Encapsulamiento y Clases predefinidas

Dr. José Lázaro Martínez Rodríguez

Encapsulamiento

- El encapsulamiento sirve para proteger los datos de los objetos y se logra declarando los atributos de una clase como private y codificando métodos especiales para controlar el acceso.
- La manera de acceder a los atributos desde afuera de la clase es por medio de los métodos *getter* y la manera de modificar los atributos desde afuera de la clase es usando los métodos *setter*.

Encapsulamiento

- El encapsulamiento oculta lo que hace un objeto de lo que hacen otros objetos y del mundo exterior por lo que se denomina también ocultación de datos.
- Un objeto tiene que presentar “una cara” al mundo exterior de modo que se puedan iniciar sus operaciones.
- esto sugiere que solamente la información sobre lo que puede hacer una clase debe estar visible desde el exterior, pero no cómo lo hace.
- Esto tiene una gran ventaja:
  - si ninguna otra clase conoce cómo está almacenada la información entonces se puede cambiar fácilmente la forma de almacenarla sin afectar otras clases.

Acceso a miembros

- Podemos reforzar la separación del qué hacer del cómo hacerlo, declarando los campos como **privados** y usando un método de acceso para acceder a ellos.
- Los métodos de acceso son el medio de acceder a los atributos privados del objeto. Son métodos públicos del objeto y pueden ser:
  - **Métodos modificadores**: Dan lugar a un cambio en el valor de uno o varios de los atributos del objeto.
  - **Métodos consultores u observadores:** Devuelven información sobre el contenido de los atributos del objeto sin modificar los valores de estos atributos.

Métodos consultores u observadores

- Cuando se crea una clase es frecuente que lo primero que se haga sea establecer métodos para consultar sus atributos
- Estos métodos suelen ir precedidos del prefijo **get** (getNombre, getValor, etc.)
- Se les conoce coloquialmente como “**getters**”

Métodos modificadores

- Métodos que permitan establecer los valores de los atributos
- Suelen ir precedidos del prefijo **set** (setNombre, setValor, etc.)
- por lo que muchas veces se alude coloquialmente a ellos como “métodos set” o “setters”

Ejemplo

```
public class Circulo {

public static double PI=3.14;
private double radio;

}
```

- Ahora cómo definir y obtener radio

Ejemplo

```
public class Circulo {

public static double PI=3.14;
private double radio;

public void setRadio(double radio) {
  this.radio = radio;
}

public double getRadio() {
  return this.radio;
}

}
```

- *this* hace referencia a las variables globales

Ejemplo

- Y colocamos los otros métodos

```
public double perimetro() {
  return 2 * Circulo.PI * radio;
}

public double area() {
  double tmp = Circulo.PI * radio * radio;

  return tmp;
}
```

Ejemplo

```
public class PruebaCirculo {
  public static void main(String[] args) {
  //creamos instancias
  Circulo c1 = new Circulo();
  Circulo c2 = new Circulo();

  //asignamos valores con el set
  c1.setRadio(10);
  c2.setRadio(5);

  //invocamos métodos e imprimimos resultados devueltos
  System.out.println("El área de c1 es: " + c1.area());
  System.out.println("El perímetro de c2 es: " + c2.perimetro());
}
}
```

Encapsulamiento

- Como el encapsulamiento es una práctica común, muchos ambientes de desarrollo orientado a objetos codifican automáticamente los métodos getters y setters con las siguientes reglas:
  1. Tanto getters como setters son públicos.
  2. Los getters no reciben parámetros y el tipo de dato que regresan es el mismo que el del atributo correspondiente. Su nombre comienza con get seguido del nombre del atributo pero iniciando con mayúscula y regresan el valor del atributo.
  3. Los setters reciben como parámetro el mismo tipo de dato que el del atributo. El nombre de los métodos setters se construye de forma análoga a la de los getters, pero iniciando con la palabra set, y asignan al atributo el valor del parámetro recibido.

Ejemplo

- Desarrollando clases con setters y getters
- Defina el tipo de dato para cada atributo