①
```
public interface I1{
    void A();
    void B();
    void C();
}

class C1 implements I1{
    public void A(){
        sout...
    }
    public void B(){
        sout...
    }
    public void C(){
        sout...
    }
}

class principal{
    public static void Main (String [] args){
        I1 i1 = new I1();
        i1.A();
        i1.B();
        i1.C();
    }
}
```

- Cannot create an interface
  - ↳ Interfaces cannot be directly instantiated
    - ↳ Necesita un mediador, C1

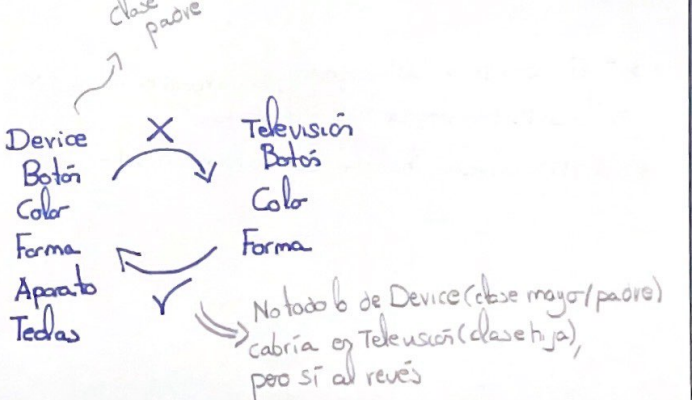
```
C1 c1 = New C1();
c1.A();
...
```

②
```
Class Device{
    public void print(){
        sout(Device Class);{
    }
class ElectronicDevice extends Device{
    public void print(){
        sout(Elec. Device class);{
                                 {
class Television extends ElectronicDevice{
    public void print(){
        sout(Television class);{
                                 (

public class Main {
    public static void main(String[] args){
        Device d = new Device();
        ElectronicDevice e = new ElectronicDevice();
        Television t = new Television();
        t = d;
        t.print();{
            (
```

↳ Error, imaginemos que ⇒

La solución "correcta" sería d = t;



③
```
Class Device{
    public void print(){
      sout(Derice class);{
            {
Class ElectronicDevice extends Device{
    public void print(){
      sout(Elec. Class);{
          {
class Television extends ElectronicDevice{
    public void print(){
      Sout(1. Television class);{
    public void print2(){
      sout(2. Television class);{
    public class Main{
       public static void main(String[] args){
       Device d[] = new Device[3]
       d[0] = new Device();
       d[1] = new ElectronicDevice();
       d[2] = new Television();
       d[2].print2();{
       {
{
```

→ Así accedemos a la clase padre. No existe el print2();
Casting ⇒ ((Television) d[2]).print2();
↳ Le decimos a qué clase acceder

④
```
public interface I1{
    void A();
    void B();
    void C();
{
class C1 implements I1{
    public void A(){
      sout(A);{
    public void B(){
      sout(B);{
{
class C2 extends C1{
    public void C(){
      sout(C);{
{
```

Esta clase debería de ser abstracta, ya que no usa los tres métodos de su clase superior (I1).
Si fuera abstracta solo haría falta que las clases hijas de C1 añadiesen el método de void C();

Como es el caso

→ "It is possible to have methods with the same name in Java, I overload them" ✓

↳ El override sirve para anular un método de otra clase y tome este mismo las riendas

→ "To execute a Java program, we should have a code written in Java, convert it a bytecode and pass it to the JVM for interpretation and execution."

→ "A JVM emulates the architecture of the computer in order to interpret a bytecode"

⑤

```
import javax.swing.*;
import java.awt.*;
public class Ventana{
    JButton boton = new JButton(click);
    JFrame ventana = new JFrame(Ventana);

  public Ventana(){
     ventana.setSize(300,300);
     ventana.setDefaultCloseOpeation(JFrame...)
     ventana.setVisible(true);
     ventana.add(boton);{
  private class Oyente implements ActionListener{
   @Override
    public void actionPerformed(ActionEvent e){
      JOptionPane.showMessageDialog(null, "hola");{
    {
  {
```

Falta → `boton.addActionListener(new Oyente());` para aceptar la acción del botón y acceder al método

⑫

```
class Device{
    public void print(){
    Sout(Device class);{
    {
class Electronic device extends Device{
    public void print(){
     Sout(Electronic class);{
    {
class Television extends Electronic Device{
    public void print(){
    Sout(Television);{
  {
public class Main{
  public static void main(String[] args){
  Device d[] = new Device[3];
  d[0] = new Device();
  d[1] = new ElectronicDevice();
  d[2] = new Television();
  for(int i=0;i<3;i++){
      d[i].print();{
    {
{
```

Bien hecho, imprime

D
E
V
I
C
E

Device → d[0] = Device class
Elec. Device → d[1]: ElectronicDevice class
Television → d[2]: Television class

Todos heredan

```
import java.util.ArrayList;
class Device {
  int numID;
  public void print() {
  Sout(Device class);(
  @Override
  public boolean equals (Object obj) {
    if (obj instanceof Device) {
      Device d = (Device) obj;
      if (this.numID == d.numID) {
        return true;(
      (
      return false
    (
(
class ElectronicDevice extends Device {
    public void print() {
    Sout(Elect. Device class);(
(
class Television extends ElectronicDevice {
    public void print() {
    Sout(1 Telev. class);(
    public void print2() {
    Sout(2 Telev. class);(
    public void print(int i) {
      for (int j = 0; j < i; j++) {
        Sout(TV class);
  { (
```

⑦

```
public class Main {
  public static void main(String[] args) {
ArrayList<Elec.Dev.> A = new ArrayList<Elec.Dev.>();
 A.add(new Device());
A.add(new Elec.Device());
A.add(new Television());
A.get(1).print();
```

↳ Según la IA no hay error, pero Sotodolara supongo que dirá que no se puede meter una clase padre (Device) dentro de una hija (Elec. Device)

⑧

```
public class Main {
 public static void main(String[] args) {
  Device d[] = new Device[3];
  d[0] = new Device();
  d[1] = Electronic Device();
  d[2] = new Television();
  Device d2[] = new Device[3]
  d2[0] = new Device();
  d2[1] = new ElectronicDevice();
  d2[2] = new Television();
  if (d[0] == d2[0]) {
    Sout(d[0] == d2[0]);(
  (
{
```

↳ Nothing appears on the screen.

Prueba con

(d[2].equals(d2[2]))

~ Con d == d2 tampoco sirve ~

⑩

```
public class Main {
  public static void main(String [] args) {
    ArrayList A = new ArrayList();
    A.add(new Device());
    A.add(new ElectronicDevice());
    A.add(new Television());
    A.get(1).print();(
(
```

No se puede usar, ya que accede al print de Device pero debería ir a Elect. Device

solución → ((ElectronicDevice)A.get(1)).print();

Casting!!!