INTERFACE, DOWNCASTING and FINAL CLASS

1. Create two interfaces, Printer and Scanner. Both interfaces have a **default method** called connect():

- Printer’s connect() prints "Connecting to printer".
- Scanner’s connect() prints "Connecting to scanner".

Create a class AllInOneDevice that implements both interfaces.
Override the connect() method to resolve the conflict by printing: "Connecting to all-in-one device".
Also, inside the overridden method, call both interfaces' default connect() methods.

2. You are designing a secure banking system. One of the core components is an immutable class called BankAccount which represents a customer's bank account. This class must be **final** to prevent subclassing and alteration of critical account behavior.

Requirements:

Create a final class BankAccount with the following properties:

- Private final fields: accountNumber (String), accountHolderName (String), and balance (double).
- A constructor to initialize these fields.
- Only getter methods, no setters, to ensure immutability.

Implement methods:

- deposit(double amount) returns a **new** BankAccount instance with updated balance.
- withdraw(double amount) returns a **new** BankAccount instance with updated balance if sufficient funds exist.
- Override toString() to print account details.

Demonstrate in your main program:

- Attempting to subclass BankAccount and explain the error.
- Create an account instance.
- Perform a deposit and withdrawal showing immutability by returning new instances rather than modifying the existing object.
- Show that the original instance remains unchanged after operations.

3. You are developing a media player application that handles different types of media files: Audio, Video, and Image. All these media types extend a base class Media.

Requirements:

1. Create a base class Media with:
    - A method play() that prints "Playing media".
2. Create three subclasses:
    - Audio overriding play() to print "Playing audio file".
    - Video overriding play() to print "Playing video file".
    - Image overriding play() to print "Displaying image file".
3. In your media player, you store all media objects in an array of Media references.
4. Write a method processMedia(Media m) that:
    - Calls play() on the media.
    - Uses **downcasting** to check the actual media type at runtime.
    - If the media is an instance of Video, cast it to Video and call a unique method displaySubtitle() (which you need to add in the Video class, printing "Displaying subtitles").
    - If the media is an instance of Audio, cast it to Audio and call a unique method adjustVolume() (which prints "Adjusting audio volume").
    - If the media is an Image, cast it to Image and call a unique method applyFilter() (which prints "Applying filter to image").
5. Demonstrate the following in your main program:
    - Create objects of each media type and store them in a Media[] array.
    - Iterate over the array and call processMedia() for each item.