Taller 4

Ejercicio 1:

AR//: El sistema que se encarga de destruir los objetos (Garbage Collection-GC), se encargara de eliminar y limpiar la memoria de los objetos que no poseen una referencia incluyendo los objetos de tipo persona, sin embargo este no los eliminara inmediatamente estos pierdan su referencia, si no cuando el JVM determina que es necesario.

BR//: Para conocer el nombre del dueno de la variable auto de la linea 11 de metodo main es necesario llamarlo de la siguiente manera auto.getDueno.getNombre()

CR//: La manera para poder agregar un dueno al vehiculo de la linea 13 del metodo main es de la siguiente manera auto2.setDueno(Nombre o referencia del dueno que desea asignarse al objeto).

DR//: Para obtener el valor del atributo de la variable auto2 de la linea 13 del metodo main, es necesario acceder de la siguiente manera auto2.getMotor().getVelocidadMaxima()

ER//: El metodo main imprimira por consola lo siguiente:

Matando a: Alejandro

Matando a: Jaime

Matando a: Alexander

Matando a: Santiago

Soy Santiago

FR//: Al momento de ejecutar System.out.println(personas[1]); despues de la linea 16, se imprimira Soy Daniel debido a que la referencia que tenia anteriormente personas[1] fue cambiada a la misma referencia a la que apunta personas[2] esto significa que ahora apunta a “Daniel” perdiendo la referencia de “Jaime”.

GR//: Para lograr que se muestren la placa y el dueño del vehículo al ejecutar el código, puedes usar los métodos getters definidos en la clase Vehiculo. El código quedaría de la siguiente forma:

System.out.println("#Placa: " + auto2.getPlaca() + " Dueño: " + auto2.getDueno());

Esto imprimirá el número de la placa y el nombre del dueño. Si el atributo dueno aún no ha sido inicializado, el resultado será "null" en lugar del nombre del propietario.

HR//: En este caso, usando la variable auto, accedemos a su dueño con getDueno(), y luego asignamos como su mejor amigo al tercer elemento del array personas (índice 2). El código quedaría así:

auto.getDueno().setMejorAmigo(personas[2]);

Este código asocia el tercer elemento de la lista personas como el mejor amigo del dueño del vehículo.

Ejercicio 2:

AR// Tengamos en cuenta que: Cuando no existe el método con el mismo tipo de dato, se asocia al método que tenga el parámetro del tipo inmediatamente superior. Y la jerarquía es generalmente de esta manera siendo el boolean con menos bits hasta el double con más de estos:

- boolean: 1 bit (true o false).
- byte: 8 bits
- char: 16 bits (caracteres Unicode).
- short: 16 bits.
- int: 32 bits.
- float: 32 bits, precisión simple.
- Long: 64 bits.
- double: 64 bits, precisión doble.

Teniendo ya en cuenta que solo hay dos funciones similares, una recibiendo parámetros tipo “int” y el otro tipo “double” por lo tanto si se ingresa un tipo de dato distinto se usara el método que tenga el parámetro inmediatamente superior por lo tanto quienes usan el método con “int” en este caso son los valores tipo: “char”, “short”, “byte” e “integer”; quienes usan el método con “double” en este caso son los valores tipo: “double”, “long” y “float”.

Se imprime lo siguiente:

char: Entra a int: 103 // (imprime 103 porque el carácter ´g´ en Unicode tiene valor 103 como numero entero).

short: Entra a int: 2 // (por jerarquía).

byte: Entra a int: 1 // (por jerarquía).

long: Entra a double: 9.99999999E8 // (puede representarse en notación científica. En este caso: 999999999=9.99999999×10^8).

integer: Entra a int: 51232 // (por jerarquía/casi que el mismo tipo).

double: Entra a double: 12.4 // (Double ingresa a su misma función con parámetro double).

float: Entra a double: 5.650000095367432 // (por jerarquía pasa a la función con parámetro double y se le da mayor precisión al valor ingresado).

B. • Active la función que recibe un short.

R// Lo impreso en pantalla cambia, ahora hay dos tipos de datos que ya no ingresan por la función que tiene como parámetro un "int" sino por la que recibe shorts al ser la superior más cerca o el mismo:

"short: Entra a short: 2

byte: Entra a short: 1"

• Active la función que recibe un float.

R// Lo impreso en pantalla cambia, ahora hay dos tipos de datos que ya no ingresan por la función que tiene como parámetro un "double" sino por la que recibe float al ser la superior más cerca o el mismo, en este caso siendo long superior a "int" pero inferior a "double":

"long: Entra a float: 1.0E9

float: Entra a float: 5.65"

• Comente la función que recibe un double y active la que recibe un float.

R// Lanza ERROR, ya que no hay otro tipo superior al double por lo que la "función(d)" crea el error al no tomar ni un método posible para el tipo "double".

• Comente todas las funciones, excepto la que recibe un double.

R// Ahora pasa que todos los tipos pasan por la función que recibe parámetros double y se imprimen de la manera que las convierte a doubles y su respectiva precisión. Se muestra lo siguiente:

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