ITID433
Programación Avanzada
Facultad de Ingeniería Eléctrica y Electrónica

Práctica 01

Gestión de datos en Java

Objetivos:

Esta segunda práctica de la asignatura persigue los siguientes objetivos:

- Practicar la declaración, inicialización y uso de variables de diferentes tipos de datos.
- Reforzar la comprensión sobre cómo realizar operaciones básicas con diferentes tipos de datos.

Instrucciones y Actividades:

1. Marco Teórico

Variables

Una variable es un espacio de memoria donde se almacenará un dato.

Una variable es un contenedor que se utiliza para almacenar y manipular los datos en un programa.

Una variable tiene un tipo de dato que define qué tipo de valor puede contener y un nombre que se utiliza para hacer referencia a ella en el código.

El valor de la variable puede modificase, pero su tipo de dato no puede cambiarse una vez declarado.

Tipos de variables

En Java, las variables se pueden agrupar en tres: variables locales, variables de instancia y variables de clase.

Las variables locales se declaran dentro de un método y solo son accesibles dentro de ese método.

Las variables de instancia o variables de objeto conforman el estado de un objeto y se declaran dentro de la clase.

Las variables de clase son compartidas por todas las instancias de una clase y se declaran utilizando la palabra clave `static`.

Tipos de datos

Los tipos de datos se dividen en dos categorías principales: tipos de datos primitivos y objetos.

Datos primitivos

Los datos primitivos son tipos de datos básicos que representan valores simples y se almacenan directamente en la memoria.

byte: Representa un tipo de dato de 8 bits con signo. Puede almacenar valores numéricos en el rango de -128 a 127, incluyendo ambos extremos.

short: Este tipo de dato utiliza 16 bits con signo y puede almacenar valores numéricos en el rango de -32,768 a 32,767.

int: Es un tipo de dato de 32 bits con signo utilizado para almacenar valores numéricos. Su rango va desde -2,147,483,648 (-2^31) hasta 2,147,483,647 (2^31 - 1). Es el tipo de dato más comúnmente utilizado para representar números enteros.

long: Este tipo de dato utiliza 64 bits con signo y puede almacenar valores numéricos en el rango de -9,223,372,036,854,775,808 (-2^63) a 9,223,372,036,854,775,807 (2^63 - 1). Se utiliza cuando se necesitan números enteros muy grandes.

float: Es un tipo de dato diseñado para almacenar números en coma flotante con precisión simple de 32 bits. Se utiliza cuando se requieren números decimales con un grado de precisión adecuado para muchas aplicaciones.

double: Este tipo de dato almacena números en coma flotante con doble precisión de 64 bits, lo que proporciona una mayor precisión. Se usa en aplicaciones que requieren una alta precisión en cálculos numéricos.

boolean: Sirve para definir tipos de datos booleanos que pueden tener solo dos valores: true o false.

char: Es un tipo de datos que representa un carácter Unicode sencillo de 16 bits (UETF-16). Se utiliza para almacenar caracteres individuales, como letras o símbolos en diferentes lenguajes y conjuntos de caracteres.

Objetos

Son tipos de datos más complejos almacenados en memoria. Estos objetos pueden ser instancias de clases personalizadas o clases predefinidas en Java, como `String`.

Java también tiene tipos de datos de referencia especiales, como `null`, que representa la ausencia de un objeto, y `void`, que se utiliza en métodos que no devuelven ningún valor.

Declaración de variables

Para declarar una variable en Java, se utiliza la siguiente sintaxis:

```
[tipo] [nombreDeVariable];
```

Asignación de variables

Para inicializar o asignar un valor a una variable en Java, se utiliza la siguiente sintaxis:

```
[nombreDeVariable] = valor;
```

Definición de variables

Para definir una variable en Java, se utiliza la siguiente sintaxis:

```
[tipo] [nombreDeVariable] = valor;
```

Convertir tipos de datos

El casting de variables en Java es una técnica que se utiliza para convertir una variable de un tipo de datos en otro tipo de datos compatible. Por ejemplo, se usa cuando deseas asignar un valor de un tipo de dato a una variable de otro tipo o cuando realizas operaciones aritméticas con tipos de datos diferentes.

Java proporciona dos tipos de casting: casting implícito (también conocido como conversión automática) y casting explícito (o conversión manual).

Casting Implícito (Conversión Automática)

El casting implícito se produce cuando Java convierte automáticamente un tipo de dato en otro sin requerir una intervención explícita del programador. Esto ocurre cuando se realiza una conversión de un tipo de datos más pequeño a uno más grande, evitando la pérdida de datos.

```
int numeroEntero = 5;
double numeroDecimal = numeroEntero;
```

Casting Explícito (Conversión Manual)

El casting explícito se utiliza cuando deseas convertir un tipo de datos en otro tipo de datos que no es compatible automáticamente. Esto requiere que el programador indique explícitamente la conversión utilizando paréntesis y el tipo de datos al que se desea convertir.

```
double numeroDecimal = 3.14;
int numeroEntero = (int) numeroDecimal;
```

Es importante tener en cuenta que la conversión explícita puede implicar una pérdida de información (en el ejemplo se perderán los decimales).

Es importante tener cuidado al realizar casting explícito, ya que puede conducir a resultados inesperados o pérdida de datos. Siempre verifica si la conversión es segura antes de realizarla.

Casting en Operaciones Aritméticas

El casting también se puede aplicar en operaciones aritméticas cuando se trabaja con tipos de datos diferentes. En este caso, Java realizará automáticamente un casting implícito para que las operaciones se realicen sin errores.

```
int numeroEntero = 5;

double resultado = numeroEntero / 2;
```

En este ejemplo, el resultado de la división se almacena en la variable `resultado` como un número decimal debido al casting implícito.

Las reglas de casting son:

1) Si cualquier operando es `double` todos se convertirán en `double`.
2) Si cualquier operando es `float` y no hay ningún `double` todos se convertirán a `float`.
3) Si cualquier operando es `long` y no hay datos reales todos se convertirán en `long`.
4) Si cualquier operando es `int` y no hay datos reales ni `long` se convertirán en `int`.
5) En cualquier otro caso el resultado será `int`.

Arrays

Los arrays son colecciones de datos de un mismo tipo. En Java los arrays son un objeto.

Un array es una estructura de datos en la que a cada elemento le corresponde una posición identificada por uno o más índices numéricos enteros.

Los elementos de un array se empiezan a numerar en la posición 0, y permiten gestionar desde una sola variable múltiples datos del mismo tipo.

Enum

Los tipos de datos enumerados son un tipo de dato definido por el programador. En su definición el programador debe indicar un conjunto de valores finitos sobre los cuales

las variables de tipo enumeración deberán tomar valores. La principal funcionalidad de los tipos de datos enumerados es incrementar la legibilidad del programa. La sintaxis es:

```
enum nombre { VALOR1,VALOR2,.. }
```

Secuencias de Escape

Son secuencias especiales que se usan para representar caracteres que no puede ser escritos. Entre los principales están:

- Retroceso (*backspace*) '\b'
- Nueva línea '\n'
- Retorno '\r'
- Tabulación '\t'
- Barra invertida (*backslash*) '\\'
- Comilla simple '\''
- Comilla doble '\"'
- Símbolo de pregunta '\?'
- Carácter centinela, carácter nulo '\0'

2. Uso de tipos de datos

1. Crea una clase llamado `UsoDatos`. Agrega el método `main` y dentro del método `main` realiza los siguientes pasos.
   ```
   public class UsoDatos {
      public static main(String[] args) {
         // aqui va el código
      }
   }
   ```

2. Declara una variable de tipo `int` llamada `edadE1`.
   ```
   int edadE1;
   ```

3. Asigna tu edad actual a la variable `edadE1` (actualiza el código con tu edad).
   ```
   edadE1 = 20;
   ```

4. Define una variable de tipo `int` llamada `edadE2` y asígnale la edad de tu compañero.

5. Define una variable de tipo `double` llamada `alturaE1` y asígnale tu altura en metros (actualiza el código con tu altura).
   ```
   double alturaE1 = 1.65;
   ```

6. Define una variable de tipo `double` llamada `alturaE2` y asígnale la altura en metros de tu compañero.

7. Define una variable de tipo `long` llamada `pesoE1` y asígnale tu peso en kg (actualiza el código con tu peso).
   ```
   long pesE1 = 82L;
   ```

8. Define una variable de tipo `long` llamada `pesoE2` y asígnale el peso en kg de tu compañero.

9. Define una variable de tipo `char` llamada `letraNombreE1` y asígnale la inicial de tu nombre (actualiza el código con tu inicial)
   ```
   char letraNombreE1 = 'D';
   ```

10. Define una variable de tipo `char` llamada `letraNombreE2` y asígnale la inicial del nombre de tu compañero.

11. Define una variable de tipo `boolean` llamada `esEstudianteE1` y asígnale `true`.
    ```
    boolean esEstudianteE1 = true;
    ```

12. Define una variable de tipo `boolean` llamada `esEstudianteE2` y asígnale `false`.

13. Define una variable de tipo `String` llamada `nombreCompletoE1` y asígnale tu nombre completo. (actualiza el código con tu nombre)
    ```
    String nombreCompletoE1 = "Juan Andrés Cruz Real";
    ```

14. Define una variable de tipo `String` llamada `nombreCompletoE2` y asígnale tu nombre completo.

15. Crea una expresión booleana que evalúe si tu edad es mayor que la de tu compañero y almacena el resultado en una variable llamada `esMayor`.
    ```
    bool esMayor = edadE1 > edadE2;
    ```

16. Crea una expresión booleana que evalúe si tu edad es menor o igual que la de tu compañero y almacena el resultado en una variable llamada `esMenor`.

17. Escribe una expresión que determine si tu altura es mayor que la de tu compañero y almacene el resultado en una variable llamada `esTallaAlta`.
    ```
    bool esTallaAlta = alturaE1 > alturaE2;
    ```

18. Crea una expresión booleana que evalúe si tu altura es menor o igual que la de tu compañero y almacena el resultado en una variable llamada `esTallaBaja`.

19. Concatena tu inicial, edad, altura y peso en una cadena de texto, para separar cada dato usa tabulación. Al final de la cadena coloca el retorno. Guárdala en una variable llamada `informacionPersonalE1`.

```
String informacionPersonalE1 = letraNombreE1 + "\t" + edadE1 + "\t" + alturaE1 + "\t" + pesoE1 + "\n";
```

20. Concatena la inicial, edad y altura de tu compañero en una cadena de texto y guárdala en una variable llamada `informacionPersonalE2`. Para la concatenación usa el símbolo de $ para separar cada dato, al final de la cadena coloca el retorno.

21. Realiza las siguientes operaciones aritméticas e imprime los resultados:
    a. Suma las dos edades.
    ```
    System.out.println("La suma de edades es: " + (edadE1 + edadE2));
    ```
    b. Resta las dos alturas.
    ```
    System.out.println("La resta de alturas es: " + (alturaE1 – alturaE2));
    ```
    c. Multiplica los dos pesos.
    ```
    System.out.println("La multiplicación de los pesos es: " + (pesoE1 * pesoE2));
    ```
    d. Divide el resultado de la suma de las dos edades para el resultado de la multiplicación de los dos pesos.
    ```
    System.out.println("La división de edades para pesos: " + ((edadE1 + edadE2)/(pesoE1 * pesoE2)));
    ```
    e. Encuentra el residuo del resultado de la suma de las edades para la resta de las alturas.
    ```
    System.out.println("El residuo de edades para alturas: " + ((edadE1 + edadE2)%(alturaE1 – alturaE2)));
    ```
    f. Divide tu peso para tu altura elevada al cuadrado y obtén el IMC y almacénalo en una variable de tipo `double`.
    ```
    double imcE1 = pesoE1 / (alturaE1 * alturaE1);
    ```
    g. Realiza el cálculo del IMC con la información de tu compañero.

22. Presenta mensajes informativos para la información de **cada** estudiante, de la siguiente forma:
Estudiante 1:
Nombre completo: valor
Inicial: valor
Edad: valor años
Peso: valor kg
Altura: valor m
IMC: valor kg/m2
¿Es estudiante? valor
¿Es el más alto? valor
¿Es mayor? valor
Información concatenada: valor

```
System.out.println("Estudiante 1:");
System.out.println("Nombre completo: " + nombreCompletoE1);
System.out.println("Inicial: " + letraInicialE1);
System.out.println("Edad: " + edadE1 + " años");
System.out.println("Peso: " + pesoE1 + " kg");
System.out.println("Altura: " + alturaE1 + " m");
System.out.println("IMC: " + imcE1 + "kg/m^2");
System.out.println("¿Es estudiante? " + esEstudianteE1);
System.out.println("¿Es el más alto? " + esTallaAlta);
System.out.println("¿Es mayor? " + esMayor);
```

23. Declara un arreglo llamado `numeros` e inicialízalo con elementos del 1 a 5. Presenta el tercer y último elemento, e imprime el número de elementos del arreglo.

```
int[] numeros = {1, 2, 3, 4, 5};
System.out.println("El tercer elemento del arreglo es: " + numeros[2]);
System.out.println("El quinto elemento del arreglo es: " + numeros[4]);
System.out.println("El número de elementos del arreglo es: " + numeros.length);
```

24. Declara una enumeración llamada `DiasSemana` y agrega a la lista los 7 días de la semana (LUNES, …, DOMING). Crea una variable denominada `feriado` de tipo `DiasSemana` e inicialé con el valor miércoles. Imprime en la pantalla el valor de la variable feriado. Es importante que la enumeración esté fuera del método main.

```
// coloca esto fuera del main, puede estar en la
// siguiente línea en la que se define la clase
```

```
enum DiasSemana { LUNES, MARTES, MIERCOLES, JUEVES,
VIERNES, SABADO, DOMINGO };
// esto va en el main
DiasSemana feriado = DiasSemana. MIERCOLES;
System.out.println("El feriado debía ser el día " +
feriado);
```

25. Utiliza el método `Math.pow()` para calcular el cuadrado de tu altura, almacénalo en una variable y preséntalo en pantalla.

```
double cuadrado = Math.pow(alturaE1, 2);
System.out.println("El cuadrado de la altura es: " +
cuadrado);
```

26. Utiliza el método `Math.swrt()` para calcular la raíz cuadrada del peso de compañero y preséntalo.

```
double raiz = Math.sqrt(pesoE2);
System.out.println("La raíz del peso es: " + raiz);
```

27. Trata de almacenar el valor 255 en una variable de tipo byte llamada `miByte`.

```
byte miByte = 255;
System.out.println(miByte);
```

Si agregas el código anterior no funciona, puesto que estás tratando de guardar un `int` en un `byte`, recuerda que si no se indica lo contrario un literal numérico entero es `int`.

28. Trata de almacenar el valor 255 en una variable de tipo byte llamada `miByte` usando casting.

```
byte miByte = (byte) 255;
System.out.println(miByte);
```

Explica por qué aparece el valor -1 en la sección Resultados del encabezado.

29. Agrega el siguiente código que muestra las diferentes entre realizar el incremento de forma previa (++x) o de forma posterior (x++). Explica en la sección Resultados del encabezado si se obtiene el mismo resultado o no, en cada ejemplo.

```
int x = 0;
int y = 0;
System.out.println("x es " + x + ", y es " + y );
x++;
System.out.println("x++ resulta en " + x);
++x;
System.out.println("++x resulta en " + x);
System.out.println("Reinicio del valor de x");
x = 0;
System.out.println("———————");
y = x++;
```

```
System.out.println("y = x++ (posterior) resulta en:");
System.out.println("x es " + x);
System.out.println("y es " + y);
System.out.println("———————");
y = ++x;
System.out.println("y = ++x (previo) resulta en:");
System.out.println("x es " + x);
System.out.println("y es " + y);
System.out.println("———————");
```

30. Agrega el siguiente código que muestra el operador + aplicado a diferentes casos (enteros y cadenas; y cadenas y cadenas). Explica en la sección Resultados del encabezado qué se obtuvo y por qué.

```
System.out.println(2 + 3 + "prueba");
System.out.println("prueba" + 2 + 3);
```

3. Entregables

Para esta práctica debes realizar lo siguiente:

1. Completa el código solicitado, completa la información faltante de tu compañero.
2. Incluye en la aplicación el ingreso de datos desde la consola y pide al usuario que ingrese su nombre, edad, altura, pero y luego imprima un mensaje en el que concatenes la información, así también calcula el IMC y preséntalo.
3. Ubica la carpeta del proyecto generada en Eclipse, elimina la carpeta `bin`, y procede a comprimir la carpeta del proyecto. Esta carpeta comprimida (.ZIP), con el nombre cumpliendo el formato indicado en clase, es la que debes subir en el Práctica 01.

4. Realizado por:

David Mejía N.