Taller JAVA programación orientada a objetos

Santiago Barrientos Medina

Universidad Nacional de Colombia

Programación orientada a objetos

Jaime Alberto Guzmán Luna

Grupo 2

Facultad de Minas

Ingeniería de sistemas e informática

EJERCICIO 1

A) Si se pierde la referencia de un objeto de tipo persona el garbage colector vendría por el y se lo llevaría pero como tenemos un finalize en la clase , cuando el garbage colector venga por el se va a ejecutar el método finalize que dice “Matando a “ y se le concatena el nombre del objeto que vamos a eliminar. Aunque esto ultimo ocurriría si el garbage colector alcanza a llegar mientras se ejecuta el programa

B) Podríamos conocer el nombre del dueño de la variable auto de la línea 11 de la siguiente manera :

PRIMERA FORMA:

System.out.println(Auto.getdueno().getNombre());

SEGUNDA FORMA:

Persona o = auto.getdueno();

System.out.println(o.getNombre());

C) Podríamos asignarle un dueño al vehículo de la línea 13 del método main de la siguiente manera:

Auto2.setDueno(personas[1(o el numero que deseemos)])

D) Mi propuesta para obtener la velocidad máxima del motor del auto 2 es :

A través de auto2 acceder al motor con el get y luego cuando nos retornen el motor, con este retorno, acceder al método get de la velocidad máxima del motor

```
System.out.println(auto2.getMotor().getVelocidadMaxima());
```

E) Lo que se imprimiría al ejecutar el método main por consola seria:

Matando a : Alejandro

Matando a : Jaime

Matando a : Santiago

Soy Santiago

F) Si se agrega esa línea de código y después se ejecuta el main lo que pasaría seria que al momento de intentar imprimir el objeto, se llamaría al tostring que imprimiría Soy Daniel, pero, ¿por qué soy Daniel y no ,soy Jaime? . esto porque en la línea anterior, el apuntador de personas en [1] se le asigno a personas [2] y entonces empezó a apuntar a Jaime asi que ahora el objeto en personas en [2] con el nombre Daniel tiene dos apuntadores personas [1] y personas [2]

G) La modificación que debe hacerse en el código es la siguiente

En la clase vehiculo tenemos que crear un tostring que retorne el atributo placa de la instancia y que retorne el dueño, como dueño ya tiene to string , solo seria necesario hacer esto en la clase Vehiculo :

```
public String toString() {
	return "El vehiculo" +placa+"es de " + this.dueno;
```

H) Seria asi:

```
auto.getDueno().setMejorAmigo(personas[1]);
```

EJERCICIO 2

A) 1). Char: entra a int: 103 . al no haber un método para el dato de tipo char , el dato se asocia al método que reciba un dato superior en la jerarquía, pero, ¿Por qué 103?, pues porque en el ascii en carácter g tiene asignado el numero 103 asi que al pasarlo por el método que recibe un int se le asigna este numero

2). Short : entra a int: 2. Al no haber un método que reciba un short como parámetro, se le asocia un método que reciba un dato inmediatamente superior en la

jerarquía , el método que recibe el dato mas cercano en la jerarquía es el de int , ya que int es el mas cercano a short en esta misma

3).byte: entra a int:1. Esto porque no tenemos un método que reciba como parámetro un dato de tipo byte , lo que por ende , significa que tenemos que ir a la tabla de jerarquía y buscar en los inmediatamente superiores cual esta en alguno de los métodos del código como variable a recibir , en este caso hay un método que recibe un int y un double pero como el int esta mas cercano en la tabla de jerarquía, por eso lo pasamos por el método que recibe el int

4).long : entra a double:9999999. Esto porque no tenemos un método que nos reciba un long . entonces al mirar en el código , nos encontramos conque hay uno que recibe un int pero el int esta por debajo del long en la tabla de jerarquía, asi que no sirve, luego nos encontramos con un método que recibe un double y el double si esta por encima del long, asi que lo pasamos por el método que recibe un double

5).integer: entra a int:51232. En este realmente no hay mucho que decir porque si hay un método que reciba el int, asi que simplemente se le pasa , no hay problema aquí

6).double: entra a double : 12.4. como en el caso anterior, tiene el método que recibe su tipo de dato asi que no hay problema y se le entrega normalmente al método que recibe su tipo de dato

7).float : entra a double: 5.65. al no haber método que reciba un float , tenemos que irnos a mirar los métodos que hay que tipo de datos reciben , encontramos que hay dos , que reciben un int y otro que recibe un double. Al mirar la tabla de jerarquía, vemos que el del int no nos sirve porque no nos podemos devolver en la tabla, pero mas arriba del float esta el double asi que a este si se lo podemos entregar o asociar

B) CAMBIO 1:

Con este cambio, la impresión por pantalla quedaría asi:

1). Char: entra a short: 103 . al no haber un método para el dato de tipo char , el dato se asocia al método que reciba un dato superior en la jerarquía, pero, ¿Por qué 103?, pues porque en el ascii en carácter g tiene asignado el numero 103 asi que al pasarlo por el método que recibe un int se le asigna este numero

2). Short : entra a short: 2. Al ya haber un método que reciba un short, no es necesario hacer el ejercicio de buscar al método que se le puede asociar sino que continuamos con normalidad y lo pasamos al método que recibe un short

3).byte: entra a short:1. Al activar el método que recibe un short, al momento de irnos a la tabla de jerarquía a buscar a cual método le podemos asociar el byte porque no hay un método que reciba el byte, nos encontramos que como ya hay un método que recibe un short y este esta mas cercano en la jerarquía que el int, le entregamos el dato de tipo byte al método que recibe al short

4).long : entra a double:9999999. Esto porque no tenemos un método que nos reciba un long . entonces al mirar en el código , nos encontramos conque hay uno que recibe un int pero el int esta por debajo del long en la tabla de jerarquía, asi que no sirve, luego nos encontramos con un método que recibe un double y el double si esta por encima del long, asi que lo pasamos por el método que recibe un double

5).integer: entra a int:51232. En este realmente no hay mucho que decir porque si hay un método que reciba el int, asi que simplemente se le pasa , no hay problema aquí

6).double: entra a double : 12.4. como en el caso anterior, tiene el método que recibe su tipo de dato asi que no hay problema y se le entrega normalmente al método que recibe su tipo de dato

7).float : entra a double: 5.65. al no haber método que reciba un float , tenemos que irnos a mirar los métodos que hay que tipo de datos reciben , encontramos que hay dos , que reciben un int y otro que recibe un double. Al mirar la tabla de jerarquía, vemos que el del int no nos sirve porque no nos podemos devolver en la tabla, pero mas arriba del float esta el double asi que a este si se lo podemos entregar o asociar

CAMBIO 2: teniendo en cuenta este cambio y el anterior

1). Char: entra a short: 103 . al no haber un método para el dato de tipo char , el dato se asocia al método que reciba un dato superior en la jerarquía, pero, ¿Por qué 103?, pues porque en el ascii en carácter g tiene asignado el numero 103 asi que al pasarlo por el método que recibe un int se le asigna este numero

2). Short : entra a short: 2. Al ya haber un método que reciba un short, no es necesario hacer el ejercicio de buscar al método que se le puede asociar sino que continuamos con normalidad y lo pasamos al método que recibe un short

3).byte: entra a short:1. Al activar el método que recibe un short, al momento de irnos a la tabla de jerarquía a buscar a cual método le podemos asociar el byte porque no hay un método que reciba el byte, nos encontramos que como ya hay un método que recibe un short y este esta mas cercano en la jerarquía que el int, le entregamos el dato de tipo byte al método que recibe al short

4).long : entra a float:9999999. Al activar el método que recibe un float, sigue sin haber un método que reciba un long pero a diferencia de en el primer punto, ya no se le asigna el método que recibe un double, esto porque en la tabla de jerarquía esta mas arriba pero mas cercano el tipo de dato float, asi que ya no le asoaciamos el método que recibe un double sino que le asociamos el de float

5).integer: entra a int:51232. En este realmente no hay mucho que decir porque si hay un método que reciba el int, asi que simplemente se le pasa , no hay problema aquí

6).double: entra a double : 12.4. como en el caso anterior, tiene el método que recibe su tipo de dato asi que no hay problema y se le entrega normalmente al método que recibe su tipo de dato

7).float : entra a float: 5.65. al ya haber un método que reciba un float, no es necesario asociar los float al método double o irnos a buscar a la tabla de la jerarquía , como tiene un método que recibe su tipo de dato, se sigue el curso normal y se le asocia el que recibe float

CAMBIO NUMERO 3 teniendo en cuenta los dos anteriores y este

Aquí el código mostraría error al intentar compilarse. Pero , ¿Por qué?. La respuesta radica en la tabla de las jerarquías , cuando nosotros no tenemos un método que reciba cierto tipo de dato pues buscamos en la tabla quien es el inmediatamente superior a el que este en el código, pero si uno se va a la tabla y busca quien hay por encima del double pues no hay nadie , el double es el tipo de dato que mas rango tiene y no puede ser almacenado por otro tipo de dato sin un casteo explicito, casteo que no esta en el código por lo que el código explota

CAMBIO NUMERO 4 :

1). Char: entra a double: 103 . al no haber un método para el dato de tipo char , el dato se asocia al método que reciba un dato superior en la jerarquía, pero, ¿Por qué 103?, pues porque en el ascii en carácter g tiene asignado el numero 103 asi que al pasarlo por el método que recibe un double se le asigna este numero

2). Short : entra a double: 2. Al ya solo haber un método que reciba algo y en este caso recibe el dato mas alto en la jerarquía, pues todos van a parar a este método y este tipo de dato no es la excepción

3).byte: entra a double:1. Al solo existir un método este se le asigna a ese pero hay una razón por la que esto pasa y no muestra error, porque el dato que recibe el único método que hay es un tipo de dato double y este es el dato mas alto en la jerarquía asi que no hay problemas

4).long : entra a double :9999999. Al solo existir el método que recibe el double, al momento de irnos a la tabla de jerarquía vemos que el mas cercano hacia arriba en la tabla es el double, asi que lo pasamos por ese método pero si en vez del double el método recibiera un tipo de dato de mas baja jerarquía que el long pues esto botaria un error porque uno no se puede devolver en la tabla

5).integer: entra a double :51232. Como no hay un método que reciba un tipo de dato int, nos vamos a buscar que métodos hay y nos encontramos con que hay un método que recibe un double asi que nos vamos por este ya que el tipo de dato que recibe es el inmediatamente superior

6).double: entra a double : 12.4. tiene el método que recibe su tipo de dato asi que no hay problema y se le entrega normalmente al método que recibe su tipo de dato

7).float : entra a double: 5.65. al solamente haber un método , pues debemos mirar si el tipo de dato que recibe es un float y nos damos cuenta de que no, no es un float pero es un tipo de dato mas grande en la jerarquía asi que podemos pasarlo por ahi