Resumen de guía 1

Java

Conceptos básicos

1. Características de la plataforma java
2. Proyecto java
3. Organizar clases en paquetes
4. El método main
5. Comentarios
6. Clases y Objetos - atributos y métodos
7. Tipos de datos primitivos

Características de la plataforma Java

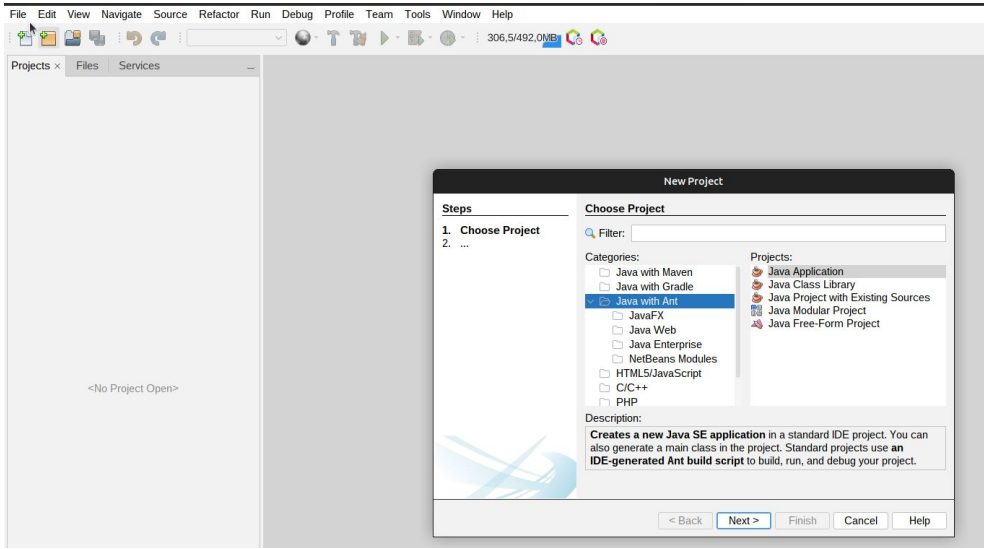
Java soporta el paradigma orientado a objetos, un enfoque de programación que se basa en la creación de objetos que interactúan entre sí para resolver problemas a partir de la definición de clases. Se enfoca en encapsular datos y comportamientos en objetos para promover la modularidad y la reutilización de código.

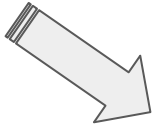
Java es un lenguaje híbrido (nuestros programas son en parte compilados y en parte interpretados) por ello los programas resultan más eficientes, flexibles y rápidos.

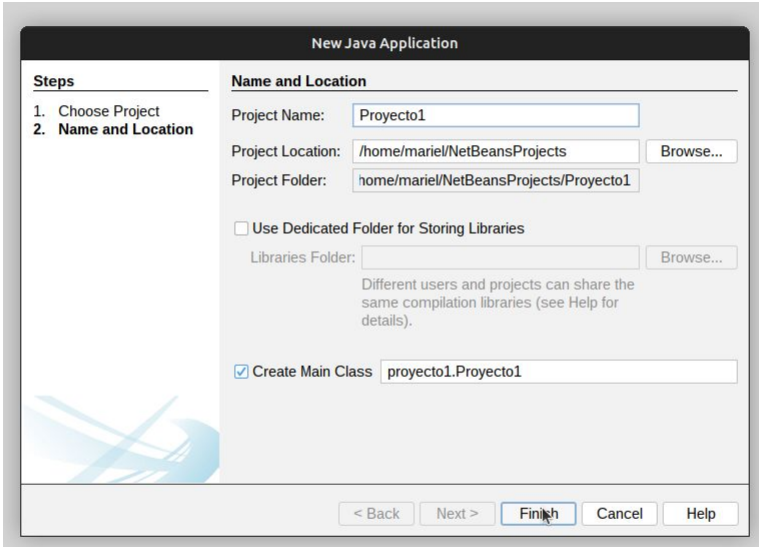
Otra característica es su portabilidad (ya que nuestros programas se pueden ejecutar en cualquier sistema operativo)

Se considera seguro (Se ejecutan en una máquina virtual, sandbox)

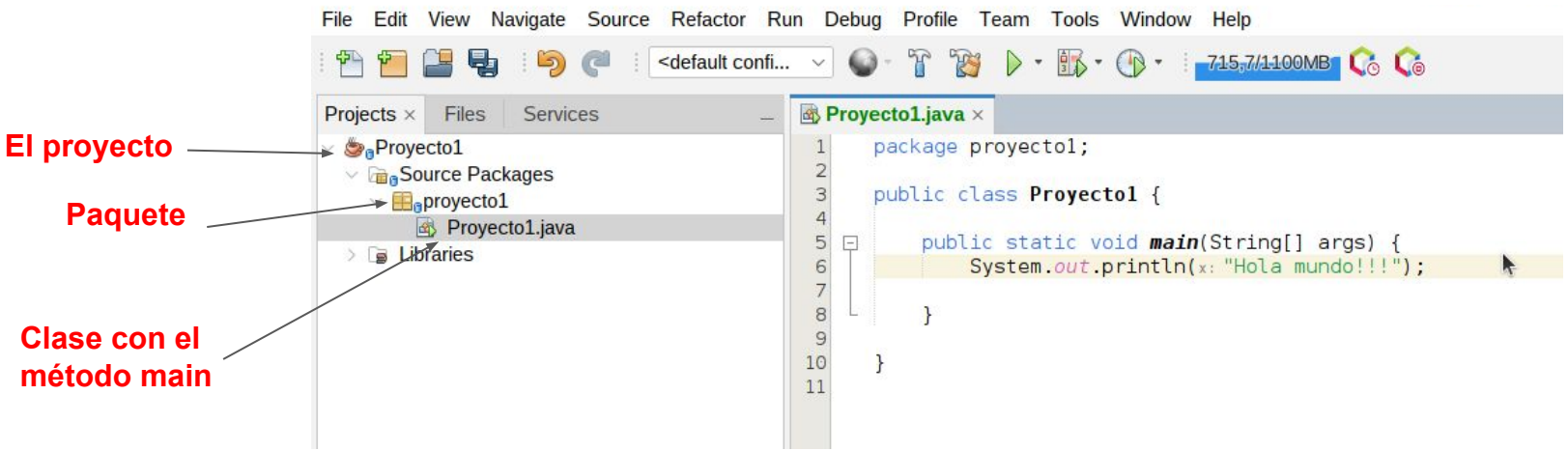
Creando un proyecto







El proyecto

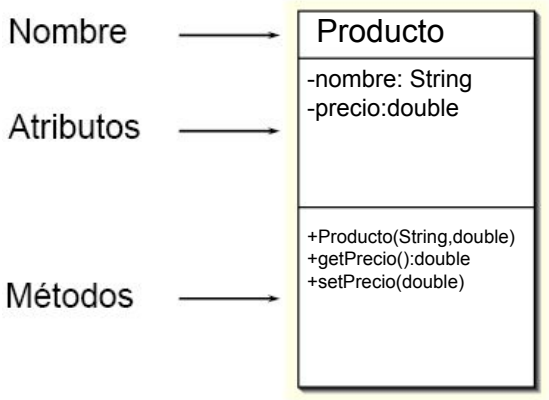


La ejecución del proyecto comienza por la clase que tiene el método main.

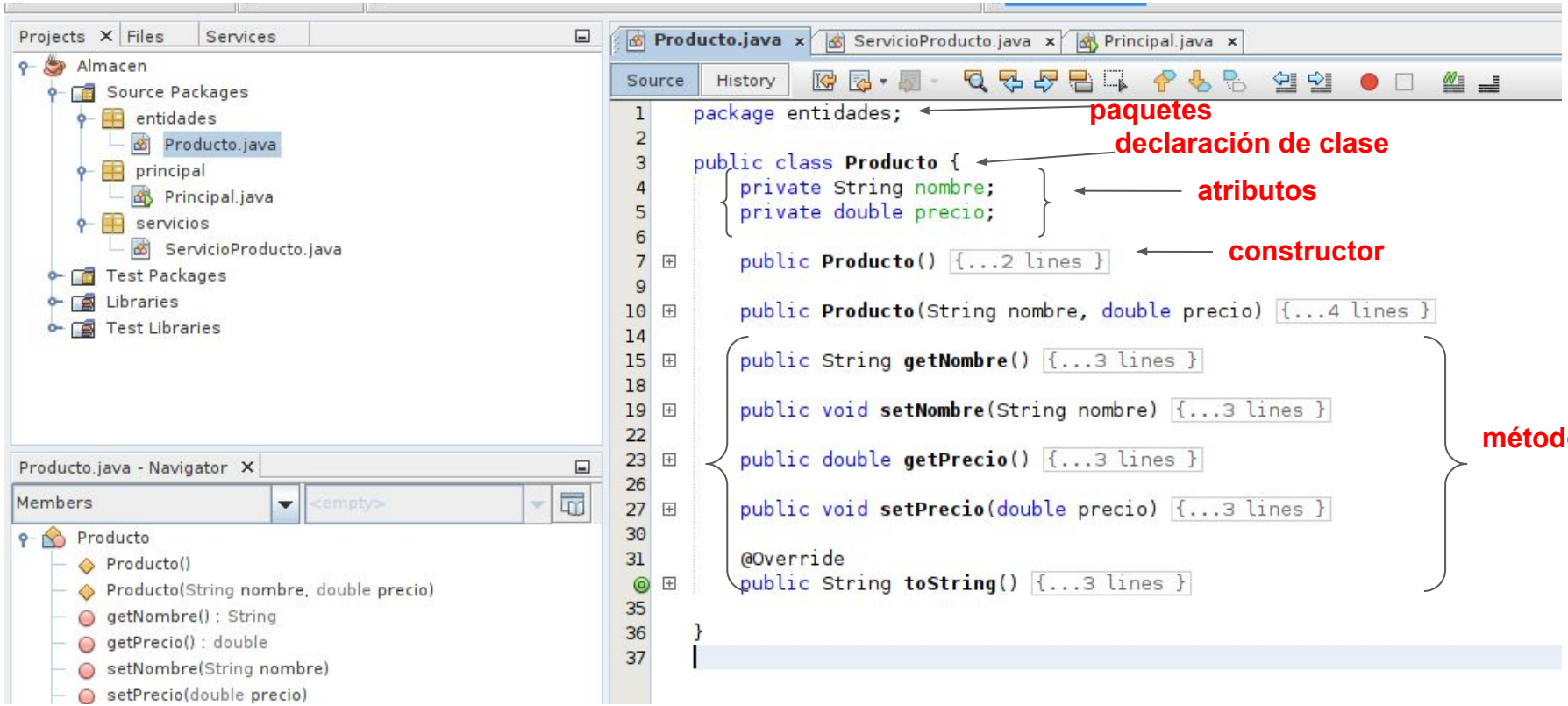
Clase y objeto java

Una clase es una plantilla que define **atributos** (estado de un objeto) y **métodos** (comportamiento de un objeto). Podemos pensar en una clase como en la definición de un nuevo tipo de datos.

Luego de generar la clase es posible declarar **Objetos (instancias)** de ese nuevo tipo de datos.



ESTRUCTURA DE UNA CLASE



Tipos de datos primitivos JAVA

Tipo	Representación / Valor	Tamaño (en bits)	Valor mínimo	Valor máximo	Valor por defecto
`boolean`	`true` o `false`	1	N.A.	N.A.	`false`
`char`	Carácter Unicode	16	`\u0000`	`\uFFFF`	`\u0000`
`byte`	Entero con signo	8	-128	128	0
`short`	Entero con signo	16	-32768	32767	0
`int`	Entero con signo	32	-2147483648	2147483647	0
`long`	Entero con signo	64	-9223372036854775808	9223372036854775807	0
`float`	Coma flotante de precisión simple Norma IEEE 754	32	±3.40282347E+38	±1.40239846E-45	`0.0`
`double`	Coma flotante de precisión doble Norma IEEE 754	64	±1.79769313486231570E+308	±4.94065645841246544E-324	`0.0`

variable vs objeto & tipo de datos vs. clase - analogías …

instancia == objeto ⟶ son sinónimos

variable ~ objeto ⟶ Tanto una variable como un objeto de POO son contenedores que pueden almacenar valores. Sin embargo, un objeto de POO es un tipo especial de contenedor que puede almacenar múltiples valores y funciones relacionados entre sí.

tipo de datos ~ clase ⟶ Un tipo de datos define la estructura y el comportamiento de un solo tipo de valor, mientras que una clase define la estructura y el comportamiento de un concepto más complejo que puede contener múltiples propiedades y métodos

El **constructor** de una clase

Es un **método especial** dentro de una clase, que se llama automáticamente cada vez que se crea un objeto de esa clase. Debe tener el mismo nombre de la clase a la cual pertenece y no puede devolver ningún valor.

Puede haber más de un constructor, mientras los tipos o cantidad de parámetros difieran.(sobrecarga)

El constructor inicializa los atributos y asigna memoria de manera dinámica, en el heap, a cada instancia que se crea.

Métodos getters y setters

Los **getters** (de la palabra inglés get - obtener) indica que podemos tomar algún valor de un atributo y los **setters** (de la palabra inglés set-poner/fijar) podemos guardar algún valor sobre un atributo.

```
public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}
```

La clase Principal

La ejecución del proyecto comienza por la clase que tiene el método main.

```
package principal;

import servicios.ServicioProducto;

public class Principal {


    public static void main (String arg[]){
        ServicioProducto sp=new ServicioProducto();
        sp.crearLista();
        sp.mostrarLista();
        sp.incrementarPrecios(10);
        sp.mostrarLista();

    }
}
```

métodos static

El método se puede ejecutar sin crear una instancia de la clase que lo contiene…

```
/**...4 lines */
public class Ejemplo {

    private int atributo;

    public Ejemplo(int atributo) {...3 lines }

    public int getAtributo() {...3 lines }

    public void setAtributo(int atributo) {...3 lines }

    public static void cartel(){
        System.out.println(x: "Mostrando un cartelito ;)");
    }
}
```

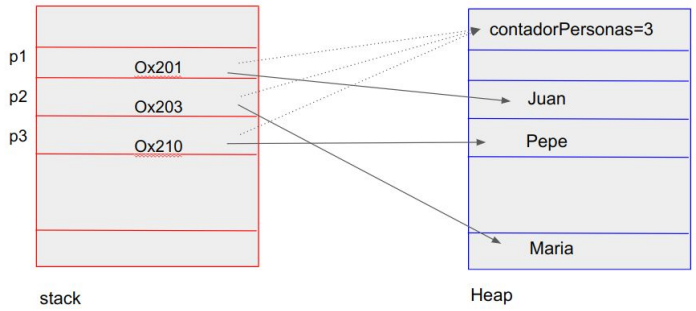
```
package Ejemplo;
public class Main {
    public static void main(String[] args) {

        Ejemplo.cartel();


    }
}
```

método estático

clase que tiene el método estático

atributo static

Son variables de la clase. Todas las instancias comparten el mismo espacio de memoria asignado al atributo estático.




```java
public class Persona {
    private static int contadorPersonas = 0;
    private String nombre;

    public Persona(String nombre) {
        this.nombre = nombre;
        contadorPersonas++;
    }

    public static int getContadorPersonas() {
        return contadorPersonas;
    }
}
```

```java
package Ejemplo;
public class Main {
    public static void main(String[] args) {

        Persona p1=new Persona(nombre: "Juan");
        Persona p2=new Persona(nombre: "Maria");
        Persona p3=new Persona(nombre: "Pepe");
        System.out.println("La cantidad de personas creadas es: "+ Persona.getContadorPersonas());

    }
}
```

jemplo.Main > main > p3 >

ut ×

mariel - /home/mariel × Guia1 (run) ×

```
run:
La cantidad de personas creadas es: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```