**HERENCIA JERÁRQUICA**

* Las clases se vuelven más específicas y concretas con cada nueva subclase (hacia abajo).
* Si se mueve de una subclase hacia una superclase (hacia arriba), las clases se vuelven más generales y menos específicas.
* El diseño de las clases debe asegurar que una superclase contiene características comunes de/para sus subclases.
* En ocasiones, una superclase es tan abstracta que no puede ser usada para crear alguna instancia (algún objeto) en específico. Dicha clase es conocida como clase abstracta.

**CLASES ABSTRACTAS**

* Una clase abstracta no puede ser utilizada para crear objetos (instancias).
* Una clase abstracta puede tener métodos abstractos, los cuales se implementan en una subclase concreta.
* Considerar el siguiente código como ejemplo:

|  |
| --- |
|  |

La clase **ObjetoGeometrico** es una superclase que representa/modela características comunes de objetos geométricos.

* El constructor de la clase **ObjetoGeometrico** es definido como protected, porque solo es usado por subclases.

**Métodos Abstractos**

* Es un hecho que se puede calcular el área y el perímetro de todos los objetos geométricos. Entonces, se crean los métodos **getArea()** y **getPerimetro()** en la clase **ObjetoGeometrico**.
* Sin embargo, estos métodos no pueden ser implementados en la clase **ObjetoGeometrico,** porque su implementación depende de un objeto geométrico en específico (de una subclase).
* Es decir, el área o perímetro de un objeto no siempre se calcula igual que la de otro objeto geométrico. Por ejemplo: el área de un círculo no se calcula de la misma manera en que se calcula el área de un rectángulo. Lo mismo ocurre con el cálculo del perímetro.
* Debido a esto, los métodos **getArea()** y **getPerimetro()** deben ser abstractos. Para que un método sea abstracto se debe anotar el modificador **abstract** en el encabezado del método. Además, se deben definir **sin implementación**, es decir, no deben tener un cuerpo, a continuación un ejemplo:

|  |
| --- |
|  |

* Una clase que tiene métodos abstractos debe definirse como **clase abstracta**, se debe anotar el modificador **abstract** en el encabezado de la clase. La clase **ObjetoGeometrico** contiene métodos abstractos, por lo tanto debe definirse como clase abstracta:

|  |
| --- |
|  |

**Entonces:**

* Un método abstracto se define sin implementación, su implementación es proporcionada por las subclases de la superclase.
* Los métodos abstractos de una superclase deben ser sobre escritos e implementados en sus subclases.

**Subclases de la Superclase ObjetoGeometrico**

|  |  |
| --- | --- |
|  |  |

* **ObjetoGeometrico** representa/modela características comunes de objetos geométricos.
* **ObjetoGeometrico** es la superclase de las subclases **Circulo** y **Rectangulo**.
* Un método abstracto no puede ser implementado (no se coloca cuerpo) en la superclase, el método abstracto debe ser implementado en las subclases.
* Los métodos getArea() y getPerimetro() de la superclase ObjetoGeometrico se implementan (se coloca cuerpo) en las subclases Circulo y Rectangulo, como se muestra en las imágenes de arriba.
* Los métodos abstractos de la superclase se sobre escriben (@Override) en las subclases.

**¿Por qué métodos abstractos?**

¿Cuál es la ventaja de definir los método **getArea()** y **getPerimetro()** como abstractos en la clase **ObjetoGeometrico**? El código siguiente muestra los beneficios:

|  |
| --- |
|  |

* El programa crea dos objetos geométricos, **miCirculo** y **miRectangulo**.
* Se invoca al método **comparacionAreas()** para verificar si los dos objetos geométricos tienen áreas iguales.
* Se invoca al método **verInformación()** para mostrar la información de dichos objetos.
* Cuando se invoca al método **compararAreas()**, el método **getArea()** definido en la clase **Circulo** es usado para **primero.getArea()** ya que el *argumento* **primero** es una instancia de tipo **ObjetoGeometrico** y el parametro **miCirculo** también lo es**.**
* Lo mismo sucede con el **Rectangulo**, el método **getArea()** definido en la clase **Rectangulo** es usado para **segundo.getArea()** ya que el *argumento* **segundo** es una instancia de **ObjetoGeometrico** y el *parametro* **miRectangulo** también lo es.
* Algo parecido ocurre cuando se invoca al método **verInformacion(miCirculo)**, los métodos **getArea()** y **getPerimetro()** definidos en **Circulo** son usados para **objeto.getArea()** y **objeto.getPerimetro().**
* Lo mismo pasa cuando se invoca de nuevo al método **verInformacion(miRectangulo)**. Los métodos **getArea()** y **getPerimetro()** definidos en **Rectangulo** son usados para **objeto.getArea()** y **objeto.getPerimetro().**
* No podríamos definir el método **compararAreas()** si el método **getArea()** no fuera definido en la clase **ObjetoGeometrico**.

**Puntos Interesantes Sobre las Clases Abstractas**

* Una clase abstracta es como una clase normal, pero no se pueden crear instancias de una clase abstracta usando el operador new.
* Un método abstracto es definido sin implementación, su implementación es proporcionada por las subclases.
* Una clase que contiene métodos abstractos debe definirse como abstracta.
* Un método abstracto no puede ser contenido por una clase no abstracta.
* Si una subclase de una superclase abstracta no implementa todos los métodos, la subclase debe definirse como abstracta.
* Los métodos abstractos son no estáticos.
* Una clase abstracta no puede ser instancia utilizando el operador new, pero puedes definir su constructor o constructores. Dicho constructor será invocado en el constructor de la subclase.
* Una clase que contiene métodos abstractos debe ser definida como abstracta. Sin embargo, es posible definir una clase abstracta sin métodos abstractos. En este caso, tampoco se puede instanciar utilizando el operador new, esta clase es utilizada como una clase base para definir subclases.
* Una subclase puede sobre escribir un método de su superclase para definirla como abstracta. Esto es muy inusual, pero es muy útil cuando la implementación del método en la superclase se vuelve inválido en la subclase. En este caso, la subclase debe definirse como abstracta.
* Una subclase puede ser abstracta incluso si su superclase no lo es (clase concreta, clase normal). Por ejemplo: la clase **Object** es concreta pero sus subclases, como **ObjetoGeometrico**, pueden ser abstractas.
* No se pueden crear instancias de una clase abstracta, pero si se puede utilizar como tipo de dato.

|  |
| --- |
|  |