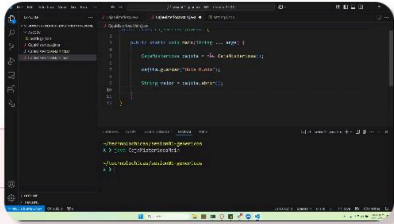
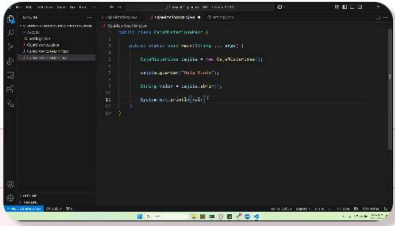


```
1 import java.util.*;
2
3 // Clase abstracta
4 abstract class OrdenProduccion {
5     protected String codigo;
6     protected int cantidad;
7
8     public OrdenProduccion(String codigo, int cantidad) {
9         this.codigo = codigo;
10        this.cantidad = cantidad;
11    }
12
13    public abstract void mostrarResumen();
14 }
15
16 // Subclase: Producción en masa
17 class OrdenMasa extends OrdenProduccion {
18     public OrdenMasa(String codigo, int cantidad) {
19         super(codigo, cantidad);
20     }
21
22     @Override
23     public void mostrarResumen() {
24         System.out.println("OrdenMasa - Código: " + codigo + " - Cantidad: " + cantidad);
25     }
26 }
```

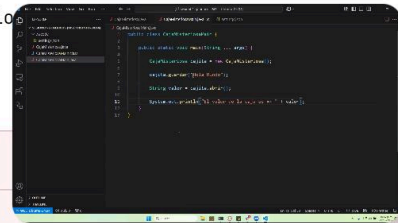
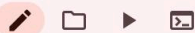


```
25 }
26 }
27
28 // Subclase: Orden personalizada
29 class OrdenPersonalizada extends OrdenProduccion {
30     private String cliente;
31
32     public OrdenPersonalizada(String codigo, int cantidad, String cliente) {
33         super(codigo, cantidad);
34         this.cliente = cliente;
35     }
36
37     @Override
38     public void mostrarResumen() {
39         System.out.println("🔧 OrdenPersonalizada - Código: " + codigo + " - Cantidad: " + cantidad + " - Cliente: " + cliente);
40     }
41
42     public void aplicarCostoAdicional(int costo) {
43         System.out.println("✅ Orden " + codigo + " ajustada con costo adicional de $" + costo);
44     }
45 }
46
47 // Subclase: Prototipo
48 class OrdenPrototipo extends OrdenProduccion {
49     private String faseDesarrollo;
```



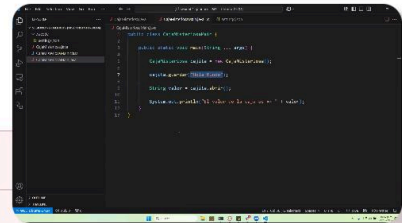
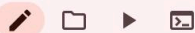
```
51 public OrdenPrototipo(String codigo, int cantidad, String faseDesarrollo) {
52     super(codigo, cantidad);
53     this.faseDesarrollo = faseDesarrollo;
54 }
55
56 @Override
57 public void mostrarResumen() {
58     System.out.println("🔧 OrdenPrototipo - Código: " + codigo + " - Cantidad: " + cantidad + " - Fase: " + faseDesarrollo);
59 }
60 }
61
62 // Clase principal
63 public class PlantaIndustrial {
64     // Método genérico para mostrar cualquier tipo de orden
65     public static void mostrarOrdenes(List<? extends OrdenProduccion> lista) {
66         for (OrdenProduccion orden : lista) {
67             orden.mostrarResumen();
68         }
69     }
70
71     // Método para procesar solo órdenes personalizadas
72     public static void procesarPersonalizadas(List<? super OrdenPersonalizada> lista, int costoAdicional) {
73         System.out.println("\n🔧 Procesando órdenes personalizadas...");
74         for (Object obj : lista) {
75             if (obj instanceof OrdenPersonalizada) {
76                 ((OrdenPersonalizada) obj).aplicarCostoAdicional(costoAdicional);
77             }
78         }
79     }
80 }
```

Home End → () { } < > ' " ;

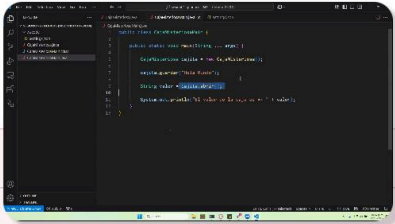


```
76         ((OrdenPersonalizada) obj).aplicarCostoAdicional(costoAdicional);
77     }
78 }
79 }
80
81 // Método adicional: contar tipos de órdenes
82 public static void contarOrdenes(List<OrdenProduccion> todas) {
83     int masa = 0, personalizada = 0, prototipo = 0;
84     for (OrdenProduccion orden : todas) {
85         if (orden instanceof OrdenMasa) masa++;
86         else if (orden instanceof OrdenPersonalizada) personalizada++;
87         else if (orden instanceof OrdenPrototipo) prototipo++;
88     }
89
90     System.out.println("\n📊 Resumen total de órdenes:");
91     System.out.println("🏭 Producción en masa: " + masa);
92     System.out.println("🔧 Personalizadas: " + personalizada);
93     System.out.println("🛠️ Prototipos: " + prototipo);
94 }
95
96 public static void main(String[] args) {
97     // Crear órdenes
98     OrdenMasa o1 = new OrdenMasa("A123", 500);
99     OrdenMasa o2 = new OrdenMasa("A124", 750);
100
101     OrdenPersonalizada o3 = new OrdenPersonalizada("P456", 100, "ClienteX");
```


Home End → () { } < > ' " ;



```
101 OrdenPersonalizada p1 = new OrdenPersonalizada("P456", 100, "ClienteX");
102 OrdenPersonalizada p2 = new OrdenPersonalizada("P789", 150, "ClienteY");
103
104 OrdenPrototipo t1 = new OrdenPrototipo("T789", 10, "Diseño");
105 OrdenPrototipo t2 = new OrdenPrototipo("T790", 5, "Pruebas");
106
107 // Listas por tipo
108 List<OrdenMasa> listaMasa = Arrays.asList(o1, o2);
109 List<OrdenPersonalizada> listaPersonalizadas = Arrays.asList(p1, p2);
110 List<OrdenPrototipo> listaPrototipos = Arrays.asList(t1, t2);
111
112 // Mostrar órdenes
113 System.out.println("📋 Órdenes registradas:");
114 mostrarOrdenes(listaMasa);
115
116 System.out.println("\n📋 Órdenes registradas:");
117 mostrarOrdenes(listaPersonalizadas);
118
119 System.out.println("\n📋 Órdenes registradas:");
120 mostrarOrdenes(listaPrototipos);
121
122 // Procesar personalizadas
123 List<OrdenProduccion> todas = new ArrayList<>();
124 todas.addAll(listaMasa);
125 todas.addAll(listaPersonalizadas);
126 todas.addAll(listaPrototipos);
```



23



```
python3 -m pip install pygments
Collecting pygments
  Downloading pygments-2.11.2-py3-none-any.whl (1.1 MB)
    100% |#####| 1.1 MB 1.1 MB/s
Installing collected packages: pygments
Successfully installed pygments-2.11.2
pygments 2.11.2 is already installed.
```