Acceso a datos privados y datos públicos

La principal diferencia entre las declaraciones de instancia de variables y las declaraciones de variables que se ha visto en el pasado es el modificador de acceso privado (private). Si declara que un miembro es privado, entonces el miembro puede acceder solo desde dentro de la clase del miembro y no desde el "mundo exterior" (es decir, mediante un código que esta fuera de la clase en la que reside el miembro). Las instancias de variables casi siempre se declaran con la modificador de acceso privado porque casi siempre se desea que los datos de un objeto estén ocultos. Hacer que una variable de instancia sea privada le da control sobre cómo se puede cambiar su valor. Por ejemplo, podría preguntar de que un peso nunca sea negativo. Restringir el acceso a los datos es de lo que trata la encapsulación, y es una de las piedras angulares de la programación orientada a objetos.

Además del modificador de acceso privado (private), también hay un modificador de acceso público (public). Dadas las definiciones estándar de las palabras "público" y "privado", probablemente pueda suponer que los miembros públicos son más fácil de acceder que los miembros privados. Si declara que un miembro es público, entonces el miembro puede acceder desde cualquier lugar (desde dentro de la clase del miembro y también desde fuera de la clase del miembro). Tú debes declarar un método como público cuando desees que sea un portal a través del cual el mundo exterior accede a los datos de sus objetos. Regresa y verifica que los tres métodos en la clase Mouse usen el modificador public de acceso. Cuando desees que un método ayude a realizar una tarea local únicamente, debes declararlo como privado, pero pospondremos esa consideración hasta el Capítulo 8.

Mire una vez más las declaraciones de variables de instancia de la clase Raton, tenga en cuenta que inicializamos edad y el peso en 0 y 1.0, respectivamente, pero no inicializamos porcientoCrecer. Eso es porque nos sentimos cómodos con edad = 0 y peso = 1.0 para todos los objetos Raton recién creados, pero no nos sentimos cómodos con un valor inicial predefinido para porcientoCrecer. Es de suponer que querremos utilizar diferentes valores de porcientoCrecer para diferentes objetos Raton (los ratones en un estudio de consumo de donas pueden tener valores de porcientoCrecer más altos que los ratones en un estudio de tabaquismo).

Sin inicialización para la instancia de la variable porcientoCrecer, ¿cómo puede configurar la tasa de crecimiento para un objeto Raton? Puede hacer que el objeto Raton llame al método setPorcientoCrecer con un valor de tasa de crecimiento como argumento. Por ejemplo, así es como un objeto Raton puede establecer su tasa de crecimiento en 10 (por ciento):

Como recordará del Capítulo 5, los valores entre paréntesis de una llamada a un método se denominan argumentos. Por tanto, en este ejemplo, 10 es un argumento. El 10 se pasa a la variable porcientoCrecer en el encabezado de PorcientoCrecer. Las variables entre paréntesis de un encabezado de método se denominan parámetros. Por lo tanto, en el ejemplo que se muestra en la Figura 6.4, porcientoCrecer es un parámetro. Dentro de setPorcientoCrecer cuerpo del método (el código entre las llaves de apertura y cierre del método), el parámetro porcientoCrecer se asigna a la variable de instancia porcientoCrecer. Aquí está la declaración de asignación correspondiente:

{pág. 202}

por la clase son las variables miembro o variables de instancia. El código que opera lo;

Definición de instancias. Cada objeto de una clase contiene la estructura y el comportamiento definidos en la clase, como si fuera grabado por un molde con la forma de la clase. Por esta razón, a los objetos se les llama a veces instancias de una clase. Una clase es una construcción lógica; un objeto tiene realidad física.

Cuando se crea una clase, hay que especificar el código y los datos que constituyen a la clase. En su conjunto, estos elementos son miembros de la clase. Los datos definidos por la clase son las variables miembros o variable de instancia. El código que opera sobre esos datos se conoce como método miembro o simplemente método.

### **Variables de instancia**

Las variables de instancia almacenan los datos para las instancias individuales de una clase. Por ejemplo, en la clase "Helper", la siguiente variable de instancia podría ser declarada:

private String myName;

Esto normalmente aparece antes del método constructor, que puede crear una instancia de la siguiente manera:

public Helper(String helperName) { myName = helperName; }

Es una práctica común asegurar que todas las variables de instancia tengan asignado un valor en el momento en el que método constructor haya terminado de ejecutarse. Las variables de instancia se declaran con visibilidad privada, lo que significa que sus valores sólo pueden cambiarse dentro de la clase.

### **Acceso**

El código externo para la clase a veces puede acceder a las variables de instancia a través de los métodos de clase, como en el siguiente código de ejemplo:

public String getName(){ return myName; }

El código externo puede acceder a la variable llamando a este método en una instancia de la clase, de la siguiente manera:

String theName = myHelp.getName();

Se trata de un método "accesor". Los métodos que proporcionan código externo con la capacidad de modificar los valores de las variables de instancia, los métodos "mutadores", pueden llevar a cabo revisiones para evitar que valores no válidos se utilicen.

{[https://techlandia.com](https://techlandia.com/), *Sue Smith}*

Ocultar una variable de instancias

Es cuando la clase y el método tienen la variable con el mismo nombre.

Se tiene una superclase y una subclase:

class Parent {

int var = 1;

}

class Child extends Parent {

int var = 2;

}

Child child = new Child();

Parent parent = (Parent)child; Trata a este objeto como si fuera una instancia de Parent.

System.out.println(parent.var); *// prints out 1, instead of 2*

Si Parent.var no debe ser visible para Child, hazlo private.

*Si Parente.var y Child.var son variables lógicamente distintas, dales nombres diferentes para evitar confusiones*

*Si Parent.var y Child.var son variables lógicamente iguale, luego usa un miembro de datos para ellos.*

class Parent {

private int var = 1;

public int getVar() {

return var;

}

public void setVar(int var) {

this.var = var;

}

}

class Child extends Parent {

private int var = 2;

public int getVar() {

return var;

}

public void setVar(int var) {

this.var = var;

}

}

{https://living-sun.com/es/java/487449-hiding-instance-variables-of-a-class-java-oop.html}