Java para Cibernética y Computación

Colegio de Ciencias y Humanidades – Plantel Oriente

Instructores:

¡¡¡TODOS LOS PROFESORES INSCRITOS EN EL CURSO!!!

Día 3

ArrayList

ArrayList

- La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el numero de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño como pasa con los Arrays.
- Los ArrayList nos permiten añadir, eliminar y modificar elementos de forma trasparente para el programador.

Declaración de un ArrayList

- `ArrayList<tipo> nombreArray = new ArrayList <tipo>();`

- Ejemplos:
- `ArrayList<String> nombreArrayList = new ArrayList<String>();`
- `ArrayList<Integer> nombreArrayList = new ArrayList<Integer>();`
- `ArrayList<Float> nombreArrayList = new ArrayList<Float>();`
- `ArrayList<Double> nombreArrayList = new ArrayList<Double>();`

Algunos de métodos de los arrayList

- **add()**//Añade un elemento
- **size()**//Devuelve el número de elementos del arreglo
- **get()**//Extrae el elemento del arreglo que se le pasa como parámetro
- **remove()**//Borra elementos del arreglo
- **clear()**//Borra todos los elementos del arreglo
- **isEmpty()**//Devuelve True si el ArrayList esta vacío. Si no devuelve False.
- **clone()**//copia en contenido de un arreglo a otro

Programación Orientada a Objetos

Objeto

- Un objeto es una entidad de software que contiene tanto información (atributos) como procedimientos (métodos).

- **Atributos:** características del objeto.

- **Métodos:** Son funciones que ejecutan operaciones sobre los atributos del objeto.

Instanciar

› La palabra instanciar significa crear objetos de una clase. La creación de objetos nos permitirá acceder a los atributos y métodos de una clase, es decir, si no se crea un objeto es imposible acceder a las características y a las acciones de una clase. Para crear un objeto se utiliza la palabra reservada ***new***.

Ejemplo:

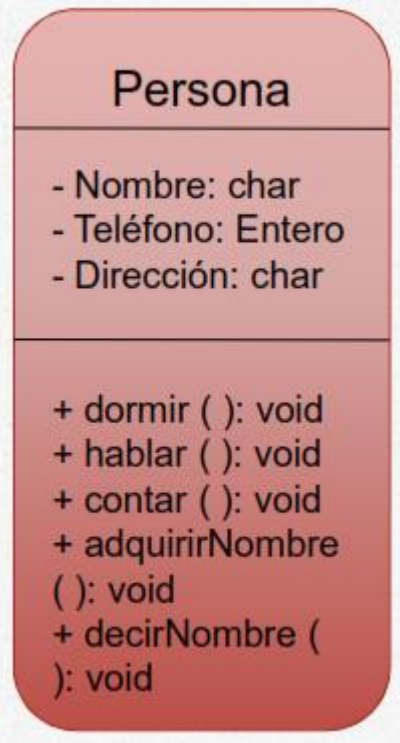
```
Persona persona1 = new Persona();
```

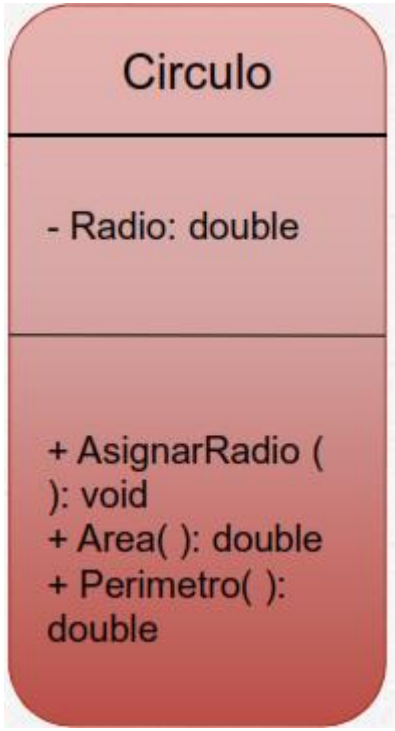
Clase

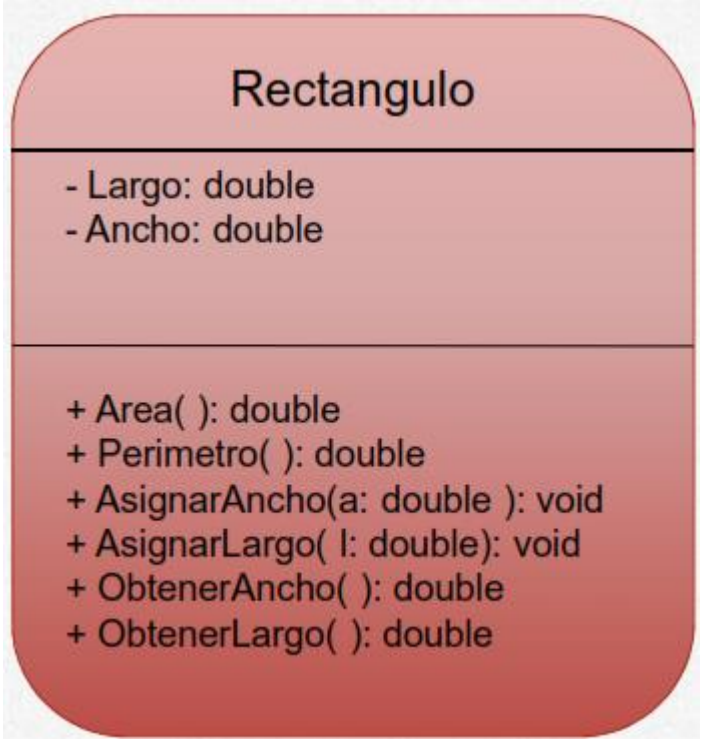
- Una clase es código que especifica tanto los atributos, como los métodos de un objeto en particular.
- Una analogía serían los moldes de pastel.

Representación de clases en UML

- El diagrama de clase es una representación gráfica de la clase que ayuda al programador a visualizar cuales son los atributos y métodos que contendrá.







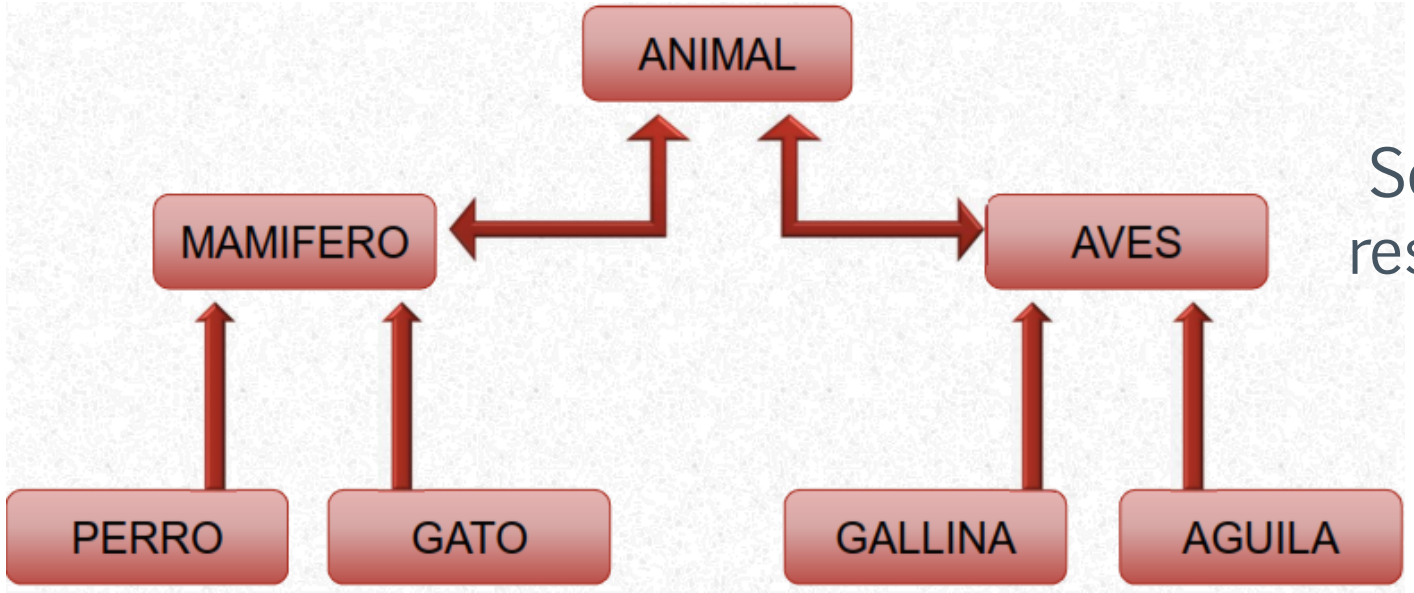
Abstracción

- La abstracción es como se pueden representar los objetos en modo de código.

- Es un método mediante el cual representamos una determinada situación de la vida real; sus características y funciones que desempeñan.

Herencia

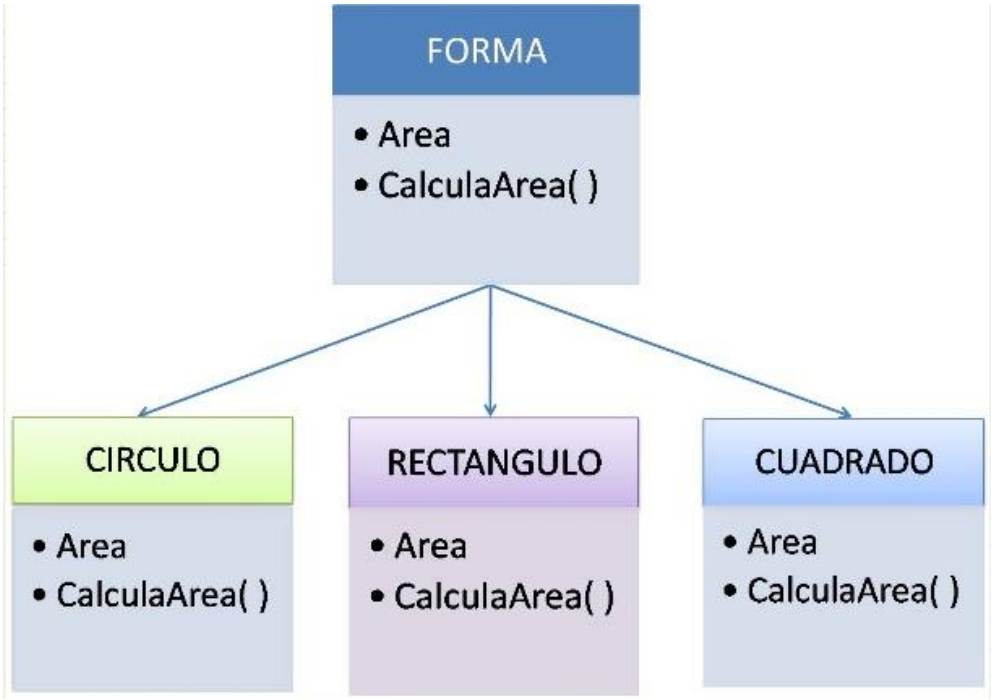
- El mecanismo de herencia permite definir nuevas clases partiendo de otras ya existentes. Las clases que derivan de otras heredan automáticamente todo su comportamiento, pero además pueden introducir características particulares propias que las diferencian.
- La principal ventaja de la herencia es evitar escribir el mismo código varias veces.



Se usa la palabra reservada **extends**

Polimorfismo

- Permite usar un nombre para varios propósitos relacionados, pero ligeramente diferentes.
- Los comportamientos pueden ser identificados bajo el mismo nombre pero procesan información de manera diferente de acuerdo al objeto que lo contenga.



Encapsulamiento

› Esta propiedad permite el ocultamiento de la información, es decir, permite asegurar que el contenido de un objeto se pueda ocultar del mundo exterior dejándose ver lo que cada objeto necesite hacer publico.

Modificadores de acceso

Modificadores de acceso

- Permiten controlar la forma de acceder a los atributos y métodos **encapsulados** dentro de una clase.

- **PÚBLICO:** Cualquier atributo o método Publico puede se accedido desde fuera de la clase.

- **PRIVADO:** Cualquier atributo o método Privado NO puede se accedido desde fuera de la clase. Solo puede ser utilizado internamente en la clase.

- **PROTEGIDO:** Cualquier atributo o método Protegido puede ser heredado por otra clase pero en esta ultima se convierten en elementos.

Métodos constructores

- Los constructores son métodos especiales que sirven para inicializar un objeto de una determinada clase al mismo tiempo que se declara.
- Tienen el mismo nombre que la clase a la que pertenecen.
- No tienen tipo de retorno, por lo tanto no retornan ningún valor, ni siquiera void.
- Deben ser públicos (no tendría ningún sentido declarar un constructor como privado, ya que siempre se usan desde el exterior de la clase).

Métodos de clase

- Son aquellos en los que no hace falta instanciar un objeto de la clase para utilizarlos. Son métodos estáticos, por lo tanto se usa la palabra reservada ***static*** para definirlos.

```
nombreClase.variable;
nombreClase.metodo();
```

Ejemplo:

```
public class Persona{
        public static int obtenerAltura() {
                //Aquí va la definición del método.
       }
}
```

Métodos de instancia

› Son aquellos en los que se necesita crear un objeto de la clase para poder utilizarlos. Son métodos que **no** usan la palabra reservada ***static*** para definirlos.

```
objeto.metodo();
```

Ejemplo:

```
public class Persona{
	public void comer (int cantidadDeAlimento){
		//Aquí va la definición del método.
	}
}
```

void

- Los métodos pueden devolver algo, por ejemplo, un método que suma dos números, devuelve el resultado de la suma; pero hay métodos que no devuelven nada y que sólo ejecutan acciones. Dichos métodos se declaran con la palabra reservada “**void**” (vacío) como tipo de retorno.

Métodos **get()** y **set()**

- Los métodos **get** y **set**, son simples métodos que usamos en las clases para mostrar (get) o modificar (set) el valor de un atributo.

- El nombre del método siempre será get o set y a continuación el nombre del atributo, su modificador siempre es **public** ya que queremos mostrar o modificar desde fuera la clase. Por ejemplo, **getNombre** o **setNombre**.

Sintaxis de **get()** y **set()**

```
public tipo_dato_atributo getAtributo(){
     return atributo;
}

public void setAtributo(tipo_dato_atributo variable){
    this.atributo = variable;
}
```

Método **toString()**

El método **toString()** devuelve un String que representa al objeto. El contenido de este String depende del objeto sobre el que se esté aplicando.

El método **toString()** se invoca de forma automática cuando se muestra un objeto mediante la instrucción **System.out.println** o **System.out.print**