Taller 4 JAVA parte 2 ejercicio 2

1) Cuando se llaman funciones sobrecargadas que reciben diferentes tipos de parámetros, y el tipo exacto del dato no coincide con ninguna función, Java promueve el tipo al nivel superior más cercano en la jerarquía de datos. La jerarquía presentada es clave para entender qué función se ejecuta en cada caso:

Jerarquía de tipos en Java (relevante para este caso):

byte → short → int → long → float → double

char → int

Integer (objeto) → int (desenvolvimiento automático)

a. **char: Entra a int: 103**
   El carácter 'g' tiene el valor Unicode de 103, que es interpretado como un número entero. Como no hay un método específico para char, se usa el método con parámetro int.
b. **short: Entra a int: 2**
   Los valores short se promueven automáticamente a int cuando no hay un método específico para short. Por ello, se llama al método con parámetro int.
c. **byte: Entra a int: 1**
   Similar al caso de short, los valores de tipo byte se promueven automáticamente a int, invocando el método correspondiente.
d. **long: Entra a double: 9.99999999E8**
   El tipo long no tiene un método específico, pero como long está por encima de int en la jerarquía, se promueve al siguiente tipo superior, que es double. El valor 999999999 se representa en notación científica como 9.99999999E8.
e. **integer: Entra a int: 51232**
   Los valores Integer (clase envolvente) se convierten automáticamente a int por autounboxing, y el método con parámetro int es llamado directamente.
f. **double: Entra a double: 12.4**
   Los valores double ya coinciden directamente con el método que acepta double como parámetro, por lo que este es utilizado sin ninguna promoción adicional.
g. **float: Entra a double: 5.650000095367432**

El tipo float no tiene un método dedicado, pero se promueve al siguiente tipo en la jerarquía, que es double. Al hacerlo, se da una mayor precisión al valor representado.

2) A) Lo mostrado en pantalla varía: ahora hay dos tipos de datos que no pasan a través de la función cuyo parámetro es un "int", sino que lo hacen por la que recibe "shorts", al ser esta más específica o equivalente:

   B) Lo que aparece en pantalla se modifica: ahora hay dos tipos de datos que no pasan por la función con parámetro “double”, sino por la que acepta “float”, ya que esta es más específica o equivalente. En este caso, “long” es superior a “int” pero inferior a “double”:
   long: Procesado por float: 1.0E9
   float: Procesado por float: 5.65

   C) Se produce un **ERROR** porque no existe un tipo superior a double, lo que significa que la función función(d) no puede invocar ningún método que sea compatible con el tipo double. Dado que no hay un tipo que exceda la jerarquía de double, el compilador no encuentra un método válido para procesar el valor pasado y genera un error.
   Esto refleja la ausencia de una ruta alternativa en la resolución de métodos, ya que double es el tipo más preciso para valores de punto flotante en este contexto.

   D) n este caso, todos los tipos de datos pasan a través de la función que acepta parámetros de tipo double. Esto se debe a que double puede representar cualquier tipo numérico compatible mediante conversión implícita, ajustando su precisión según el tipo original. El resultado en pantalla muestra cada tipo convertido a double y su respectiva representación de punto flotante con la precisión asociada:
   char : Entra a double: 103.0
   short : Entra a double: 2.0
   byte : Entra a double: 1.0
   long : Entra a double: 9.99999999E8
   integer : Entra a double: 51232.0
   double : Entra a double: 12.4
   float : Entra a double: 5.650000095367432