalera de números, donde cada línea de números comience en uno y termine en el número de la línea. Solicitar la altura de la escalera al usuario al comenzar. Ejemplo: si se ingresa el número 3:

1

12

123

1. Se dispone de un conjunto de N familias, cada una de las cuales tiene una M cantidad de hijos. Escriba un programa que pida la cantidad de familias y para cada familia la cantidad de hijos para averiguar la media de edad de los hijos de todas las familias.
   * + 1. Dado un tiempo en minutos, calcular su equivalente en días y horas. Por ejemplo, si el usuario ingresa 1600 minutos, el sistema debe calcular su equivalente: 1 día, 2 horas.

package tiempo;

// @author Horacio

 import java.util.Scanner; //Importar la clase Scanner del paquete java.util

public class Tiempo { //Clase principal Tiempo

    public static void main(String[] args) { //Método main

        Scanner input = new Scanner(System.in); //Crear un objeto Scanner para leer la entrada del usuario

        System.out.print("Ingrese el tiempo en minutos: "); //Mostrar mensaje al usuario

        int minutos = input.nextInt(); //Leer el tiempo en minutos ingresado por el usuario

        int dias = minutos / 1440; //Calcular el número de días a partir de los minutos ingresados. 1440´ equivalen a 1 día.

        int horas = (minutos % 1440) / 60; //Calcular el número de horas a partir de los minutos ingresados

        System.out.println(minutos + " minutos equivalen a " + dias + " día(s) y " + horas + " hora(s)."); //Mostrar el resultado al usuario

    }

}

Este programa en Java recibe un tiempo en minutos ingresado por el usuario y calcula su equivalente en días y horas.

Primero se importa la clase Scanner del paquete java.util para leer la entrada del usuario.

Luego se crea un objeto Scanner y se muestra un mensaje pidiendo al usuario que ingrese el tiempo en minutos.

Después de leer el tiempo en minutos ingresado por el usuario, se realizan los cálculos necesarios

para obtener el número de días y horas. Para calcular el número de días, se divide el tiempo en minutos entre

el número de minutos en un día (1440).

Para calcular el número de horas, se utiliza el operador módulo (%) para obtener el resto de la división

 entre los minutos ingresados y el número de minutos en un día,

y luego se divide el resultado entre el número de minutos en una hora (60).

Finalmente, se muestra el resultado al usuario utilizando System.out.println(),

donde se concatena el tiempo en minutos ingresado, el número de días y el número de horas calculados.

1. Declarar cuatro variables de tipo entero A, B, C y D y asignarle un valor diferente a cada una. A continuación, realizar las instrucciones necesarias para que: B tome el valor de C, C tome el valor de A, A tome el valor de D y D tome el valor de B. Mostrar los valores iniciales y los valores finales de cada variable. Utilizar sólo una variable auxiliar.
2. package cambiovariables;
3. // @author Horacio
4. public class CambioVariables {
5. public static void main(String[] args) {
6. int A = 5; // Declaración y asignación de la variable A
7. int B = 10; // Declaración y asignación de la variable B
8. int C = 15; // Declaración y asignación de la variable C
9. int D = 20; // Declaración y asignación de la variable D
10. int auxiliar = B; // Se guarda el valor de B en la variable auxiliar
11. B = C; // Se asigna el valor de C a la variable B
12. C = A; // Se asigna el valor de A a la variable C
13. A = D; // Se asigna el valor de D a la variable A
14. D = auxiliar; // Se asigna el valor de la variable auxiliar a la variable D
15. // Se muestran los valores iniciales y finales de cada variable
16. System.out.println("Valores iniciales:");
17. System.out.println("A: " + 5);
18. System.out.println("B: " + 10);
19. System.out.println("C: " + 15);
20. System.out.println("D: " + 20);
21. System.out.println("Valores finales:");
22. System.out.println("A: " + A);
23. System.out.println("B: " + B);
24. System.out.println("C: " + C);
25. System.out.println("D: " + D);
26. }
27. }

\*Este programa en Java declara cuatro variables de tipo entero y les asigna un valor

\*diferente a cada una. Luego, se utiliza una variable auxiliar para intercambiar los

\*valores de las variables de la siguiente manera: B toma el valor de C, C toma el valor

\*de A, A toma el valor de D y D toma el valor de B. Finalmente, se muestran los

\*valores iniciales y finales de cada variable.

1. Elaborar un algoritmo en el cuál se ingrese una letra y se detecte si se trata de una vocal. Caso contrario mostrar un mensaje. Nota: investigar la función equals() de la clase String.

package vocaldetector;

// @author Horacio

import java.util.Scanner; // Importamos la clase Scanner para poder leer la entrada del usuario

public class VocalDetector {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in); // Creamos un objeto Scanner para leer la entrada del usuario

        System.out.println("Ingresa una letra: ");

        String letra = sc.nextLine(); // Leemos la letra ingresada por el usuario y la almacenamos en una variable

        // Comparamos la letra ingresada con las vocales

        if (letra.equals("a") || letra.equals("e") || letra.equals("i") || letra.equals("o") || letra.equals("u")) {

            System.out.println("La letra ingresada es una vocal.");

        } else {

            System.out.println("La letra ingresada no es una vocal.");

        }

    }

}

\*En este ejemplo, primero importamos la clase Scanner para poder leer la entrada

\*del usuario. Luego, creamos un objeto Scanner y le pedimos al usuario que ingrese

\*una letra. La letra ingresada se almacena en una variable llamada letra.

\*Después, comparamos la letra ingresada con las vocales utilizando la función

\*equals() de la clase String. Si la letra es una vocal, se muestra un mensaje

\*indicando que la letra es una vocal. Si no es una vocal, se muestra un mensaje

\*indicando que la letra no es una vocal.

1. Elaborar un algoritmo en el cuál se ingrese un número entre 1 y 10 y se muestre su equivalente en romano.
2. package numeroaromano;
3. // @author Horacio
4. import java.util.Scanner; //importamos la clase Scanner para poder leer la entrada del usuario
5. public class NumeroARomano {
6. public static void main(String[] args) {
7. Scanner scanner = new Scanner(System.in); //creamos una instancia de la clase Scanner
8. System.out.println("Ingrese un número entre 1 y 10: "); //solicitamos al usuario que ingrese un número
9. int numero = scanner.nextInt(); //leemos el número ingresado por el usuario
10. String numeroRomano = ""; //inicializamos la variable que contendrá el número romano
11. switch (numero) { //evaluamos el número ingresado por el usuario
12. case 1:
13. numeroRomano = "I";
14. break;
15. case 2:
16. numeroRomano = "II";
17. break;
18. case 3:
19. numeroRomano = "III";
20. break;
21. case 4:
22. numeroRomano = "IV";
23. break;
24. case 5:
25. numeroRomano = "V";
26. break;
27. case 6:
28. numeroRomano = "VI";
29. break;
30. case 7:
31. numeroRomano = "VII";
32. break;
33. case 8:
34. numeroRomano = "VIII";
35. break;
36. case 9:
37. numeroRomano = "IX";
38. break;
39. case 10:
40. numeroRomano = "X";
41. break;
42. default:
43. System.out.println("El número ingresado no está dentro del rango permitido."); //si el número ingresado no está dentro del rango permitido, mostramos un mensaje
44. break;
45. }
46. System.out.println("El número [" + numero + "] en números romanos es: " + numeroRomano); //mostramos el resultado
47. }
48. }
49. =============================================================================================================
50. \*Este algoritmo utiliza una estructura switch para evaluar el número ingresado por el
51. \*usuario y asignarle su equivalente en números romanos a la variable
52. \*numeroRomano. Luego, se muestra el resultado al usuario en la última línea. Si el
53. \*número ingresado por el usuario no está dentro del rango permitido (1-10), se muestra
54. \*un mensaje de error.
55. Una obra social tiene tres clases de socios:

* Los socios tipo ‘A’ abonan una cuota mayor, pero tienen un 50% de descuento en todos los tipos de tratamientos.
* Los socios tipo ‘B’ abonan una cuota moderada y tienen un 35% de descuento para los mismos tratamientos que los socios del tipo A.
* Los socios que menos aportan, los de tipo ‘C’, no reciben descuentos sobre dichos tratamientos.
* Solicite una letra (carácter) que representa la clase de un socio, y luego un valor real que represente el costo del tratamiento (previo al descuento) y determine el importe en efectivo a pagar por dicho socio.
* package obrasocial; // Declaración del paquete obrasocial
* import java.util.Scanner; // Importación de la clase Scanner
* public class ObraSocial{ // Declaración de la clase ObraSocial
* public static void main(String[] args) { // Declaración del método principal
* Scanner sc = new Scanner(System.in); // Creación del objeto Scanner
* // Solicitar al usuario la clase del socio
* System.out.print("Ingrese la clase del socio (A, B o C): ");
* char claseSocio = sc.next().charAt(0); // Lectura del carácter ingresado por el usuario
* // Solicitar al usuario el costo del tratamiento
* System.out.print("Ingrese el costo del tratamiento: ");
* double costoTratamiento = sc.nextDouble(); // Lectura del número decimal ingresado por el usuario
* // Calcular el importe a pagar por el socio
* double importeAPagar; // Declaración de la variable importeAPagar
* if (claseSocio == ('A') || (claseSocio == 'a')) { // Condicional if para la clase A. Comparamos con un condicional caracter 'a'
* importeAPagar = costoTratamiento \* 0.5;
* } else if (claseSocio == ('B') || (claseSocio == 'b')) { // Condicional if para la clase B. Comparamos con un condicional caracter 'b'.
* importeAPagar = costoTratamiento \* 0.65;
* } else { // Para cualquier otra clase
* importeAPagar = costoTratamiento;
* }
* // Mostrar el importe a pagar por el socio
* System.out.println("El importe a pagar es: " + importeAPagar); // Impresión del mensaje y del valor de la variable importeAPagar
* sc.close(); // Cierre del objeto Scanner
* }
* } // Cierre de la clase ObraSocial
* /\*
* Para reemplazar Condicional if por switch el código es:
* switch (claseSocio) {
* case 'A':
* importeAPagar = costoTratamiento \* 0.5;
* break;
* case 'B':
* importeAPagar = costoTratamiento \* 0.65;
* break;
* default:
* importeAPagar = costoTratamiento;
* break;
* }
* \*/

1. Leer la altura de N personas y determinar el promedio de estaturas que se encuentran por debajo de 1.60 mts. y el promedio de estaturas en general.
2. package estaturas;
3. // @author Horacio
4. import java.util.Scanner; // Importamos la clase Scanner para poder leer datos del usuario
5. public class Estaturas {
6. public static void main(String[] args) {
7. Scanner sc = new Scanner(System.in); // Creamos un objeto Scanner para leer datos del usuario
8. double sumaEstaturas = 0; // Inicializamos la variable que almacenará la suma de todas las estaturas ingresadas por el usuario
9. int contadorEstaturas = 0; // Inicializamos el contador de estaturas ingresadas por el usuario
10. double sumaEstaturasBajo160 = 0; // Inicializamos la variable que almacenará la suma de todas las estaturas por debajo de 1.60 metros
11. int contadorEstaturasBajo160 = 0; // Inicializamos el contador de estaturas por debajo de 1.60 metros
12. System.out.print("Ingresa el número de personas: "); // Pedimos al usuario que ingrese el número de personas
13. int numeroPersonas = sc.nextInt(); // Leemos el número de personas ingresado por el usuario
14. for (int i = 1; i <= numeroPersonas; i++) { // Iteramos desde 1 hasta el número de personas ingresado por el usuario
15. System.out.print("Ingresa la estatura de la persona " + i + " en metros: "); // Pedimos al usuario que ingrese la estatura de la persona
16. double estatura = sc.nextDouble(); // Leemos la estatura ingresada por el usuario
17. sumaEstaturas += estatura; // Sumamos la estatura ingresada a la suma total de estaturas
18. contadorEstaturas++; // Incrementamos el contador de estaturas
19. if (estatura < 1.6) { // Comprobamos si la estatura ingresada es menor a 1.60 metros
20. sumaEstaturasBajo160 += estatura; // Sumamos la estatura ingresada a la suma total de estaturas por debajo de 1.60 metros
21. contadorEstaturasBajo160++; // Incrementamos el contador de estaturas por debajo de 1.60 metros
22. }
23. }
24. double promedioEstaturas = sumaEstaturas / contadorEstaturas; // Calculamos el promedio de todas las estaturas ingresadas
25. double promedioEstaturasBajo160 = sumaEstaturasBajo160 / contadorEstaturasBajo160; // Calculamos el promedio de las estaturas por debajo de 1.60 metros
26. System.out.println("El promedio de estaturas es: " + promedioEstaturas); // Mostramos el promedio general de estaturas
27. System.out.println("El promedio de estaturas por debajo de 1.60 metros es: " + promedioEstaturasBajo160); // Mostramos el promedio de estaturas por debajo de 1.60 metros
28. sc.close(); // Cerramos el objeto Scanner para liberar recursos
29. }
30. }
31. /\*
32. ==========================================================================
33. \*Este programa en Java lee la altura de N personas y calcula el promedio de \*estaturas que se encuentran por debajo de 1.60 metros y el promedio de las \*estaturas en general.
34. \*Primero, se importa la clase Scanner para poder leer los datos del usuario. Luego, \*se inicializan las variables que se usarán para almacenar la suma y el contador de \*todas las estaturas ingresadas y de las estaturas por debajo de 1.60 metros \*ingresadas.
35. \*Se solicita al usuario que ingrese el número de personas y se lee con el objeto \*Scanner. Luego, se itera desde 1 hasta el número de personas ingresado por el \*usuario y se solicita al usuario que ingrese la estatura de cada persona.
36. \*La estatura ingresada se suma a la suma total de estaturas y se incrementa el \*contador de estaturas. Si la estatura es menor a 1.60 metros, se suma a la suma \*total de estaturas por debajo de 1.60 metros y se incrementa el contador de \*estaturas por debajo de 1.60 metros.
37. \*Finalmente, se calcula el promedio de todas las estaturas y el promedio de las \*estaturas por debajo de 1.60 metros y se muestran en la pantalla. Se cierra el objeto \*Scanner para liberar recursos.
38. \*/
39. Realice un programa que calcule y visualice el valor máximo, el valor mínimo y el promedio de n números (n>0). El valor de n se solicitará al principio del programa y los números serán introducidos por el usuario. Realice dos versiones del programa, una usando el bucle “while” y otra con el bucle “do - while”.
40. package valormaximo;
41. // @author Horacio
42. import java.util.Scanner;
43. public class ValorMaximo {
44. public static void main(String[] args) {
45. Scanner input = new Scanner(System.in); // Crear objeto Scanner para leer entrada del usuario
46. int n; // Variable para almacenar el valor de n
47. double num, total = 0, max = Double.MIN\_VALUE, min = Double.MAX\_VALUE; // Inicializar variables totales, máximo y mínimo
48. // Pedir al usuario que ingrese el valor de n
49. System.out.print("Ingrese el valor de n: ");
50. n = input.nextInt();
51. // Bucle while para calcular el valor máximo, el valor mínimo y el promedio de n números
52. int i = 1; // Inicializar contador
53. while (i <= n) {
54. System.out.print("Ingrese el número " + i + ": ");
55. num = input.nextDouble();
56. // Actualizar valor máximo y mínimo
57. if (num > max) {
58. max = num;
59. }
60. if (num < min) {
61. min = num;
62. }
63. // Agregar número al total
64. total += num;
65. // Incrementar contador
66. i++;
67. }
68. // Calcular promedio
69. double promedio = total / n;
70. // Mostrar resultados
71. System.out.println("Valor máximo: " + max);
72. System.out.println("Valor mínimo: " + min);
73. System.out.println("Promedio: " + promedio);
74. // Bucle do-while para calcular el valor máximo, el valor mínimo y el promedio de n números
75. i = 1; // Reinicializar contador
76. total = 0; // Reinicializar variable total
77. max = Double.MIN\_VALUE; // Reinicializar variable máxima
78. min = Double.MAX\_VALUE; // Reinicializar variable mínima
79. do {
80. System.out.print("Ingrese el número " + i + ": ");
81. num = input.nextDouble();
82. // Actualizar valor máximo y mínimo
83. if (num > max) {
84. max = num;
85. }
86. if (num < min) {
87. min = num;
88. }
89. // Agregar número al total
90. total += num;
91. // Incrementar contador
92. i++;
93. } while (i <= n);
94. // Calcular promedio
95. promedio = total / n;
96. // Mostrar resultados
97. System.out.println("Valor máximo: " + max);
98. System.out.println("Valor mínimo: " + min);
99. System.out.println("Promedio: " + promedio);
100. }
101. }
102. ==========================================================================
103. \*En este programa en Java, se pide al usuario que ingrese el valor de n y luego
104. \*introduzca n números. Luego, se calcula el valor máximo, el valor mínimo y el
105. \*promedio de los números ingresados.
106. \*Para hacer esto, se utilizan dos bucles diferentes: el bucle while y el bucle do- while.
107. \*Ambos bucles realizan la misma operación, pero el bucle do-while se
108. \*ejecuta al menos una vez, mientras que el bucle while puede no ejecutarse en
109. \*absoluto si n es igual a cero.
110. \*La entrada del usuario se lee utilizando un objeto Scanner. Se inicializan las
111. \*variables totales, máximas y mínimas con valores predeterminados. Luego, se
112. \*solicita al usuario que ingrese n y se ejecuta el bucle while o el bucle do- while, según corresponda.
113. \*Dentro del bucle, se solicita al usuario que ingrese un número y se actualiza el
114. \*valor máximo y mínimo según sea necesario. Además, el número se agrega al
115. \*total. Una vez que se han ingresado todos los números, se calcula el promedio y se
116. \*muestran los resultados.
117. Escriba un programa que lea números enteros. Si el número es múltiplo de cinco debe detener la lectura y mostrar la cantidad de números leídos, la cantidad de números pares y la cantidad de números impares. Al igual que en el ejercicio anterior los números negativos no deben sumarse. Nota: recordar el uso de la sentencia break.
118. package multiploscinco;
119. // @author Horacio
120. import java.util.Scanner; // importar la clase Scanner para recibir entrada del usuario
121. public class MultiplosCinco {
122. public static void main(String[] args) {
123. Scanner entrada = new Scanner(System.in); // crear objeto Scanner para recibir entrada del usuario
124. int num, contador = 0, pares = 0, impares = 0; // inicializar variables contador, pares e impares
125. while (true) { // bucle infinito
126. System.out.print("Ingrese un número entero: ");
127. num = entrada.nextInt(); // leer número entero del usuario
128. if (num % 5 == 0) { // si el número es múltiplo de 5
129. break; // salir del bucle
130. }
131. if (num < 0) { // si el número es negativo
132. continue; // saltar al siguiente número
133. }
134. contador++; // incrementar el contador de números leídos
135. if (num % 2 == 0) { // si el número es par
136. pares++; // incrementar el contador de números pares
137. } else { // si no
138. impares++; // incrementar el contador de números impares
139. }
140. }
141. // mostrar resultados
142. System.out.println("Cantidad de números leídos: " + contador);
143. System.out.println("Cantidad de números pares: " + pares);
144. System.out.println("Cantidad de números impares: " + impares);
145. }
146. }
147. =========================================================================================================================
148. \*El programa primero importa la clase Scanner para recibir entrada del usuario. Luego se
149. \*define la clase MultiplosCinco con su método main. Dentro del método main, se crea
150. \*un objeto Scanner para recibir entrada del usuario. También se inicializan las variables
151. \*num, contador, pares e impares en 0.
152. \*A continuación, se inicia un bucle while infinito. En cada iteración, el programa le pide al
153. \*usuario que ingrese un número entero y lo lee con el método nextInt() del objeto Scanner.
154. \*Si el número es múltiplo de 5, se sale del bucle usando la sentencia break. Si el
155. \*número es negativo, se salta al siguiente número usando la sentencia continue.
156. \*Si el número es positivo, se incrementa el contador contador y se comprueba si es par o impar.
157. \*En caso de ser par, se incrementa el contador pares y en caso de ser impar, se incrementa \*el contador impares.
158. \*Finalmente, se muestran los resultados de los contadores contador, pares e impares.
159. Simular la división usando solamente restas. Dados dos números enteros mayores que uno, realizar un algoritmo que calcule el cociente y el residuo usando sólo restas. Método: Restar el dividendo del divisor hasta obtener un resultado menor que el divisor, este resultado es el residuo, y el número de restas realizadas es el cociente.

Por ejemplo: 50 / 13:

50 – 13 = 37 una resta realizada

37 – 13 = 24 dos restas realizadas

24 – 13 = 11 tres restas realizadas

dado que 11 es menor que 13, entonces: el residuo es 11 y el cociente es 3.

¿Aún no lo entendiste? Recomendamos googlear división con restas sucesivas.

package divisionrestassucesivas;

// @author Horacio

 import java.util.Scanner;

public class DivisionRestasSucesivas {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Ingrese el dividendo: ");

        int dividendo = sc.nextInt();

        System.out.print("Ingrese el divisor: ");

        int divisor = sc.nextInt();

        int cociente = 0;

        while (dividendo >= divisor) { // mientras el dividendo sea mayor o igual al divisor

            dividendo -= divisor; // restar el divisor del dividendo

            cociente++; // aumentar el contador de cociente

        }

        System.out.println("El cociente es: " + cociente);

        System.out.println("El residuo es: " + dividendo);

    }

}

/\*

==========================================================================================

\*El código anterior toma dos números enteros, el dividendo y el divisor, y usa un bucle while

\*para restar el divisor del dividendo hasta que el dividendo sea menor que el divisor.

\*El número de veces que se realiza la resta se almacena en una variable contador de cociente,

\* y el resultado final es el cociente y el residuo.

\*/

Versión optimizada del código:

import java.util.Scanner;

public class DivisionRestasSucesivasOptimizada {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Ingrese el dividendo: ");

        int dividendo = sc.nextInt();

        System.out.print("Ingrese el divisor: ");

        int divisor = sc.nextInt();

        int cociente = 0;

        while (dividendo >= divisor) { // mientras el dividendo sea mayor o igual al divisor

            int resta = divisor; // guardar el valor del divisor en una variable resta

            int contadorRestas = 1; // inicializar el contador de restas en 1

            while (resta <= dividendo - resta) { // mientras el valor de la resta sea menor o igual que la diferencia entre el dividendo y la resta

                resta += resta; // duplicar el valor de la resta

                contadorRestas += contadorRestas; // duplicar el contador de restas

            }

            dividendo -= resta; // restar el valor de la resta del dividendo

            cociente += contadorRestas; // sumar el contador de restas al contador de cociente

        }

        System.out.println("El cociente es: " + cociente);

        System.out.println("El residuo es: " + dividendo);

    }

}

/\*

==========================================================================================

En este código optimizado, se utiliza una técnica de duplicación para reducir el número de restas necesarias.

En lugar de restar el divisor de forma repetida, se duplica el valor de la resta en cada iteración del bucle interior

hasta que sea mayor que la diferencia entre el dividendo y la resta.

Luego, se resta el valor de la resta del dividendo y se suma el contador de restas al contador de cociente.

Esto reduce el número total de restas necesarias para calcular el cociente y el residuo.

\*/

1. Realice un programa para que el usuario adivine el resultado de una multiplicación entre dos números generados aleatoriamente entre 0 y 10. El programa debe indicar al usuario si su respuesta es o no correcta. En caso que la respuesta sea incorrecta se debe permitir al usuario ingresar su respuesta nuevamente. Para realizar este ejercicio investigue como utilizar la función Math.random() de Java.
2. package randommultiplicationgame;
3. // @author Horacio
4. import java.util.Scanner;
5. public class RandomMultiplicationGame {
6. public static void main(String[] args) {
7. Scanner input = new Scanner(System.in);
8. int num1, num2, respuesta, respuestacorrecta;
9. boolean correcta = false;
10. //generar números aleatorios entre 0 y 10
11. num1 = (int) (Math.random() \* 11);
12. num2 = (int) (Math.random() \* 11);
13. //calcular la respuesta correcta
14. respuestacorrecta = num1 \* num2;
15. //bucle hasta que el usuario adivine la respuesta correcta
16. while(!correcta){
17. System.out.print("¿Cuánto es " + num1 + " multiplicado por " + num2 + "? ");
18. respuesta = input.nextInt();
19. if(respuesta == respuestacorrecta){
20. System.out.println("Correcto!");
21. correcta = true;
22. }
23. else{
24. System.out.println("Incorrecto. Vuelva a intentarlo.");
25. }
26. }
27. }
28. }
29. /\*
30. //Función Math.random() optimizada
31. public static double optimizedRandom(){
32. long a = 25214903917L;
33. long c = 11L;
34. long m = (long) Math.pow(2, 48);
35. long x = System.nanoTime();
36. x = (a \* x + c) % m;
37. return ((double) x / m);
38. }
39. \*/

El programa comienza importando la clase Scanner para permitir que el usuario ingrese datos desde la consola. Luego, se declaran las variables para los dos números aleatorios, la respuesta del usuario, la respuesta correcta y un valor booleano para indicar si el usuario ha adivinado correctamente el resultado de la multiplicación.

Se generan los números aleatorios utilizando la función Math.random(), que devuelve un número aleatorio entre 0 y 1. Para obtener un número aleatorio entre 0 y 10, se multiplica el número generado por 11 y se convierte a un entero.

Luego, se calcula la respuesta correcta multiplicando los dos números generados aleatoriamente.

El programa entra en un bucle while que se ejecuta hasta que el usuario adivina la respuesta correcta. En cada iteración del bucle, se le pide al usuario que ingrese su respuesta y se verifica si es correcta o no. Si la respuesta es correcta, se establece la variable booleana correct en true y se imprime un mensaje de confirmación. Si la respuesta es incorrecta, se imprime un mensaje de error y el bucle continúa.

La función optimizedRandom() utiliza una implementación optimizada de la función Math.random(). En lugar de utilizar la función incorporada, se utiliza una fórmula personalizada que utiliza un generador de números pseudoaleatorios lineal congruente. Esto se hace para mejorar la calidad y la aleatoriedad de los números generados.

1. Escribir un programa que lea un número entero y devuelva el número de dígitos que componen ese número. Por ejemplo, si introducimos el número 12345, el programa deberá devolver 5. Calcular la cantidad de dígitos matemáticamente utilizando el operador de división. Nota: recordar que las variables de tipo entero truncan los números o resultados.
2. package contadordedigitos;
3. // @author Horacio
4. import java.util.Scanner; // Importar la clase Scanner para leer la entrada del usuario
5. public class ContadorDeDigitos {
6. public static void main(String[] args) {
7. Scanner input = new Scanner(System.in); // Crear un objeto Scanner para leer la entrada del usuario
8. System.out.print("Ingrese un número entero: "); // Tener en cuenta que al System.out.print[ln] le falta la 'ln'.
9. int numero = input.nextInt(); // Leer el número ingresado por el usuario
10. int digitos = 0; // Inicializar la variable que cuenta los dígitos en cero
11. while (numero != 0) { // Mientras el número no sea cero
12. digitos++; // Incrementar la variable de conteo de dígitos
13. numero /= 10; // Dividir el número entre 10 para eliminar el último dígito
14. }
15. System.out.println("El número ingresado tiene " + digitos + " dígitos.");
16. }
17. }

Ejemplo: Clase optimizada para truncar números o resultados, se puede utilizar la clase DecimalFormat de Java para especificar el número de decimales que se desea mostrar. Aquí hay un ejemplo de cómo se puede utilizar:

package truncador;

// @author Horacio

 import java.text.DecimalFormat;

public class Truncador {

  public static void main(String[] args) {

    double numero = 3.141592653589793238;

    DecimalFormat df = new DecimalFormat("#.##"); // Especificar el formato de truncamiento

    System.out.println(df.format(numero)); // Imprimir el número truncado con dos decimales

  }

}

En la línea DecimalFormat df = new DecimalFormat("#.##");, se especifica que se desea mostrar el número con dos decimales (#.##). Luego, en la línea System.out.println(df.format(numero));, se aplica el formato de truncamiento al número y se imprime en la consola. El resultado sería 3.14.

1. Necesitamos mostrar un contador con 3 dígitos (X-X-X), que muestre los números del 0-0-0 al 9-9-9, con la particularidad que cada vez que aparezca un 3 lo sustituya por una E. Ejemplo:

0-0-0

0-0-1

0-0-2

0-0-E

0-0-4

0-1-2

0-1-E

**Nota:** investigar función equals() y como convertir números a String.

package contador;

// @author Horacio

public class Contador {

    public static void main(String[] args) {

        // Iteramos sobre los números del 0 al 999

        for (int i = 0; i < 1000; i++) {

            // Convertimos el número a una cadena de tres dígitos

            String numero = String.format("%03d", i);

            // Reemplazamos los 3 por E

            numero = numero.replace("3", "E");

            // Mostramos el número en el formato X-X-X

            System.out.println(numero.charAt(0) + "-" + numero.charAt(1) + "-" + numero.charAt(2));

        }

    }

    // Función equals optimizada para comparar dos números en formato String

    public static boolean equals(String numero1, String numero2) {

        if (numero1.length() != numero2.length()) {

            return false;

        }

        for (int i = 0; i < numero1.length(); i++) {

            if (numero1.charAt(i) != numero2.charAt(i)) {

                return false;

            }

        }

        return true;

    }

    // Función para convertir un número entero a una cadena de n dígitos rellenando con ceros a la izquierda

    public static String convertirNumero(int numero, int n) {

        return String.format("%0" + n + "d", numero);

    }

}

En primer lugar, se define una clase Contador. Dentro de esta clase se define un método main que es el punto de entrada del programa.

Dentro del método main, se utiliza un bucle for para iterar sobre los números del 0 al 999. En cada iteración, se convierte el número a una cadena de tres dígitos utilizando String.format("%03d", i). La cadena resultante tiene siempre tres dígitos y rellena con ceros a la izquierda si es necesario.

A continuación, se utiliza el método replace de la clase String para reemplazar todos los caracteres "3" por la letra "E". Finalmente, se muestra el número en el formato X-X-X utilizando System.out.println.

Además, se han definido dos funciones adicionales. La primera es una función equals optimizada para comparar dos números en formato **String**. La función compara los dos números carácter por carácter y devuelve true si son iguales y false en caso contrario.

La segunda función es convertirNumero, que se utiliza para convertir un número entero a una cadena de n dígitos rellenando con ceros a la izquierda. Esta función es útil cuando se quiere generar números con un formato específico.

1. Crear un programa que dibuje una escalera de números, donde cada línea de números comience en uno y termine en el número de la línea. Solicitar la altura de la escalera al usuario al comenzar. Ejemplo: si se ingresa el número 3:

1

12

123

package escaleranumeros;

// @author Horacio

import java.util.Scanner;

public class EscaleraNumeros {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Ingrese la altura de la escalera: ");

        int altura = sc.nextInt();

        sc.close();

        // Bucle para imprimir cada línea de la escalera

        for (int i = 1; i <= altura; i++) {

            // Bucle para imprimir los números en cada línea

            for (int j = 1; j <= i; j++) {

                System.out.print(j);

            }

            System.out.println();

        }

    }

}

Este programa en Java dibuja una escalera de números según la altura ingresada por el usuario. Primero, se importa la clase Scanner para poder leer la entrada del usuario. Luego se solicita al usuario que ingrese la altura de la escalera y se guarda en una variable "altura". Se cierra el objeto Scanner para liberar recursos.

Luego, se utiliza un bucle "for" para imprimir cada línea de la escalera. El bucle comienza en 1 y se repite hasta que la variable "i" sea igual a la altura ingresada por el usuario. En cada iteración del bucle exterior, se utiliza otro bucle "for" para imprimir los números en cada línea. Este segundo bucle comienza en 1 y se repite hasta que la variable "j" sea igual a la variable "i" (número de la línea actual). Dentro de este segundo bucle, se imprime el valor de la variable "j". Después de imprimir todos los números de la línea actual, se imprime una nueva línea para comenzar la siguiente línea de la escalera.

1. Se dispone de un conjunto de N familias, cada una de las cuales tiene una M cantidad de hijos. Escriba un programa que pida la cantidad de familias y para cada familia la cantidad de hijos para averiguar la media de edad de los hijos de todas las familias.
2. package mediaedadhijos;
3. // @author Horacio
4. import java.util.Scanner;
5. public class MediaEdadHijos {
6. public static void main(String[] args) {
7. Scanner entrada = new Scanner(System.in); //Se crea un objeto Scanner para obtener datos de la consola
8. int nFamilias, nHijos = 0, sumaEdades = 0; //Se declaran las variables para el número de familias, hijos y la suma de edades
9. double media; //Se declara la variable para la media de edad de los hijos
10. System.out.print("Ingrese la cantidad de familias: ");
11. nFamilias = entrada.nextInt(); //Se obtiene la cantidad de familias
12. for(int i = 1; i <= nFamilias; i++) { //Se inicia un ciclo para cada familia
13. System.out.print("Ingrese la cantidad de hijos de la familia " + i + ": ");
14. nHijos = entrada.nextInt(); //Se obtiene la cantidad de hijos de la familia actual
15. for(int j = 1; j <= nHijos; j++) { //Se inicia un ciclo para cada hijo de la familia actual
16. System.out.print("Ingrese la edad del hijo " + j + " de la familia " + i + ": ");
17. sumaEdades += entrada.nextInt(); //Se va sumando la edad de cada hijo a la variable sumaEdades
18. }
19. }
20. media = (double) sumaEdades / (nFamilias \* nHijos); //Se calcula la media de edad de los hijos de todas las familias
21. System.out.println("La media de edad de los hijos de todas las familias es: " + media);
22. }
23. }

El programa comienza importando la clase Scanner que se utilizará para leer los datos de entrada proporcionados por el usuario. Luego, se declaran las variables necesarias para almacenar la cantidad de familias, la cantidad de hijos y la suma de edades de los hijos. También se declara la variable para la media de edad de los hijos.

El programa solicita al usuario que ingrese la cantidad de familias que se van a calcular, utilizando el método nextInt() de la clase Scanner. A continuación, comienza un ciclo for que se ejecuta una vez para cada familia. Dentro de este ciclo, se solicita al usuario que ingrese la cantidad de hijos de la familia.