Objetos

Contenido

Objetos

- ¿Qué es un objeto?
- Creación de un objeto.
- Características de un objeto.
- Destrucción de un objeto.
- Ciclo de vida de un objeto.
- Métodos de acceso.

Paquetes

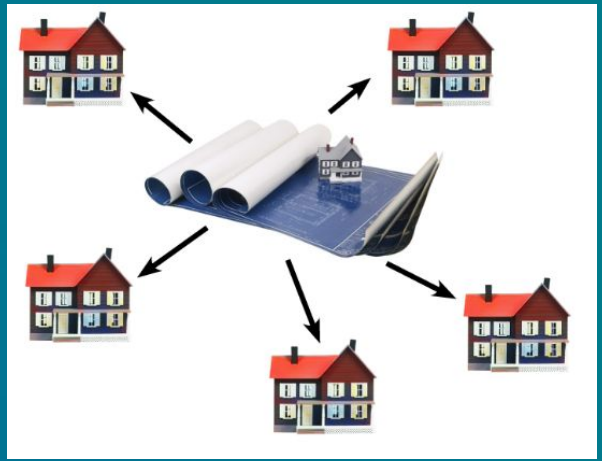
- ¿Qué es un *package* en Java?
- ¿Cómo crear paquetes propios?
- ¿Cómo importar un paquete?
- Definir clases en paquetes.
- ¿Qué es un archivo .jar?

¿Qué es un objeto?

Un objeto es una **instancia** de una clase.

Una instancia es una **manifestación concreta** de algo.

Una clase es el modelo o plano a partir del cual se crean los objetos.



Creación de un objeto

Los objetos se crean a partir de su **constructor**.

En Java, un constructor es un método especial cuya función es darle un valor inicial a los atributos de un objeto, para asegurar el correcto funcionamiento del mismo.

Si no se define un constructor, se invoca al constructor por defecto. El cual inicializa los atributos con valores predeterminados (***null*** o ***cero***).

Se invocan automáticamente cuando se crea una instancia de la clase utilizando la palabra clave ***new***.

Poseen el mismo identificador de la clase que lo contiene. No retorna nada, ni siquiera ***void***.

Creación de un objeto - Sintaxis

```java
public class Persona {

    public String apellido;
    public String nombre;
    public int edad;


    public Persona(){

        this.apellido = "Doe";
        this.nombre = "John";
        this.edad = 20;
    }

}
```

```java
// DECLARAR AL OBJETO
Persona persona;

// INSTANCIAR AL OBJETO
persona = new Persona();
```

```java
// DECLARAR E INSTANCIAR AL OBJETO
Persona otraPersona = new Persona();
```

Creación de un objeto - Constructor

Tipos de constructores en Java:

Constructor de instancia:

- Se utilizan para crear instancias de una clase.
- Aunque no se escriba ningún constructor, existe uno por **defecto**.
- Pueden recibir parámetros.
- Inicializan los atributos de instancia (**no estáticos**) de la clase.

Si bien en Java **no** hay constructores estáticos (cómo en C#), existen los bloques estáticos:

- Se ejecutan cuando la clase se carga en memoria por primera vez.
- Son conocidos cómo **bloques estáticos de inicialización**.
- No llevan identificador alguno, solo la palabra reservada ***static***.
- No tiene modificador de acceso.
- Inicializan los atributos **estáticos**.

Creación de un objeto - Constructor

```java
public class Persona {

    public String apellido;
    public String nombre;
    public int edad;
    public static String datoEstatico;

    static{

        Persona.datoEstatico = "valor inicial";
    }

    public Persona(){

        this.apellido = "Doe";
        this.nombre = "John";
        this.edad = 20;
    }

}
```

Creación de un objeto - Singleton

El patrón de diseño **Singleton** (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

```
// NO SE PUEDE HACER ESTO, PORQUE EL CONSTRUCTOR
// ES PRIVADO: Singleton s = new Singleton();

// OBTENER LA ÚNICA INSTANCIA, UTILIZANDO EL
// MÉTODO ESTÁTICO
Singleton s = Singleton.getInstancia();
```

```
public class Singleton {

    private static Singleton instancia;

    // CONSTRUCTOR PRIVADO
    private Singleton(){

        // INICIALIZACIÓN DE LA INSTANCIA
    }

    // MÉTODO ESTÁTICO PARA OBTENER
    // LA ÚNICA INSTANCIA DE LA CLASE
    public static Singleton getInstancia(){

        if( Singleton.instancia == null ){

            Singleton.instancia = new Singleton();
        }

        return Singleton.instancia;
    }
}
```

Características de un objeto

Memoria Heap

- Los objetos en Java se almacenan en la memoria del *heap*, que es una porción de la memoria reservada para la alocación dinámica.
- Cuando se crea un objeto usando el operador ***new***, se asigna memoria en el heap para el objeto.

Memoria Stack

- Las referencias de los objetos se almacenan en la memoria del *stack*. Las variables locales y las referencias se almacenan aquí.

Tipos y referencias

- En Java, las variables de tipo objeto son **referencias** que apuntan a los datos reales en el *heap*.
- Las referencias pueden ser ***null***, lo que indica que no apuntan a ningún objeto.

Destrucción de un objeto

El tiempo de vida de una variable local está vinculado al ámbito en el que está declarada.

- Tiempo de vida corto (en general).
- Creación y destrucción ***deterministas***.

El tiempo de vida de un objeto **no está vinculado a su ámbito**.

- Tiempo de vida más largo.
- Destrucción ***no determinista***.

Los objetos se destruyen por un proceso conocido como **recolección de basura.**

En este proceso, un programa (**Garbage collector**) busca objetos inalcanzables y los destruye. Los convierte de nuevo en memoria binaria no utilizada.

Destrucción de un objeto

Garbage collection (Recolección de basura)

- Java usa un recolector de basura automático para liberar la memoria ocupada por los objetos que ya no son accesibles.
- No es posible liberar memoria manualmente como en lenguajes como C++.
- La recolección de basura es administrada por la **JVM** y se ejecuta en segundo plano. No hay garantía de cuándo exactamente se liberará dicha memoria.

Aunque no se recomienda, Java proporciona el método ***finalize()*** que puede ser sobrescrito para realizar la limpieza antes de que un objeto sea recolectado por el ***GC***.

Ciclo de vida de un objeto



CREACIÓN (Instanciar)	UTILIZACIÓN	DESTRUCCIÓN
El **operador new** lo <u>único</u> que hace es reservar memoria binaria sin inicializar. El **constructor** inicializa el estado del objeto en valores seguros (y sólo eso).	Una vez instanciado el objeto se pueden invocar sus métodos y atributos a partir de la referencia.	El **Garbage Collector** es el **encargado de liberar memoria** sin intervención de los programadores. *En general, liberará memoria de objetos sin referencia.* Es administrado por la **JVM**.

Ejercitación

Paquetes

Paquetes en Java

Un paquete es una agrupación de clases, interfaces y subpaquetes que se organizan de una manera jerárquica y lógica.

Los paquetes ayudan a evitar conflictos de nombres y a organizar el código de manera estructurada.

En Java, los paquetes (*packages*) son el equivalente a los namespaces de C#.

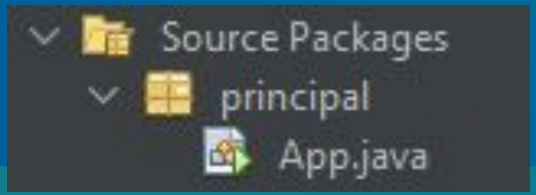
Paquetes en Java

- **Organización del código**: Los paquetes ayudan a organizar clases e interfaces en grupos relacionados.
- **Evitar conflictos de nombres**: Al usar paquetes, se puede tener clases con el mismo nombre en diferentes paquetes.
- **Control de acceso**: Los paquetes pueden definir niveles de acceso diferentes para las clases y sus miembros.
- **Facilitar el mantenimiento y la búsqueda**: Tener un esquema de organización claro facilita encontrar y mantener el código.

Paquetes en Java - Creación

Para crear un paquete, simplemente se coloca una declaración ***package*** y su identificador al inicio del archivo de código.

Recordar que el archivo *.java* debe estar en el directorio **Source Packages/principal/**




```java
package principal;

public class App {

    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

Paquetes en Java - Importación

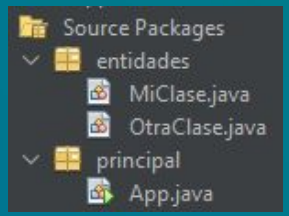
Para usar clases de otros paquetes hay que utilizar la declaración ***import***.

Hay dos formas principales de importación:

- Importar una clase específica.
- Importar todas las clases del paquete.

```java
package entidades;

public class MiClase {

}
```

```java
package entidades;

public class OtraClase {

}
```




```java
package principal;

import entidades.MiClase;

public class App {

    public static void main(String[] args) {

        MiClase mc = new MiClase();

        entidades.OtraClase oc = new entidades.OtraClase();
```

```java
package principal;

import entidades.*;

public class App {

    public static void main(String[] args) {

        MiClase mc = new MiClase();

        OtraClase oc = new OtraClase();
```

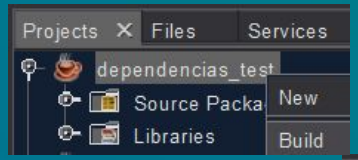
¿Qué es un archivo .jar?

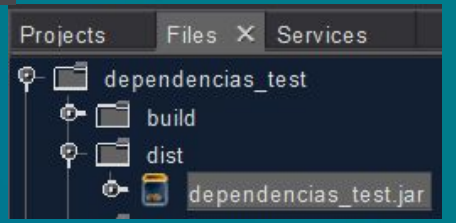
Los archivos ***.jar*** (Java ARchive) son archivos comprimidos que se utilizan para agrupar múltiples archivos de clase de Java, bibliotecas, recursos asociados (como imágenes y archivos de configuración), y metadatos en un solo archivo para facilitar la distribución y la ejecución.

- **Formato de archivo comprimido**: Un archivo ***.jar*** es un archivo ZIP que contiene archivos de clase Java, archivos de metadatos y otros recursos.
- **Facilita la distribución**: Permite empaquetar todas las clases y recursos necesarios en un único archivo.
- **Ejecución**: Puede ser ejecutable si contiene un archivo MANIFEST.MF con la entrada Main-Class.
- **Reutilización de bibliotecas**: Facilita la reutilización y distribución de bibliotecas de terceros.

¿Cómo generar un archivo .jar?

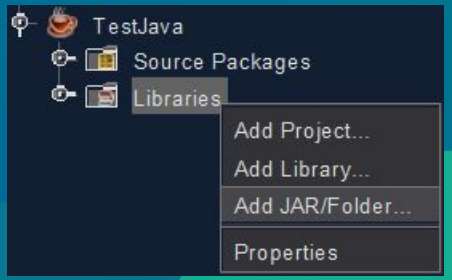
Para crear un archivo ***.jar*** hay que compilar el proyecto que contiene las entidades.
El .jar se ubicará en la carpeta **dist/**

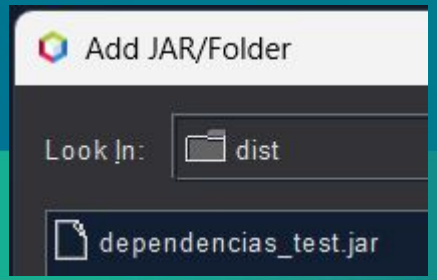




¿Cómo agregar un archivo .jar?

En el proyecto destino, click derecho sobre la carpeta *Libraries* y seleccionar *Add JAR/Folder*
Seleccionar el .jar y ¡listo!





Ejercitación