1) Indica si los siguientes identificadores de variables en Java serían válidos, justifica tu respuesta.

a) double: No es válido por ser palabra reservada en Java.
b) *horaactual: No es válido porque no se puede comenzar un identificador con '/'*
c) $hora: Válido, aunque creo que no se usa mucho.
d) _hora: Válido.
e) 5hora: No es válido comenzar con un dígito.

2) Explica razonadamente los tipos de datos más adecuados para almacenar los siguientes elementos.

a) Número de teléfono : O int o String si tiene caracteres tipo +, guiones...
b) Precio de un artículo: double por los decimales.
c) Estado de un pedido: String, aunque como estará en un rango de valores como 'Recibido' o 'Devolución iniciada' se podría usar enum.
d) Edad de un cliente: int porque es un número y no se suelen usar decimales.
e) Constante de gravitación universal (G) : double como constante.

3) Señala y describe el tipo de los literales contenidos en las siguientes sentencias.

```
double r = 2.5;
System.out.println("El volumen de la esfera es: "
+ 4 / 3 * Math.PI * Math.pow(r, 3.0f));
```

Veo como literales el 2.5 de tipo double como vemos tipado, Math.PI es una constante de tipo double, un string ("El volumen de la esfera es:"), un float en 3.0f y en 4/3 veo dos literales de tipo int, 4 y 3.

4) Identifica los operadores del siguiente fragmento de código e indica sus tipos.

```
public static boolean esPrimo(int numero) {
   if (numero <= 1) { // Operador relacional (<=)
      return false;
   }
   for (int i = 2; i <= Math.sqrt(numero); i++) { // Operador de asignación (=), relacional (<=) y aritmético (++ y /)
      if (numero % i == 0) { // Operadores aritmético (%) y relacional (==)
         return false;
      }
   }
   return true;
}
```

- <=: Operador relacional (comparación).
- %: Operador aritmético (módulo).
- ==: Operador relacional (igualdad).
- = : Operador de asignación.
- ++: Operador aritmético de incremento.

5) Teniendo en cuenta que var1, var2, var3 son variables de tipo boolean y están inicializadas a los siguientes valores: var1 = true, var2 = true y var3 = false. Las variables x, y, z son variables numéricas enteras con valores: x = 5, y = -8 y z = 10. Indica el resultado de las siguientes expresiones.

a) var1 || var2 && var3
b) (var2 || !var1 || !var3) && var1
c) (var1 || var3) && (var2 && !var1)
d) (x + z == 15) && (y != 2)
e) (x > 3 || y > 3) && z < -3

a) var1 || var2 && var3: True porque var2 y var3 es false. Luego var1 or false es true.

- **b)** (var2 || !var1 || !var3) && var1. True porque del paréntesis de las negaciones acaba habiendo true y var1 es true. True and true es true.
- **c)** (var1 || var3) && (var2 && !var1). False porque el paréntesis de la derecha es false.
- **d)** (x + z == 15) && (y != 2). True porque x + z es 15 e la y no es 2 sino -8. true and true es true.
- **e)** (x > 3 || y > 3) && z < -3. paréntesis da true. Z no es menor que -3. true and false es false.

6) Explica el valor almacenado en cada una de las siguientes variables.

a) int n = (int) 'A';
b) float f = 1.25f * (float)Math.E
c) char c = (char) 65;
d) char d = (char) ((int) 'A' + 1);
e) string s = "";

a) int n = (int) 'A'; Valor de n es 65 (valor Unicode de A).

- **b)** float f = 1.25f * (float)Math.E Se multiplica 1.25 por el valor de e, que creo que es el número de Euler, o sea 2.71...
- **c)** char c = (char) 65;:Valor de c es A. (La inversa del subejercicio a)
- **d)** char d = (char) ((int) 'A' + 1); Valor de d es B porque si 65+1=66, y 66 en Unicode es B
- **e)** string s = ""; s es un String vacío, sin caracteres