• Reconocimiento del formato general correcto de una clase

Los programadores pueden crear sus propias clases, las clases son esencialmente una plantilla o plano para todas las instancias de una clase, el código de la clase también comunica al compilador cómo definir, crear e interactuar con los objetos de la clase, el código de la siguiente diapositiva comienza a crear el vehículo de la clase que representaría el esquema básico para los objetos del vehículo

Ejemplo de la creación de la plantilla de una clase

public class Vehicle {

// the Vehicle class has two fields

private String make;

private int milesPerGallon;

//constructor

public Vehicle(){

}//end constructor

//mutator/setter methods

public void setMake(String m){

make = m;

}//end method setMake

public void setMilesPerGallon(int mpg){

milesPerGallon = mpg;

}//end method setMilesPerGallon

public String getMake(){

return make;

}//end method getMake

public int getMilesPerGallon(){

return milesPerGallon;

}//end method getMilesPerGallon

}//end class Vehicle

• Creación de un objeto de clase

En Java, la creación de una instancia significa la creación de objetos de una clase.

Para crear una instancia de una instancia de un vehículo denominado myCar, escriba:

public class VehicleTester{

public static void main(String[] args){

Vehicle myCar = new Vehicle();

}//end method main

}//end class VehicleTester

• Descripción de referencias de objetos

Una vez que ha creado una clase, puede crear instancias de esta clase (objetos) en una clase de controlador o dentro de otras clases de objetos

Instancias:

−heredan todos los atributos y métodos definidos en la plantilla de una clase

−Interactúan independientemente entre sí

−Son objetos de referencia

−Se crean utilizando el nuevo operador

• Creación de métodos de compilación sin errores

Los constructores son métodos que permiten al usuario crear instancias de una clase (instanciar), en programación se aconseja incluir un constructor predeterminado en las clases, los constructores que contienen parámetros generalmente inicializan las variables privadas de la clase en los valores aprobados por el usuario los constructores no tienen un tipo de devolución (anulados o de otro tipo)

• Retorno de un valor a partir de un método

Métodos de acceso:

Los métodos de acceso acceden y devuelven el valor de una variable privada específica de la clase, el tipo de devolución no nula corresponde al tipo de datos de la variable a la que está accediendo incluye una sentencia de devolución generalmente, no tiene parámetros.

public String getMake(){

return make;

}//end method getMake

public int getMilesPerGallon(){

return milesPerGallon;

}//end method getMilesPerGallon

• Uso de parámetros en un método

Métodos de descriptores de mutación:

Los métodos de descriptores de mutación establecen o modifican el valor de una variable privada específica de la clase. tipo de devolución nula, parámetro con un tipo que corresponde al tipo de variable establecida

public void setMake(String m){\

make = m;

}//end method setMake

public void setMilesPerGallon(int mpg){

milesPerGallon = mpg;

}//end method setMilesPerGallon

• Crear una clase controlador y agregar instancias de clases objetos

public class VehicleTester{

public static void main(String[] args){

Vehicle v;

v=new Vehicle();

v.setMake("Ford");

v.setMilesPerGallon(35);

System.out.print("My "+v.getMake() + " gets " + v.getMilesPerGallon() + " mpg.");

}//end method main

}//end class VehicleTester

• Agregar un constructor a una clase

public Vehicle() {

make = "";

milesPerGallon = 0;

}//end constructor

• Aplicar el nuevo operador

public Vehicle(String m, int mpg){

make=m;

milesPerGallon=mpg;

}//end constructor

• Descripción de la recolección de elementos no utilizados y finalizadores

\*Comprensión de la recolección de elementos no utilizados:

Al considerar el código que figura a continuación, ¿qué ocurrirá en la memoria después de la línea c2 = c;?, al ejecutarla, c2 = c; toma la referencia c2 y hace referencia al mismo objeto como c, de este modo, el objeto c2 original queda efectivamente inutilizable y la recolección de elementos no utilizados se ocupa de este objeto eliminándolo de la memoria.

\*Finalizadores

Un finalizador es un código invocado por el recolector de elementos no utilizados cuando determina que no existen más referencias al objeto, todos los objetos heredan un método finalize() de java.lang.Object, este método no toma parámetros y se escribe para que no realice ninguna acción al ser invocado al invalidar el método finalize() en las clases podrá modificar lo que ocurre antes de la recolección de elementos no utilizados, por ejemplo:

−Notificar al usuario acerca de la recolección de elementos no

utilizados que va a ocurrir

−Limpiar los recursos que no sean de Java, por ejemplo, cerrar

un archivo

• Aplicar esta referencia

Ejemplo del método de finalización

protected void finalize(){

try{

close(); //close all files

}//end try

finally{

System.out.println("Finalization has occured");

}//end finally

}//end method finalize

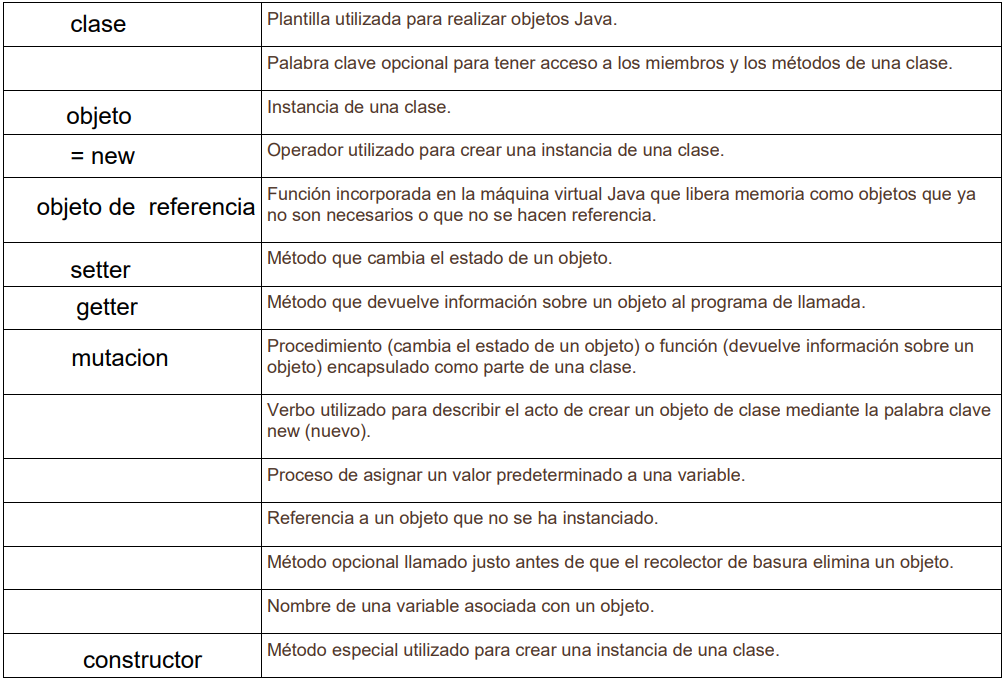
• Agregar un constructor para inicializar un valor

public Vehicle() {

make = "";

milesPerGallon = 0;

}//end constructor



inicialización

instancia

null

finalize

instancia

public

2. Identifique las partes clave de la Clase Java a continuación. Coloque asteriscos junto a todas las variables de instancias.

Coloque un casillero junto a cada constructor. Haga un círculo en la firma de los métodos diferentes al método de constructor.

Coloque triángulos alrededor de los parámetros. Subraye los tipos de retornos de los métodos.

public class Animal {

\*int weight, \*int height;

\*double speed;

\*Animal() {

weight = 50;

height = 4;

speed = 2; // miles per hour

}

\*Animal(int w, \*int h, \*int s) {

weight = w;

\*height = h;

speed = s;

}

public \*double getTime(\*double miles) { // gets the number of hours to go these miles

return miles/speed;

}

public \*int getWeight() {

return weight;

}

public \*int getHeight() {

return height;

}

public \*double getSpeed() {

return speed;

}

}