CLASES ABSTRACTAS JAVA

- HABLAR DE CLASES ABSTRACTAS ES GENERALIZAR LAS CLASES, FORZANDOLAS A SEGUIR UN PATRON DE COMPORTAMIENTO(QUE HACER, NO COMO HACER).
- ESTE TIPO DE CLASES NOS PERMITEN CREAR METODOS GENERALES QUE RECREAN UN COMPORTAMIENTO COMUN PERO SIN ESPECIFICAR COMO LO HACEN.
- PARA DECLARAR UNA CLASE O METODO COMO ABSTRACTOS, SE UTILIZA LA PALABRA RESERVADA abstract.

```
abstract class Figura {
      protected int ancho,alto;
      void setDatos(int x,int y)   //método no abstracto
      {
            ancho=x;
            alto=y;
      }
      void mostrarDatos()    //método no abstracto
      {
            System.out.println(ancho);
            System.out.println(alto);
      }
      abstract float getArea();   //método abstracto
}
```

- UNA CLASE ABSTRACTA NO SE PUEDE INSTANCIAR, PERO SI SE PUEDEN HEREDAR Y LAS CLASES HIJAS AGREGARAN LA FUNCIONALIDAD A LOS METODOS ABSTRACTOS

CLASES ABSTRACTAS JAVA

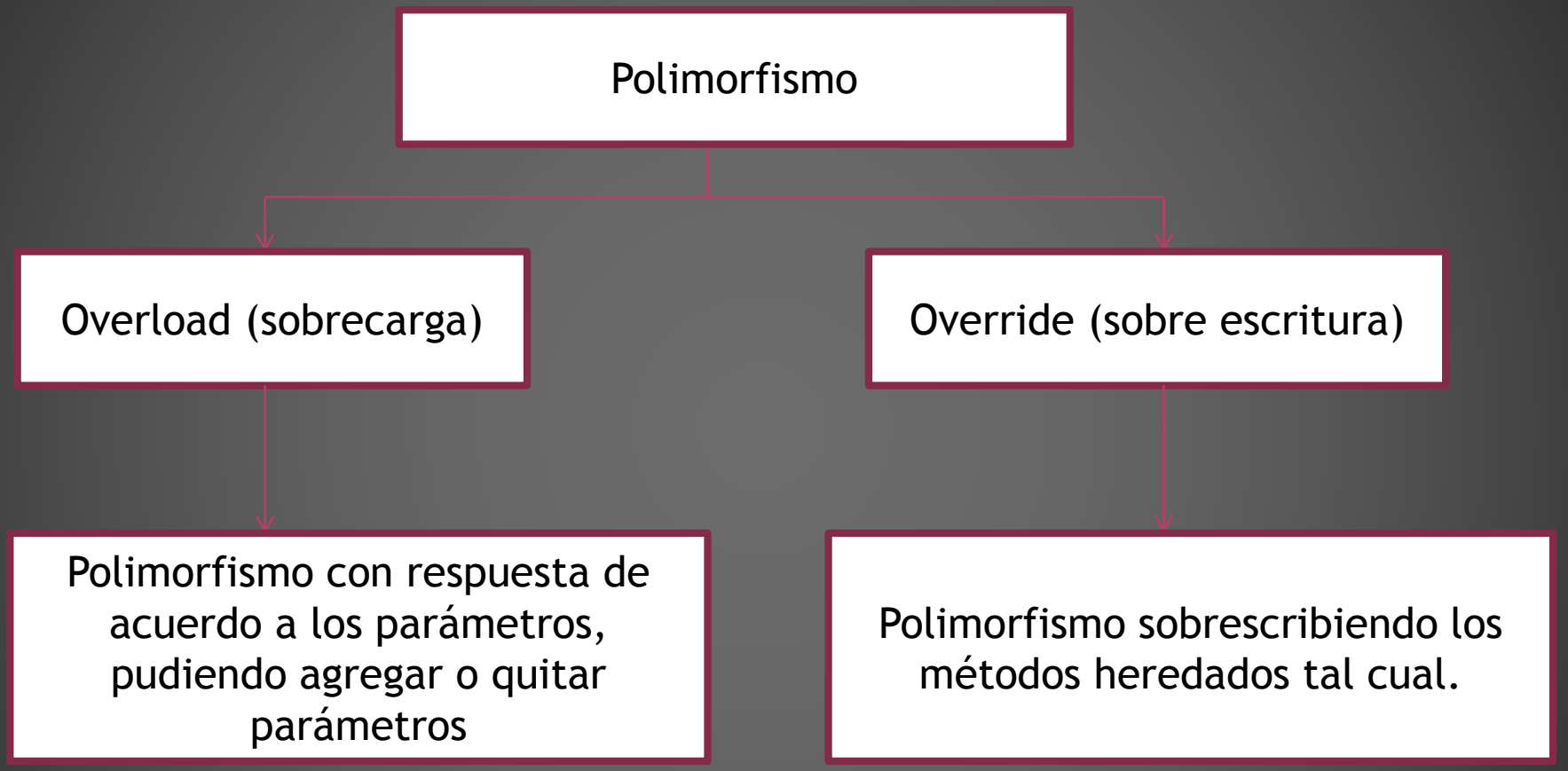
- SI UNA CLASE CONTIENE UN METODO ABSTRACTO, LA CLASE DEBE DECLARARSE TAMBIEN COMOABSTRACTA.
- UNA CLASE ABSTRACTA PUEDE CONTENER METODOS ABSTRACTOS Y TAMBIEN NO ABSTRACTOS.
- UN METODO ABSTRACTO NO TIENE CUERPO, SOLO TERMINA CON UN ;.

```
class Rectangulo extends Figura{

      float getArea()  //implementación (desarrollo del método)
      {
            return (ancho*alto);
      }
}
```

```
class Triangulo extends Figura{

      float getArea()   //implementación (desarrollo del método)
      {
            return ((ancho*alto)/2);
      }
}
```

```
public class prueba {
      public static void main(String[] args) {
            Rectangulo r=new Rectangulo();
            r.setDatos(4,3);
            System.out.println(r.getArea());
            Triangulo t=new Triangulo();
            t.setDatos(4,3);
            System.out.print(t.getArea());
      }
}
```

EL POLIMORFISMO EN JAVA



EL POLIMORFISMO EN JAVA

- Se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos.
- El polimorfismo nos permite flexibilidad.
- Polimorfismo: muchas formas, múltiples formas de llamar algo.
- Un objeto se puede comportar de diferente forma dependiendo del contexto. Las variables objeto son polimórficas.

Para que exista polimorfismo:

- Debe existir herencia
- Los métodos de las clases padre e hijos deben llamarse igual y tener los mismos parámetros y mismo tipo de retorno.
- El método de la clase hijo al que se quiera tener acceso con un objeto de la clase padre debe de tener la anotación @override.

INTERFACES EN JAVA

- Una **interfaz** en Java es una colección de métodos abstractos y propiedades constantes.

- En las interfaces se especifica qué se debe hacer pero no su implementación.

- Las clases que implementen estas interfaces serán las que describan la lógica del comportamiento de los métodos.

Ventajas

- Organizar la programación.

- Sustituye de alguna manera la necesidad de uso de herencia múltiple

- Permiten declarar constantes que van a estar disponibles para todas las clases que queramos (implementando esa interfaz).

- Obligar a que ciertas clases utilicen los mismos métodos (nombres y parámetros).
- Establecer relaciones entre clases que no estén relacionadas.

INTERFACES EN JAVA

Para declarar una interfaz:

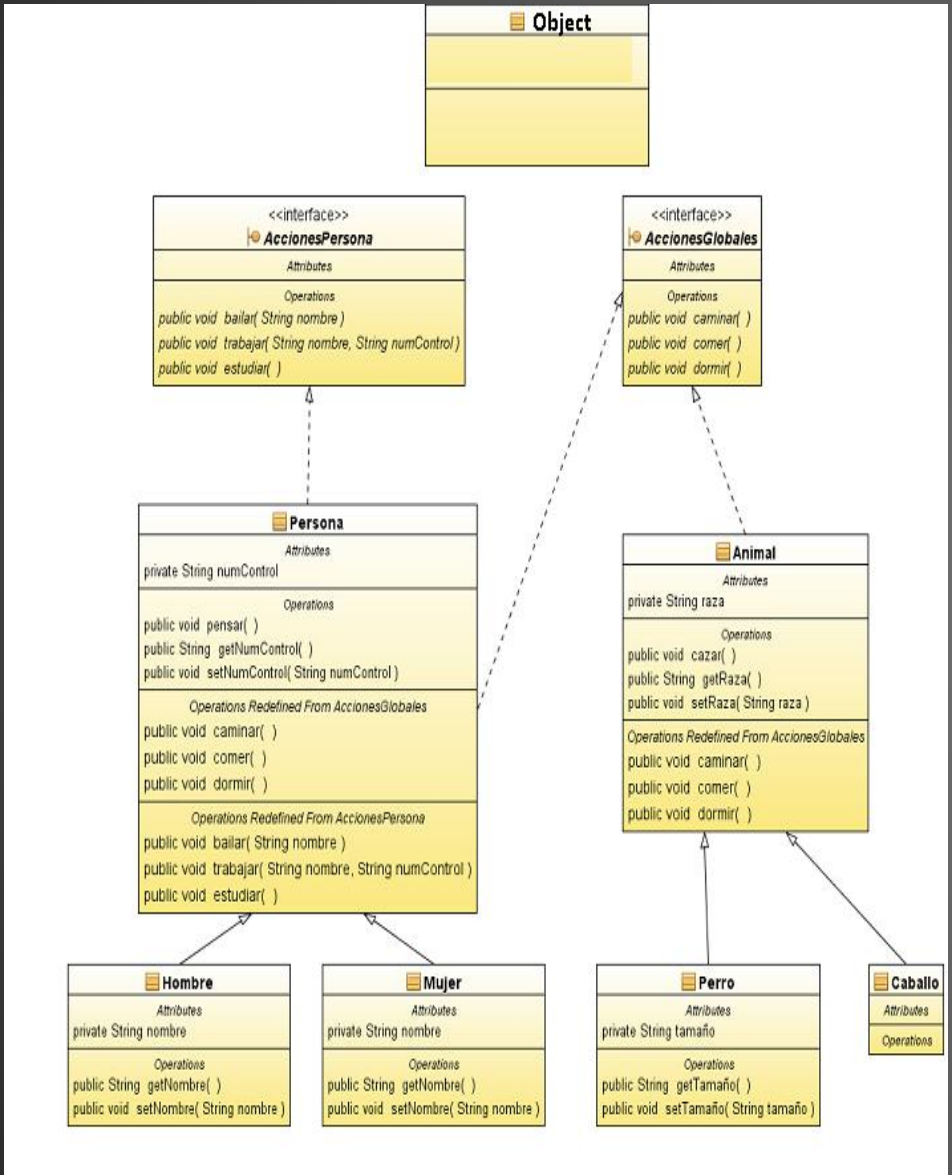
```
interface Nave {
    public abstract moverPosicion (int x, int y);
    public abstract disparar();
    .....
}
```

- Uso de la interfaz definida:

```
public class NaveJugador implements Nave {
    public void moverPosicion (int x, int y) {
        //Implementación del método
        posActualx = posActualx - x;
        posActualy = posActualy - y;
    }

    public void disparar() {
        //Implementación del método
    }

    ...
}
```

- Los métodos no contienen ninguna implementación.
- Los atributos declarados son por default públicos, estáticos y finales.
- Los métodos declarados en una interface son públicos y abstractos, terminan con punto y coma.
- Una interfaz puede heredar de otra interfaz pero no de una clase concreta.
- Una interfaz no puede ser instanciada

OPERADOR "instanceof"



- Sirve para conocer si un objeto es de un tipo determinado (clase o interface) en tiempo de ejecución, regresa true en caso de corresponder de lo contrario es false.
- Sólo puede usarse con variables que contengan la referencia a un objeto.
- Todos los objetos derivan de la clase *Object*.

```
static void queTipoDeObjetoEs(Object objeto){
    if(objeto instanceof Caballo ){

    } if(objeto instanceof Persona){

    } if(objeto instanceof AccionesPersona){

    } if(objeto instanceof Object){

    }
}
```

CREACION DE ARCHIVOS JAR

- El formato de archivo JAR (java archive file) es un formato de compresión usado principalmente para distribuir aplicaciones y bibliotecas Java.

- Esta basado en el formato ZIP y funciona de manera similar.

- Se puede crear usando el Java Development Kit (JDK) y el símbolo del sistema de tu computadora.

PASOS A SEGUIR:

1. Prepara tus archivos.
2. Abre el símbolo del sistema.
3. Navega a la carpeta donde tienes almacenados todos tus archivos.
4. Crear un archivo de manifiesto (manifest.mf), contendra el nombre de la clase principal.
5. Crea el archivo JAR. "jar cmf 'archivo-jar'.jar archivo(s)" cmf(crear,manifiesto,file).
6. Ejecutar nuestra aplicación con java -jar /path/MyProgram.jar

MODIFICADORES DE ACCESO EN JAVA

- Los *modificadores* de acceso nos introducen al concepto de *encapsulamiento*.
- Permiten dar un nivel de seguridad mayor a nuestras aplicaciones restringiendo el acceso
- Asegura que el usuario deba seguir una "ruta" especificada por nosotros para acceder a la información.
- La palabra default no es una palabra reservada.

Modificador	La misma clase	Mismo pa quete	Subclase	Otro paque te
private	Sí	No	No	No
default	Sí	Sí	No	No
protected	Sí	Sí	Sí/No	No
public	Sí	Sí	Sí	Sí

Visibilidad	Significado	Java	UML
Pública	Se puede acceder al miembro de la clase desde cualquier lugar.	`public`	+
Protegida	Sólo se puede acceder al miembro de la clase desde la propia clase o desde una clase que herede de ella.	`protected`	#
Por defecto	**Se puede acceder a los miembros de una clase desde cualquier clase en el mismo paquete**		~
Privada	Sólo se puede acceder al miembro de la clase desde la propia clase.	`private`	-

MODIFICADORES DE ACCESO EN JAVA