CRUD data handling
Dialog Widget

- Revision
- List<Art>
- DartPad – CRUD with Data Service class
- Art App (Enter New Art)
- Art App (Show Art Details – GestureDetector Widget)
- Art App (Delete Art)
- Art App (Delete Art with Dialog Widget)

Revision

Recap

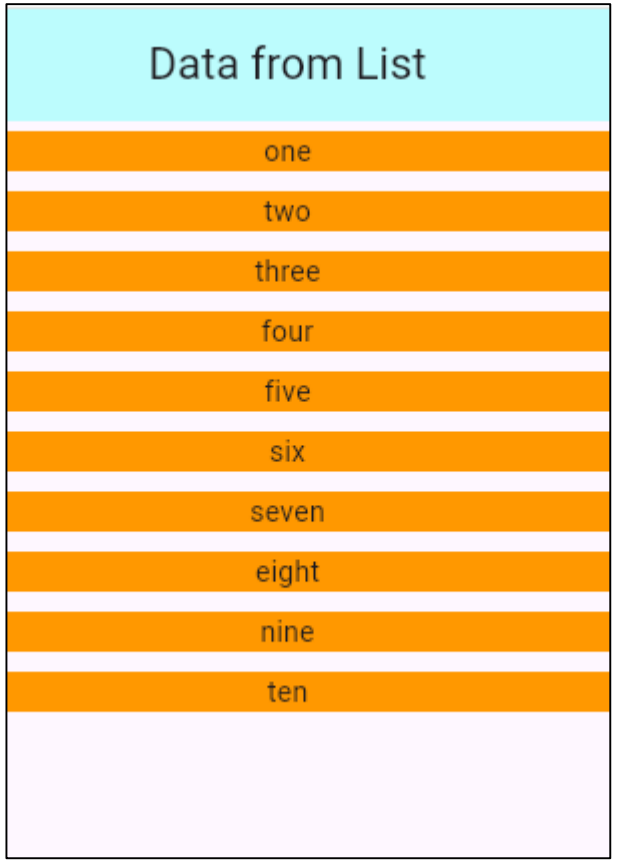
Recap: ListView Widget – Dynamic data (Topic 3)

Instead of always displaying the same text ‘hello’ in all the 10 Padding Widget, the text can be made as a variable.

```
List<String> x = ['one', 'two','three','four','five',
                  'six','seven','eight','nine','ten'];
```

The value for the variable can then be fed from a List by index.

```
x[index]
```



Recap: ListView Widget – Dynamic data (Topic 3)

```
class _PlayItemBuilderState extends State<PlayItemBuilder> {
  List<String> x = ['one', 'two','three','four','five',
                    'six','seven','eight','nine','ten'];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Data from List'),
        centerTitle: true,
        backgroundColor: Color.fromARGB(252, 187, 252, 253),
      ),
      body: ListView.builder(
          itemCount: x.length,
          itemBuilder: (context, index) {
            return Padding(
              padding: const EdgeInsets.all(10),
              child: Container(
                  color: Colors.orange,
                  child: const Center(child: Text( x[index]  ))),
            );
          }),
    );   }  }
```

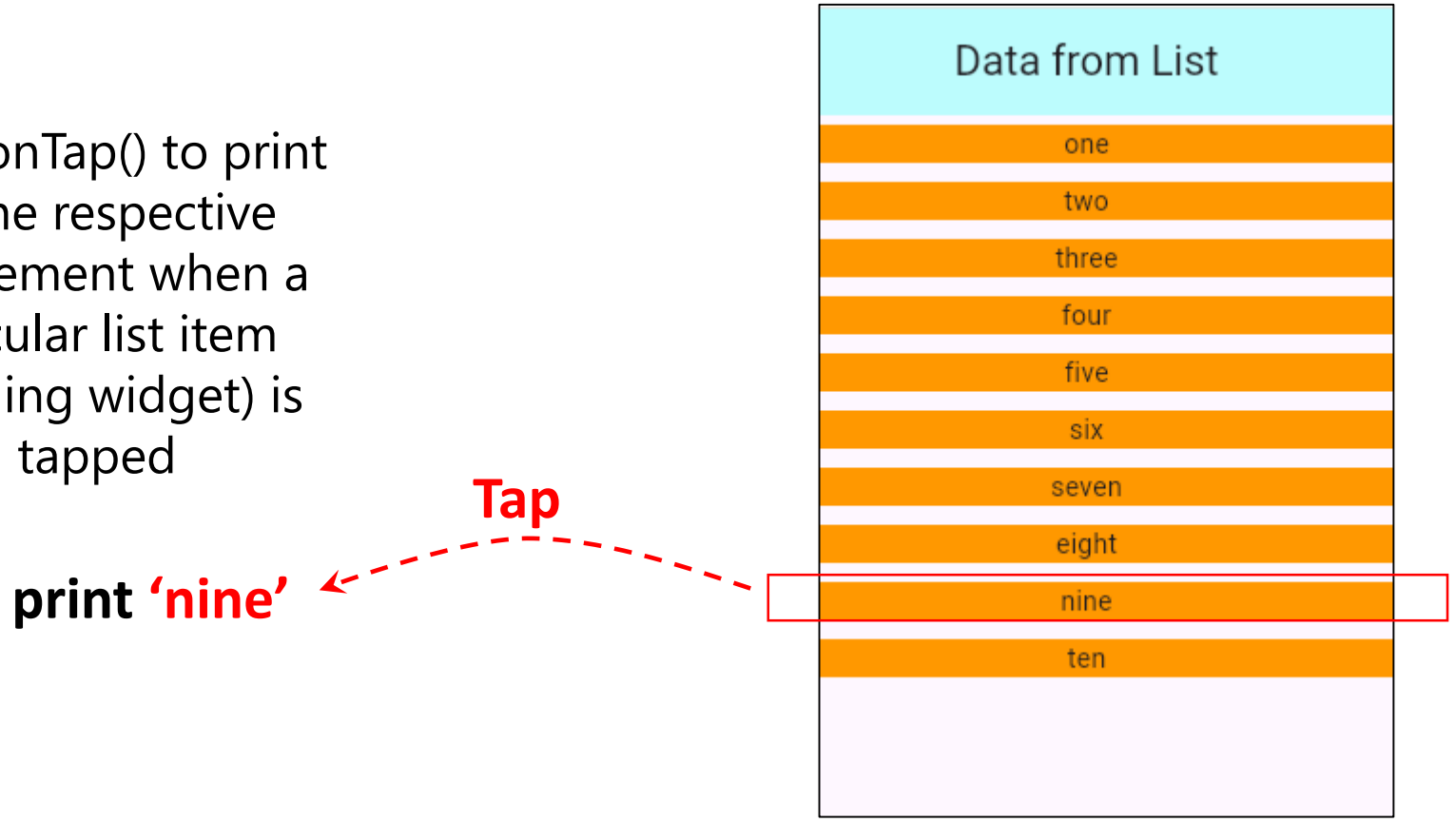
Instead of always displaying the same text 'hello' in all the 10 Padding Widget, the text can be made as a variable.

The value for the variable can then be fed from a List by index.

Recap

Apply onTap() to Padding in ListView (Topic 3)

Add onTap() to print out the respective list element when a particular list item (Padding widget) is being tapped



Apply onTap() to Padding in ListView (Topic 3)

Recap

Edit the code inside GestureDetector.

Add onTap() to print out the respective list element when a particular list item (Padding widget) is being tapped

```
class _PlayItemBuilderState extends State<PlayItemBuilder> {

  List<String> x = ['one', 'two','three','four','five',
                    'six','seven','eight','nine','ten'];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Data from List '),
        centerTitle: true,
        backgroundColor: Color.fromARGB(252, 187, 252, 253),
      ),
      body: ListView.builder(
        itemCount: x.length,
        itemBuilder: (context, index) {
          return GestureDetector(
            onTap: (){
              print(x[index]);
            },
            child: Padding(
              padding: const EdgeInsets.all(5),
              child: Container(
                color: Colors.orange,
                child: Center(child: Text(x[index])),
              ),
            ),          );        },      ),    );  }  }
```

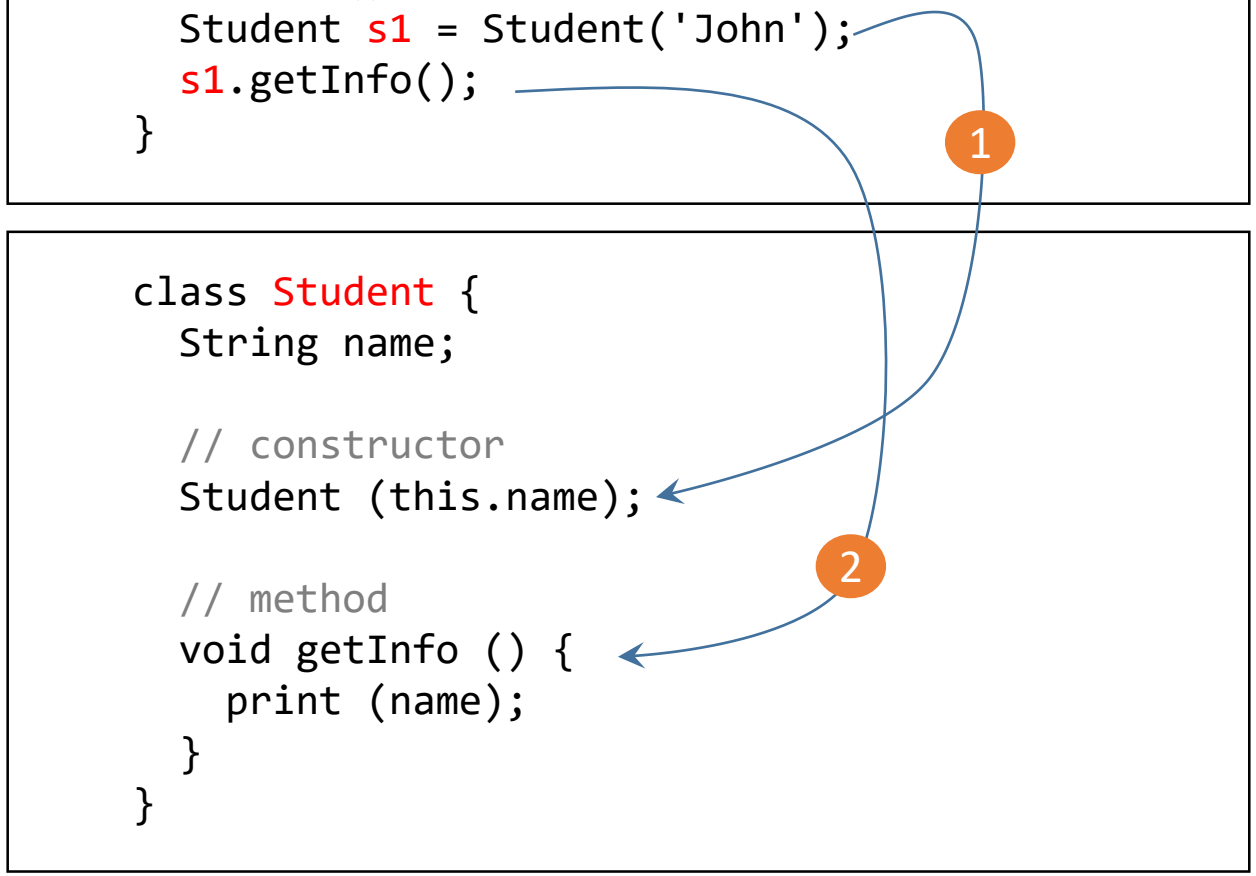
Dart Basics – Class (Topic 1)

Recap

Flow of the triggering the constructor and method within the Student object.

```
void main() {
  Student s1 = Student('John');
  s1.getInfo();
}
```

```
class Student {
  String name;

  // constructor
  Student (this.name);

  // method
  void getInfo () {
    print (name);
  }
}
```



1. Object s1 is created, with the constructor setting the name for s1 as 'John'.

2. method **getInfo()** in object **s1** is being invoked and the name in object **s1** is being printed: **John**

In this topic, an Art class is used. It is defined as:

```
class Art {
    String artId="", title="",artist="",image="";

    // Constructor
    // will run when creating an Art object
    Art (this.artId, this.title, this.artist, this.image);
}
```

It has 4 attributes (or properties), all Strings, artId, title, artist and image.

The constructor takes in 4 input parameters (all Strings) in the sequence of artId, title, artist and image.

The following state creates an Art object **a1** with values for respective variables:

```
Art a1 = Art("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png");
```

With the above statement, the following statement then retrieves title from the Art object **a1** :

```
print (a1.title);   // Woman with a Parasol
```

TextField and TextEditingController (Topic 3)

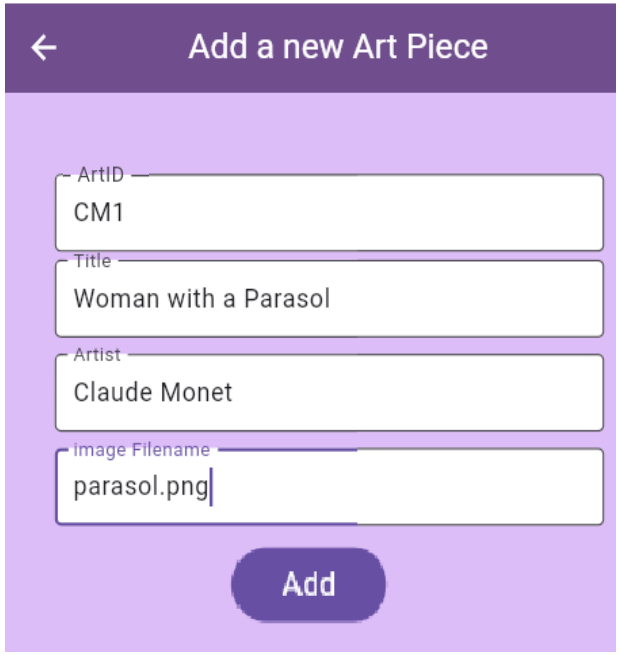
Every TextField requires its own TextEditingController

```
TextEditingController controllerId = TextEditingController();
```

```
TextField(
  controller: controllerId,
  decoration: const InputDecoration(
      fillColor: Colors.white, filled: true,
      labelText: "ArtId", border: OutlineInputBorder()),
),
```

```
String art_id = controllerId.text;
```
// to capture the input from the TextField

In this topic, there is a page where user can enter 4 pieces of input data via TextFields for an Art object.




```
TextEditingController controllerId = TextEditingController();
TextEditingController controllerTitle = TextEditingController();
TextEditingController controllerArtist = TextEditingController();
TextEditingController controllerImage = TextEditingController();
```

```
TextField(
  controller: controllerId,
  decoration: const InputDecoration(
      fillColor: Colors.white, filled: true,
      labelText: "ArtID", border: OutlineInputBorder()),
),
const SizedBox(height: 5),
TextField(
  controller: controllerTitle,
  decoration: const InputDecoration(
      fillColor: Colors.white, filled: true,
      labelText: "Title", border: OutlineInputBorder()),
),
const SizedBox(height: 10),
TextField(
  controller: controllerArtist,
  decoration: const InputDecoration(
      fillColor: Colors.white, filled: true,
      labelText: "Artist", border: OutlineInputBorder()),
),
const SizedBox(height: 10),
TextField(
  controller: controllerImage,
  decoration: const InputDecoration(
      fillColor: Colors.white, filled: true,
      labelText: "Image Filename", border: OutlineInputBorder()),
),
```

End of Revision

List<Art>

List <>

List<String> x stores simple strings:

```
List<String> x = ['one', 'two','three','four','five',
                  'six','seven','eight','nine','ten'];
```

To store a list of integers:

```
List<int> y = [123, 888];
```

List can be used to store more complex data, such as objects.
We will use it to store *Art* objects (*Art* is a custom class that is defined by us)

```
List<Art> z = [];
```

List – commonly used methods

Some of the commonly operations on List:

```
List<String> x  = [];     // declaration
x.add("SP");              // add new element
x.add("NP");
x[0] = "NYP";             // update element by index
x.removeAt(1);            // remove element by index
print (x.length);         // returns no of elements
```

List<Art>

The following statements creates an empty List to store Art objects:

```
List<Art> z = [];
```

```
class Art {
   String artId="", title="",artist="",image="";

    // Constructor
    // will run when creating an Art object
    Art (this.artId, this.title, this.artist, this.image);
}
```

List<Art>

The following statements creates an **Art** object a1, with the 4 input strings and then adds it into the List<Art> z :

```
List<Art> z = [];
Art a1 = Art("CM1", "Woman with a Parasol",
             "Claude Monet", "parasol.png");
z.add(a1);
```

```
class Art {
   String artId="", title="",artist="",image="";

    // Constructor
    // will run when creating an Art object
    Art (this.artId, this.title, this.artist, this.image);
}
```

List<Art>

Alternatively, the logic in previous slide can be written *in one single statement*.

```
class Art {
   String artId="", title="",artist="",image="";

    // Constructor
    // will run when creating an Art object
    Art (this.artId, this.title, this.artist, this.image);
}
```

```
List<Art> z = [];

z.add( Art("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png") );
```

Adds the **Art** object into List<Art> z

creates an **Art** object with the 4 input strings

(We will use this format later)

DartPad
(CRUD with Data Service class)

Data

- Flutter app is just an empty shell (or UI) if it is without *data*.
- Data storage can be temporary or persistent.
  - Temporary - Using variables (such as List) to store data – they are temporary as the data will be gone once the app is turned off
  - Persistent - can be in the form of file or database. It can be local in the device or in the cloud
- Topic 5 – focuses on temporary storage, using List<Art> to store Art objects
- Topic 6 – expanding to store data in cloud database (Google's Firebase)

DartPad – the CRUD operations on List<>

Before coding in VS code, we will use DartPad to get familiar with:

- Create objects of class Art and add it into List<Art>
- Retrieve Art object from the List<Art>
- Update the data in Art object inside the List<Art>
- Delete Art object from the List<Art>

Commonly known as CRUD operations, when dealing with data storage.

https://dartpad.dev/

DartPad –the Data Service class

- More often than not, data handling (especially interfaces to the database) is responsible by a specially designated class.
- All the necessary CRUD operations is performed by this class with related functions.
- Hence, this type of classes are often called 'Service' class or 'Helper class'
- This program structure paves a smoother transition into the adoption of database for apps later in Topic 6

DartPad – class ArtDataService

- The class that we are going to create is called **ArtDataService**
- It consists of `List<Art>` which stores the Art objects created

```
class ArtDataService {

  List<Art> z = [];   // storage of Art objects


}
```

DartPad – class ArtDataService

- It also has a handy method getCount(), which returns the number of elements in **z** currently.

```
class ArtDataService {

  List<Art> z = [];   // storage of Art objects

   getCount()  // returns the no of elements in z

}
```

DartPad – class ArtDataService

- More importantly, it has **CRUD** data manipulation functions that add/read/update/delete elements in the List<Art>:

```
class ArtDataService {

  List<Art> z = [];   // storage of Art objects

   getCount() - returns the no of elements in z

   addArt() - takes in 4 strings to create an Art object and add it into the z
   getArtAt() - returns the Art object read for a given index in z
   getArtByArtId() - returns the Art object read for a given art Id
   updateArtAt() - takes in index and 3 strings to update Art object at index
   updateArtByArtId() - takes in art id and 3 strings o update Art object with the art id
   removeArtAt() - deletes the Art object for a given index
   removeArtByArtId() - deletes the Art object for a given art Id
}
```

DartPad –the Data Service class

- We will get started in DartPad
- Familiarise with the ArtDataService class using DartPad
- Then, move on to VS Code to incorporate the ArtDataService with various screens
- Let us now create the class ArtDataService in Dartpad, and add in the CRUD methods progressively.

DartPad – program structure

- Go to https://dartpad.dev/

- In DartPad, this is the program structure that we are going to use.
- There are main(), class Art and class ArtDataService

```
void main() {

}

class Art {

}

class ArtDataService {

}
```

DartPad – class Art

- Firstly, complete the class Art. It is the same as the class shown in earlier slides.

```
void main() {

}

class Art {
   String artId="", title="",artist="",image="";

   // Constructor
   // will run when creating an Art object
   Art (this.artId, this.title,  this.artist,  this.image);
}

class ArtDataService {

}
```

DartPad – class ArtDataService

- The following slides focus on the building of the class ArtDataService
- Due to constraint of space, the main() and class Art will be omitted in some of the slides but they are still part of the program

DartPad – class ArtDataService

Firstly, declare an empty List<Art>

```
class ArtDataService {
   static   List<Art> z = [];
}
```

DartPad –class ArtDataService

- Next, add the first static method getCount() to class ArtDataService
- It returns the number of elements in the List<Art> z currently

```
class ArtDataService {
  static   List<Art> z = [];

  static int getCount() {
    return (z.length);
  }

}
```

* All the **static** keywords are crucial

DartPad –class ArtDataService

With getCount() defined, try this in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
}
```

```
class ArtDataService {
  static   List<Art> z = [];

  static int getCount() {
    return (z.length);
  }

}
```

DartPad –class ArtDataService

With getCount() defined, try this in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
}
```

```
class ArtDataService {
  static   List<Art> z = [];

  static int getCount() {
    return (z.length);
  }

}
```

DartPad –class ArtDataService

It is possible to call the method *getCount()* directly by using the class name, i.e. *ArtDataService.getCount()*, as the method is defined with the **static** keyword

```
void main() {
    print (ArtDataService.getCount());  // 0
}
```

```
class ArtDataService {
  static   List<Art> z = [];

  static int getCount() {
    return (z.length);
  }

}
```

DartPad – class ArtDataService

- Next, add the static method addArt()
- The method takes in 4 Strings and uses them to create an Art object, then adds the newly created Art object into List<Art> z

```
class ArtDataService {
    static   List<Art> z = [];

    static int getCount() { return (z.length); }

    static void addArt (String artId, String artTitle, String artArtist, String artImage){
       z.add( Art( artId, artTitle, artArtist, artImage));
    }


}
```

* All the **static** keywords are crucial

Adds the **Art** object into List<Art> z

creates an **Art** object with the 4 input strings

DartPad – class ArtDataService

With addArt() defined, try this in main():

```
void main() {
    print (ArtDataService.getCount());  // 0

    // create 2 Art objects in List<Art>
    ArtDataService.addArt("CM1", "Woman with a Parasol", "Claude Monet",
    "parasol.png");
    ArtDataService.addArt("VG1", "Sunflowers", "Vincent van Gogh", "sunflowers.png");

    print (ArtDataService.getCount());  // 2
}
```

```
class ArtDataService {
    static   List<Art> z = [];

    static int getCount() { return (z.length); }

    static void addArt (String artId, String artTitle, String artArtist, String artImage){
       z.add( Art( artId, artTitle, artArtist, artImage));
    }
}
```

DartPad – class ArtDataService

- Next, add the static method getArtAt()
- The method takes in index (which is integer) and retrieves the Art object using the index. The overly simplified code here does not check if the index is out of bound.

```
class ArtDataService {
    static   List<Art> z = [];

    static int getCount() { return (z.length); }

    static void addArt (String artId, String artTitle, String artArtist, String artImage){
       z.add( Art( artId, artTitle, artArtist, artImage));
    }

    static Art getArtAt(int index){
       return (z[index]);
    }
}
```

* All the **static** keywords are crucial

DartPad – class ArtDataService

With getArtAt() defined, try this in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
    // create 2 Art objects in List<Art>
    ArtDataService.addArt("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png");
    ArtDataService.addArt("VG1", "Sunflowers", "Vincent van Gogh", "sunflowers.png");

    print (ArtDataService.getCount());  // 2

    print (ArtDataService.getArtAt(0).title);    // Woman with a Parasol
    print (ArtDataService.getArtAt(1).artist);   // Vincent van Gogh
}
```

DartPad – class ArtDataService

Next, add the static method getArtByArtId(). The method takes in an artId (string), scans through the List<Art> z to find a match. If there is a match, it returns the matched Art object. Otherwise, it returns null. As there is a possibility of returning null, a ? has to be added at the return type Art?

```
class ArtDataService {
    static   List<Art> z = [];
    static int getCount() { return (z.length); }
    static void addArt (String artId, String artTitle, String artArtist, String artImage){
       z.add( Art( artId, artTitle, artArtist, artImage));
    }
    static Art getArtAt(int index){ return (z[index]); }

    static Art? getArtByArtId (String id){
        // loop to match artId
        for (int i=0; i<z.length; i++){
          if (z[i].artId == id){
            return z[i];
          }
        }
        return null;
    }
}
```

* All the **static** keywords are crucial

DartPad – class ArtDataService

With getArtByArtId() defined, try this, with a valid artId, in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
    // create 2 Art objects in List<Art>
    ArtDataService.addArt("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png");
    ArtDataService.addArt("VG1", "Sunflowers", "Vincent van Gogh", "sunflowers.png");
    print (ArtDataService.getCount());  // 2
    print (ArtDataService.getArtAt(0).title);   // Woman with a Parasol
    print (ArtDataService.getArtAt(1).artist);  // Vincent van Gogh

    print (ArtDataService.getArtByArtId("VG1").image);

}
```

Error!
As there is a possibility that getArtByArtId() will return null, a ? must be added before the dot

```
print (ArtDataService.getArtByArtId("VG1")?.image); // sunflowers.png
```

DartPad – class ArtDataService

Try again, with an invalid artId, in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
    // create 2 Art objects in List<Art>
    ArtDataService.addArt("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png");
    ArtDataService.addArt("VG1", "Sunflowers", "Vincent van Gogh", "sunflowers.png");
    print (ArtDataService.getCount());  // 2
    print (ArtDataService.getArtAt(0).title);   // Woman with a Parasol
    print (ArtDataService.getArtAt(1).artist);  // Vincent van Gogh

    print (ArtDataService.getArtByArtId("vvvvv")?.image); // null

}
```

Invalid artId

As such, getArtByArtId() will return null,

DartPad – class ArtDataService

Next, add two more static methods to handle the update of Art objects in List<> z. Again, the code does not check for invalid index.

```
class ArtDataService {
    static   List<Art> z = [];
    static int getCount() { return (z.length); }
    static void addArt (String artId, String artTitle, String
    artArtist, String artImage){
       z.add( Art( artId, artTitle, artArtist, artImage));
    }
    static Art getArtAt(int index){ return (z[index]); }

    static Art? getArtByArtId (String id){
        // loop to match artId
        for (int i=0; i<z.length; i++){
          if (z[i].artId == id){
            return z[i];
          }
        }
        return null;
    }
```

```
    static void updateArtAt(int index, String
    newTitle, String newArtist, String newImage){
            z[index].title = newTitle;
            z[index].artist = newArtist;
            z[index].image = newImage;
    }

    static void updateArtByArtId(String id,
                                 String newTitle,
                                 String newArtist,
                                 String newImage) {
        for (int i = 0; i < z.length; i++) {
          if (z[i].artId == id) {
            z[i].title = newTitle;
            z[i].artist = newArtist;
            z[i].image = newImage;
          }
        }
    }
}
```

* All the **static** keywords are crucial

DartPad – class ArtDataService

With the 2 update methods added, try this in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
    // create 2 Art objects in List<Art>
    ArtDataService.addArt("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png");
    ArtDataService.addArt("VG1", "Sunflowers", "Vincent van Gogh", "sunflowers.png");
    print (ArtDataService.getCount());  // 2
    print (ArtDataService.getArtAt(0).title);    // Woman with a Parasol
    print (ArtDataService.getArtAt(1).artist);   // Vincent van Gogh
    print (ArtDataService.getArtByArtId("VG1").image); // sunflowers.png
    // update info for Art at index = 0
    ArtDataService.updateArtAt(0,"Starry Night","Vincent van Gogh", "starry.png");
    print("Updated 1st element's title to \'${ArtDataService.getArtAt(0).title}\' ");

}
```

```
Updated 1st element's title to 'Starry Night'
```

DartPad – class ArtDataService(Full Code)

Next, complete the class ArtDataService with two more static methods to handle the removal of Art objects in List<> z.

```
class ArtDataService {

  static   List<Art> z = [];

  static int getCount() { return (z.length); }

  static void addArt (String artId, String artTitle,
                String artArtist, String artImage){
          z.add( Art( artId, artTitle,
                 artArtist, artImage));
  }

  static Art getArtAt(int index){
     return (z[index]);
  }

  static Art? getArtByArtId (String id){
      // loop to match artId
      for (int i=0; i<z.length; i++){
        if (z[i].artId == id){
          return z[i];
        }
      }
      return null;
  }

  static void updateArtAt(int index, String
  newTitle, String newArtist, String newImage){
            z[index].title = newTitle;
            z[index].artist = newArtist;
            z[index].image = newImage;
  }

  static void updateArtByArtId(String id,
                               String newTitle,
                               String newArtist,
                               String newImage) {
      for (int i = 0; i < z.length; i++) {
        if (z[i].artId == id) {
            z[i].title = newTitle;
            z[i].artist = newArtist;
            z[i].image = newImage;
      }
    }
  }

  static void removeArtAt(int index){
        z.removeAt(index);
  }
  static void removeArtByArtId(String id) {
      for (int i = 0; i < z.length; i++) {
        if (z[i].artId == id) {
          z.removeAt(i);
        }
      }
  }
}
```

* All the **static** keywords are crucial

DartPad – class ArtDataService

With the complete ArtDataSrervice, try this in main():

```
void main() {
    print (ArtDataService.getCount());  // 0
    // create 2 Art objects in List<Art>
    ArtDataService.addArt("CM1", "Woman with a Parasol", "Claude Monet", "parasol.png");
    ArtDataService.addArt("VG1", "Sunflowers", "Vincent van Gogh", "sunflowers.png");
    print (ArtDataService.getCount());  // 2
    print (ArtDataService.getArtAt(0).title);   // Woman with a Parasol
    print (ArtDataService.getArtAt(1).artist);  // Vincent van Gogh
    print (ArtDataService.getArtByArtId("VG1").image); // sunflowers.png
    // update info for Art at index = 0
    ArtDataService.updateArtAt(0,"Starry Night","Vincent van Gogh", "starry.png");
    print("Updated 1st element's title to \'${ArtDataService.getArtAt(0).title}\' ");

    // delete Art by index=0
    ArtDataService.removeArtAt(0);

    // check no of elements in List<Art> again
    print("No of elements: ${ArtDataService.getCount()}");
}
```

```
No of elements: 1
```

DartPad –class ArtDataService

- We have completed constructing the class ArtDataService
- We will use it in our Art App later in VS Code to
  - create storage of Art information entered by the user in TextFields
  - retrieve the Art information stored and display in ListView
  - update existing Art information
  - delete existing Art information
- All these CRUD operations can be done by calling the respective method in ArtDataService using this syntax:

```
ArtDataService.(method)
```

Back to Visual Studio Code

This is a series of continuous projects

art_add_new

art_details

art_delete

art_dialog (Ex 5-1)

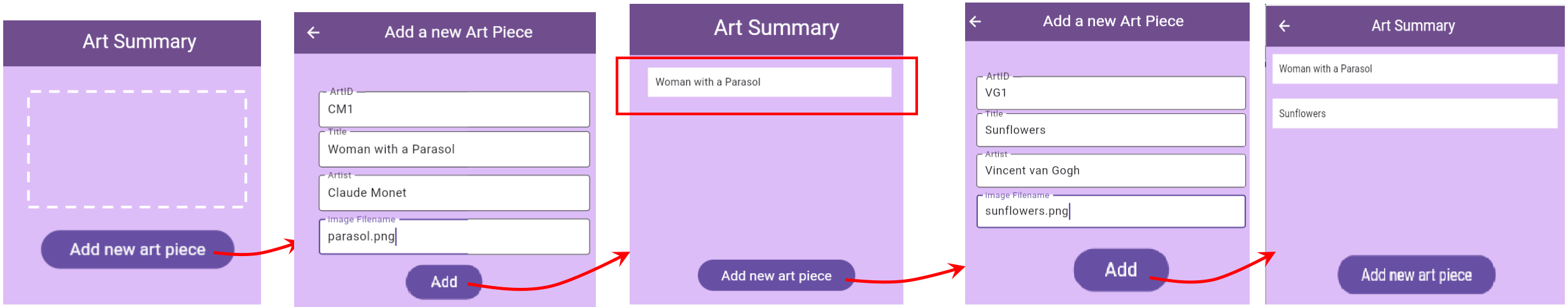
Next topic, Topic 6, on Firebase (database) will build on the last project: art_dialog (Ex 5-1)

Art App (Enter New Art)

Project: art_add_new

About the app

This is a simple app with 2 screens – *Art Summary* and *Add a new Art Piece*.
Default home screen *Art Summary* displays a ListView of titles of arts (initially empty).
Add a new Art Piece allows the user to enter new art pieces.



Initial display (empty list) when project is run. Click button to next screen.

Enter info for 1st Art object. Click Add button to return.

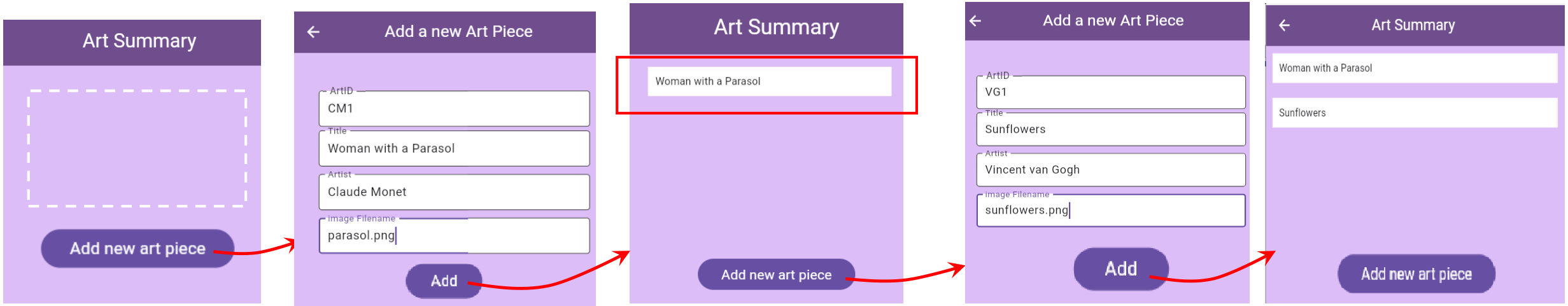
The ListView is updated with the 1st Art object added.

Repeat to enter info for 2nd Art object.

The ListView after 2 Art added.

Repeat to add a few more Art objects.

The data



Enter info for 1st Art object. Click Add button to return.

This is where Art object is created and added to List<Art>.

The ListView is updated with the 1st Art object added.

This is where Art object is being retrieved from List<Art> and pushed to UI

Repeat to enter info for 2nd Art object.

This is where Art object is created again and added to List<Art>.

The ListView after 2 Art added.

This is where Art object is being retrieved from List<Art> and pushed to UI

In Visual Studio Code

- Create a new Flutter project **art_add_new**
- 5 files (only **artsummary.dart** and **newart.dart** have Scaffold, i.e. user interface)

main.dart

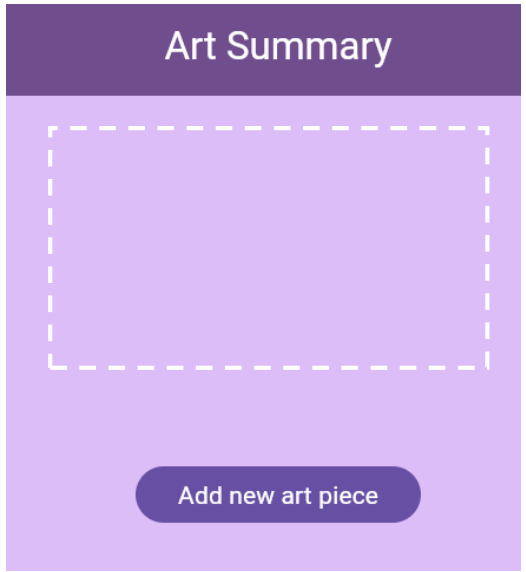
```
void main() {
  runApp(MaterialApp(
     home: ArtSummary(),
  ));
}
```

art.dart

```
class Art {
   (same as Dartpad example)
}
```

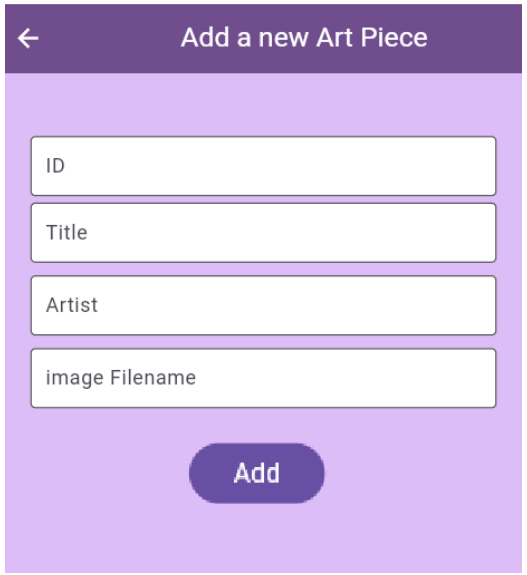
artdataservice.dart

```
class ArtDataService {
   (same as Dartpad example)
}
```



artsummary.dart

Clicking the button will switch to ***newart.dart***

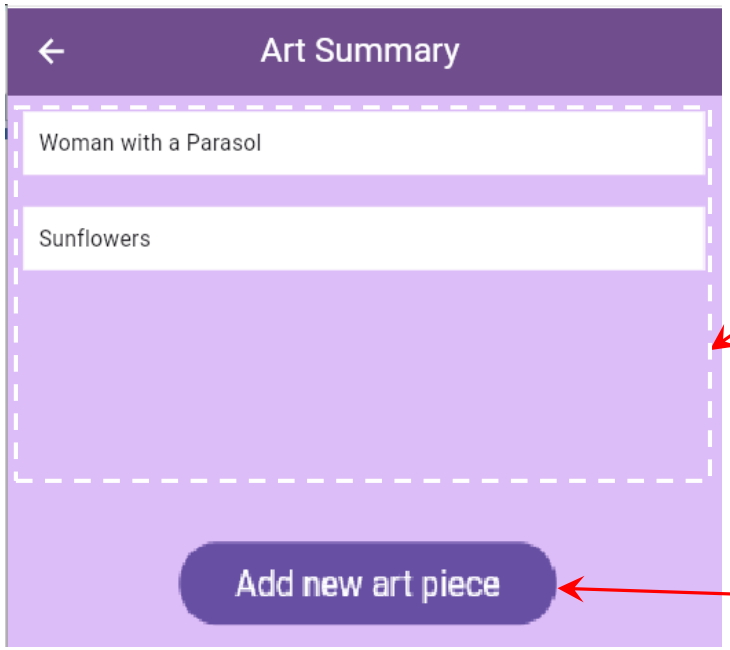


newart.dart

Enter the art info and click the button will switch back to ***artsummary.dart*** with updated list

Functional Requirements (artsummary.dart)

artsummary.dart – This is the "home page". It is similar to ListView in Topic 3 but with data handling of List<Art> (instead of List<String>) and an additional "Add new art piece" Button.



ListView displaying ***title*** from the content of **List<Art>,** initially it is blank.

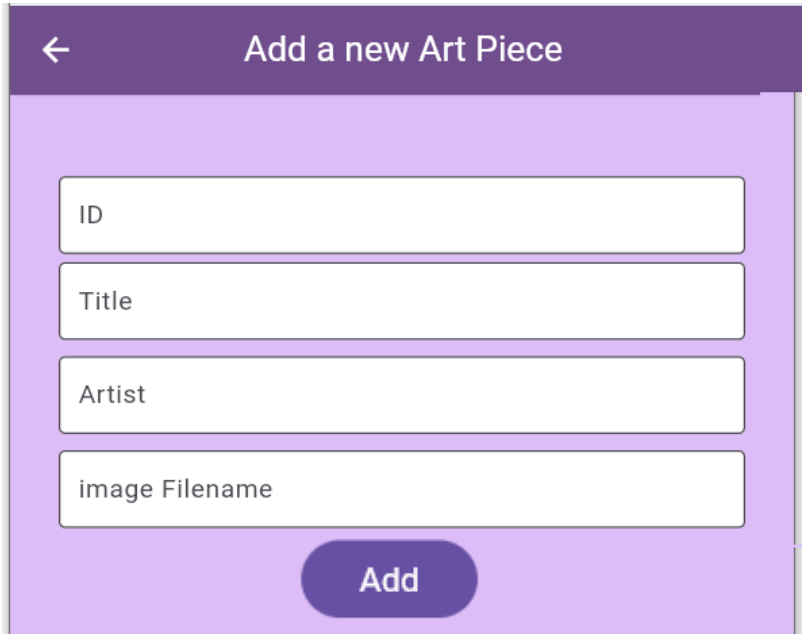
As more new Art objects are added, the ListView will display more ***titles*** here. (This is handled by ***ListView.builder***)

Switch to **newart.dart** with *Navigator.pushNamed(context, "/newartpage")*

Functional Requirements (newart.dart)

newart.dart

- has 4 TextFields to allow input of art info and an ***Add*** button
- When the ***Add*** button is clicked
  - Read in all the input TextField
  - Use the inputs to create new an Art object
  - Add the new object into List<Art>
  - Navigate back to ***artsummary.dart***



art.dart (Full code)

(same as the class Art we coded in DartPad)

art.dart

```
class Art {
   String artId="", title="",artist="",image="";

  // Constructor - will run when creating an Art object
   Art (this.artId, this.title,  this.artist,  this.image);
}
```

artdataservice.dart(Full Code)

(same as the class ArtDataService we coded in DartPad)

artdataservice.dart

```
class ArtDataService {

   static   List<Art> z = [];

   static int getCount() { return (z.length); }

   static void addArt (String artId, String artTitle,
                String artArtist, String artImage){
          z.add( Art( artId, artTitle,
                 artArtist, artImage));
   }

   static Art getArtAt(int index){
      return (z[index]);
   }

   static Art? getArtByArtId (String id){
       // loop to match artId
       for (int i=0; i<z.length; i++){
         if (z[i].artId == id){
           return z[i];
         }
       }
       return null;
   }
   static void updateArtAt(int index, String
   newTitle, String newArtist, String newImage){
            z[index].title = newTitle;
            z[index].artist = newArtist;
            z[index].image = newImage;
   }

   static void updateArtByArtId(String id,
                              String newTitle,
                              String newArtist,
                              String newImage) {
       for (int i = 0; i < z.length; i++) {
         if (z[i].artId == id) {
            z[i].title = newTitle;
            z[i].artist = newArtist;
            z[i].image = newImage;
       }
     }
   }

   static void removeArtAt(int index){
         z.removeAt(index);
   }
   static void removeArtByArtId(String id) {
       for (int i = 0; i < z.length; i++) {
         if (z[i].artId == id) {
           z.removeAt(i);
         }
       }
   }
}
```

artsummary.dart (Full Code)

```dart
import 'package:flutter/material.dart';

class ArtSummary extends StatefulWidget {
  const ArtSummary({super.key});

  @override
  State<ArtSummary> createState() => _ArtSummaryState();
}

class _ArtSummaryState extends State<ArtSummary> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color.fromARGB(255, 221, 189, 247),
      appBar: AppBar(
        title: const Text("Art List"),
        backgroundColor: Color.fromARGB(255, 112, 78, 142),
        foregroundColor: Colors.white,
        centerTitle: true,
      ),
```

artsummary.dart

```dart
body: Column(children: [
  SizedBox(
    height: 200,
    child: ListView.builder(
      itemCount: ArtDataService.getCount(),
      itemBuilder: (context, index) {
        return Padding(
          padding: EdgeInsets.all(10),
          child: Container(
            padding: EdgeInsets.all(10),
            color: Colors.white,
            child: Text(ArtDataService.getArtAt(index).title),
          ),
        );
      },
    ),
  ),
  SizedBox(height: 10),
  FilledButton(
      child: Text("Add new art piece"),
      onPressed: () {
        Navigator.pushNamed(context, "/newartpage");
      })
]),    );  }  }
```

newart.dart (Full code)

newart.dart

```dart
import 'package:flutter/material.dart';
class NewArt extends StatefulWidget {
  const NewArt({super.key});

  @override
  State<NewArt> createState() => _NewArtState();
}

class _NewArtState extends State<NewArt> {

  TextEditingController controllerId = TextEditingController();
  TextEditingController controllerTitle = TextEditingController();
  TextEditingController controllerArtist = TextEditingController();
  TextEditingController controllerImage = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color.fromARGB(255, 221, 189, 247),
      appBar: AppBar(
        title: const Text('Add a new Art Piece'),
        backgroundColor: Color.fromARGB(255, 112, 78, 142),
        foregroundColor: Colors.white,
        centerTitle: true,
      ),
      body: Container(
        padding: const EdgeInsets.all(30),
        child: Column(children: [
          SizedBox(height: 20),
          TextField(
            controller: controllerId,
            decoration: const InputDecoration(
                fillColor: Colors.white, filled: true,
                labelText: "ArtID", border: OutlineInputBorder()),
          ),
          const SizedBox(height: 5),
          TextField(
            controller: controllerTitle,
            decoration: const InputDecoration(
                fillColor: Colors.white, filled: true,
                labelText: "Title", border: OutlineInputBorder()),
          ),
          const SizedBox(height: 10),
          TextField(
            controller: controllerArtist,
            decoration: const InputDecoration(
                fillColor: Colors.white, filled: true,
                labelText: "Artist", border: OutlineInputBorder()),
          ),
          const SizedBox(height: 10),
          TextField(
            controller: controllerImage,
            decoration: const InputDecoration(
                fillColor: Colors.white, filled: true,
                labelText: "Image Filename", border: OutlineInputBorder()),
          ),
          const SizedBox(height: 10),
          FilledButton( child: const Text('Add'),
            onPressed: () {
              ArtDataService.addArt (controllerId.text,
                controllerTitle.text, controllerArtist.text,
                controllerImage.text);
              Navigator.pushNamed(context, "/artsummarypage");
            }),
        ]),
      ),
    );
  }
}
```

main.dart (Full Code)

main.dart

- the **home** page is set **ArtSummary()**
- the **routes** defined to facilitates the navigation needed later when switching from one screen to another.

main.dart

```
void main() {
  runApp(MaterialApp(
  home: ArtSummary(),
  routes: {
    "/newartpage": (context) => NewArt(),
    "/artsummarypage": (context) => ArtSummary(),
  }));
}
```

named routes

The following slides are code highlights for artsummary.dart and newart.dart

Code highlights (artsummary.dart)

ListView.builder – displays the titles of elements in List<Art>.

`ArtDataService.getCount()`
(no of elements in List<Art>)

`ArtDataService.getArtAt(index).title`
(title of the current Art object at position index in List<Art>)

Name route defined in main is used here to switch to newart.dart

artsummary.dart

```
body: Column(children: [
  SizedBox(
    height: 200,
    child: ListView.builder(
      itemCount: ArtDataService.getCount(),
      itemBuilder: (context, index) {
        return Padding(
          padding: EdgeInsets.all(10),
          child: Container(
            padding: EdgeInsets.all(10),
            color: Colors.white,
            child: Text(ArtDataService.getArtAt(index).title),
          ),
        );
      },
    ),
  ),
  SizedBox(height: 10),
  FilledButton(
      child: Text("Add new art piece"),
      onPressed: () {
        Navigator.pushNamed(context, "/newartpage");
      })
]),    );  }  }
```

Code highlights (newart.dart) – the 4 TextFields

```
TextEditingController controllerId = TextEditingController();
TextEditingController controllerTitle = TextEditingController();
TextEditingController controllerArtist = TextEditingController();
TextEditingController controllerImage = TextEditingController();
```

Controllers for the 4 TextFields

```
TextField(
   controller: controllerId,
   decoration: const InputDecoration(
       fillColor: Colors.white, filled: true,
       labelText: "ArtID", border: OutlineInputBorder()),
),
```

This is one of the TextFields (artId). The other 3 TextFields are similar.

Code Highlights (newart.dart) – the Add button

When the ***Add*** button is clicked:

ArtDataService.addArt()
Create an Art object with all the inputs from the TextFields and add it into List<Art>

Return to home page

newart.dart

```
FilledButton( child: const Text('Add'),
  onPressed: () {
    ArtDataService.addArt ( controllerId.text,
      controllerTitle.text, controllerArtist.text,
      controllerImage.text);

    Navigator.pushNamed(context, "/artsummarypage");

  },   ),  }
```

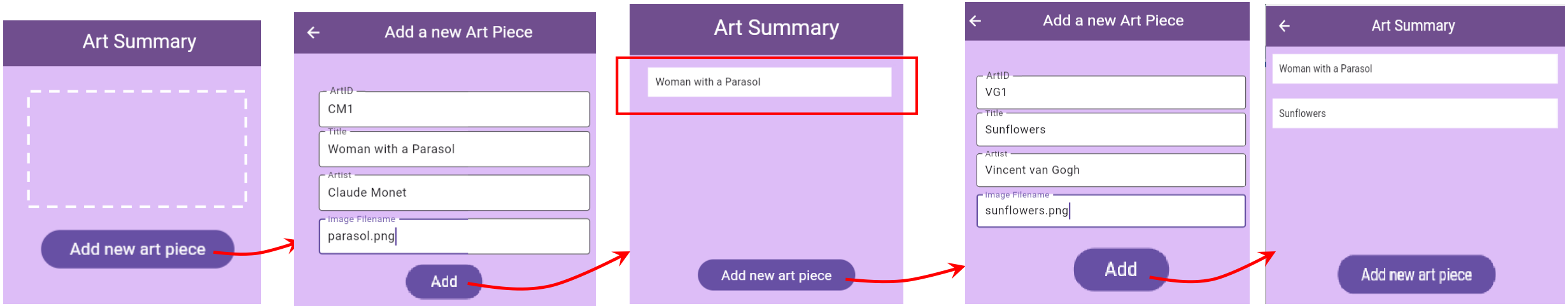
Art data

As we are going to type these data many times during testing, it might be handy if we put them in a notepad file (text file):

CM1
Woman with a Parasol
Claude Monet
parasol.png

VG1
Sunflowers
Vincent van Gogh
sunflowers.png

Run the project



Initial display (empty list) when project is run. Click button to next screen.

Enter info for 1st Art object. Click Add button to return.

CM1
Woman with a Parasol
Claude Monet
parasol.png

The ListView is updated with the new Art object added.

Repeat to enter info for 2nd Art object.

VG1
Sunflowers
Vincent van Gogh
sunflowers.png

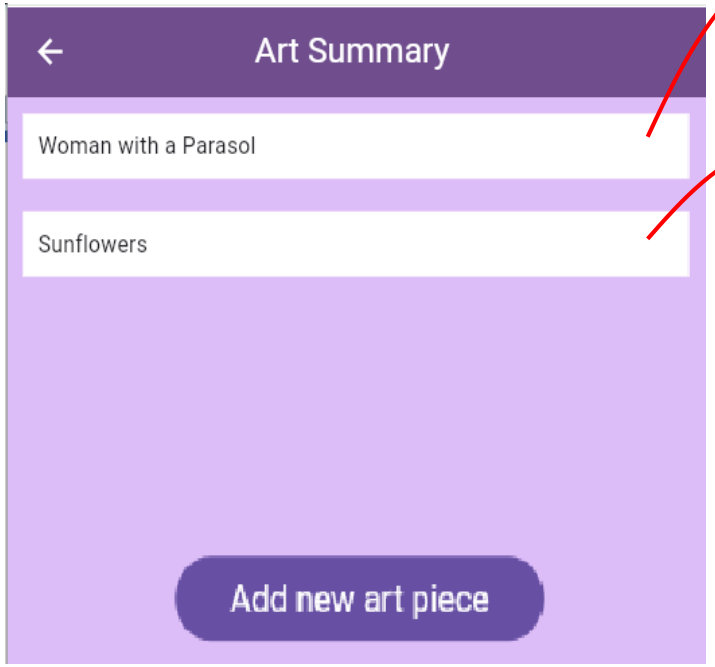
The ListView after 2 Art added.

Art App (Show Art Details) (GestureDetector Widget)

Project: art_details

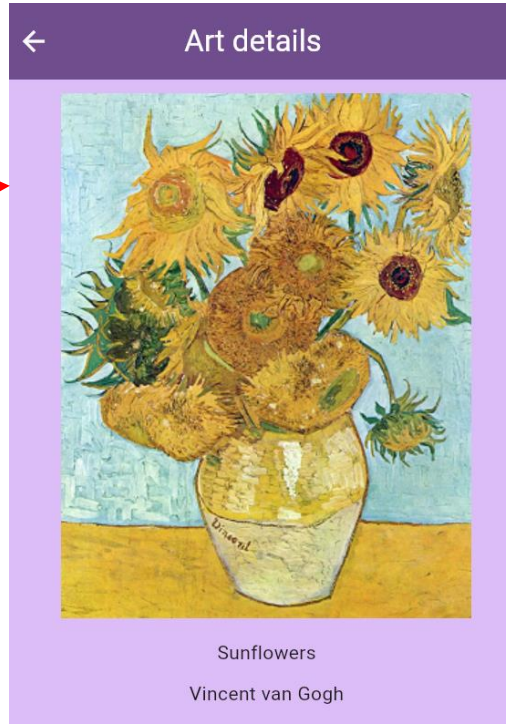
Enhance the app – show Art Details

Tapping on an art title in the list will bring up the details of the respective art piece.



tap

tap

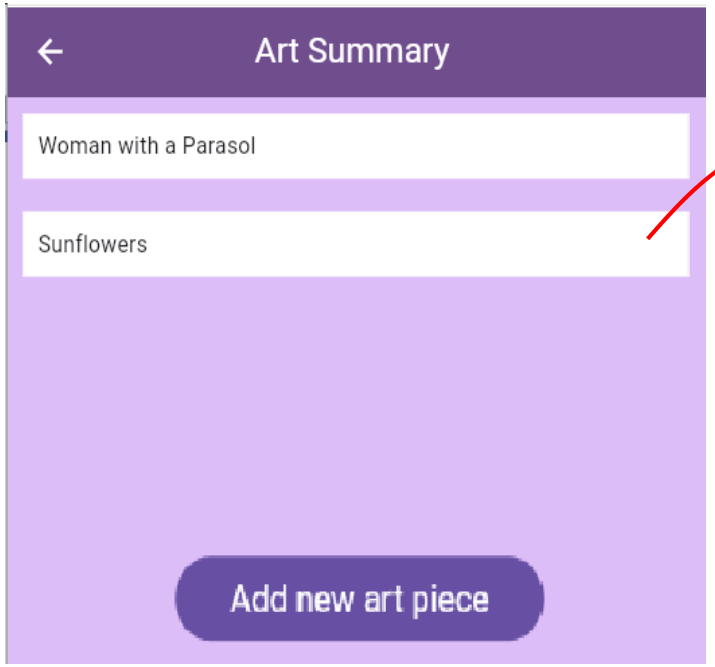


These are in fact the same page except that the image, title and artist are dynamic variables.

artdetails.dart (new page)

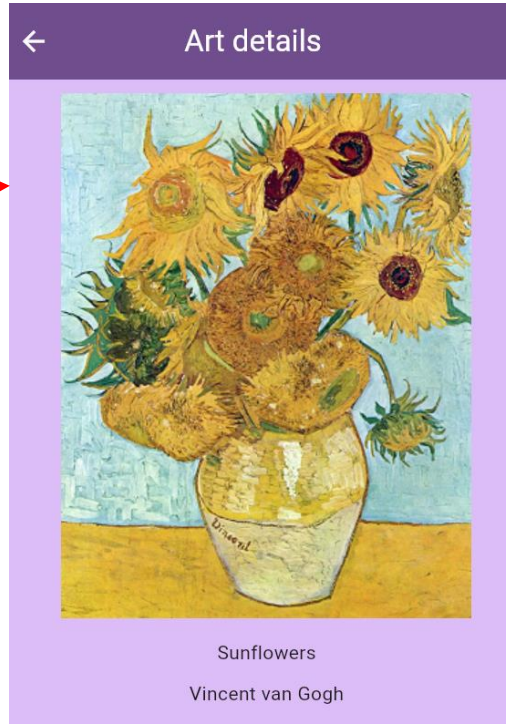
The data

The stored title, artist and image of the tapped item will be retrieved to display the image and information accordingly



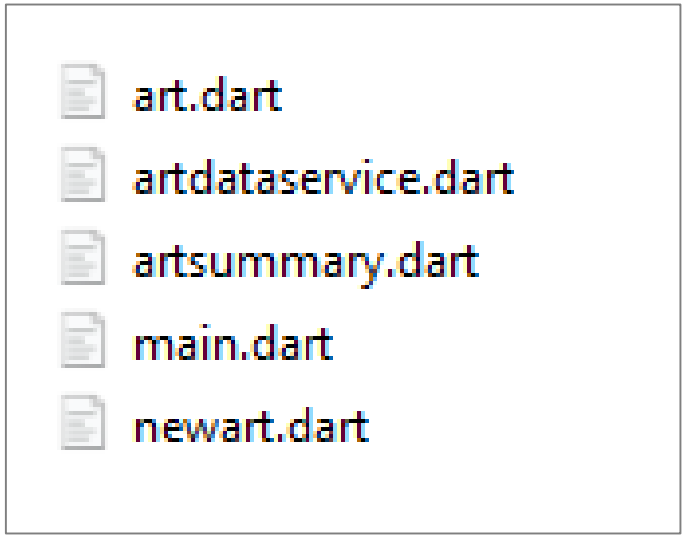
tap

The title, artist and image of the tapped item will be stored away



In Visual Studio Code

- Create a new Flutter project **art_details**
- copy all 5 files from previous project **art_add_new**
- To add 1 new file **artdetails.dart** (with Scaffold)

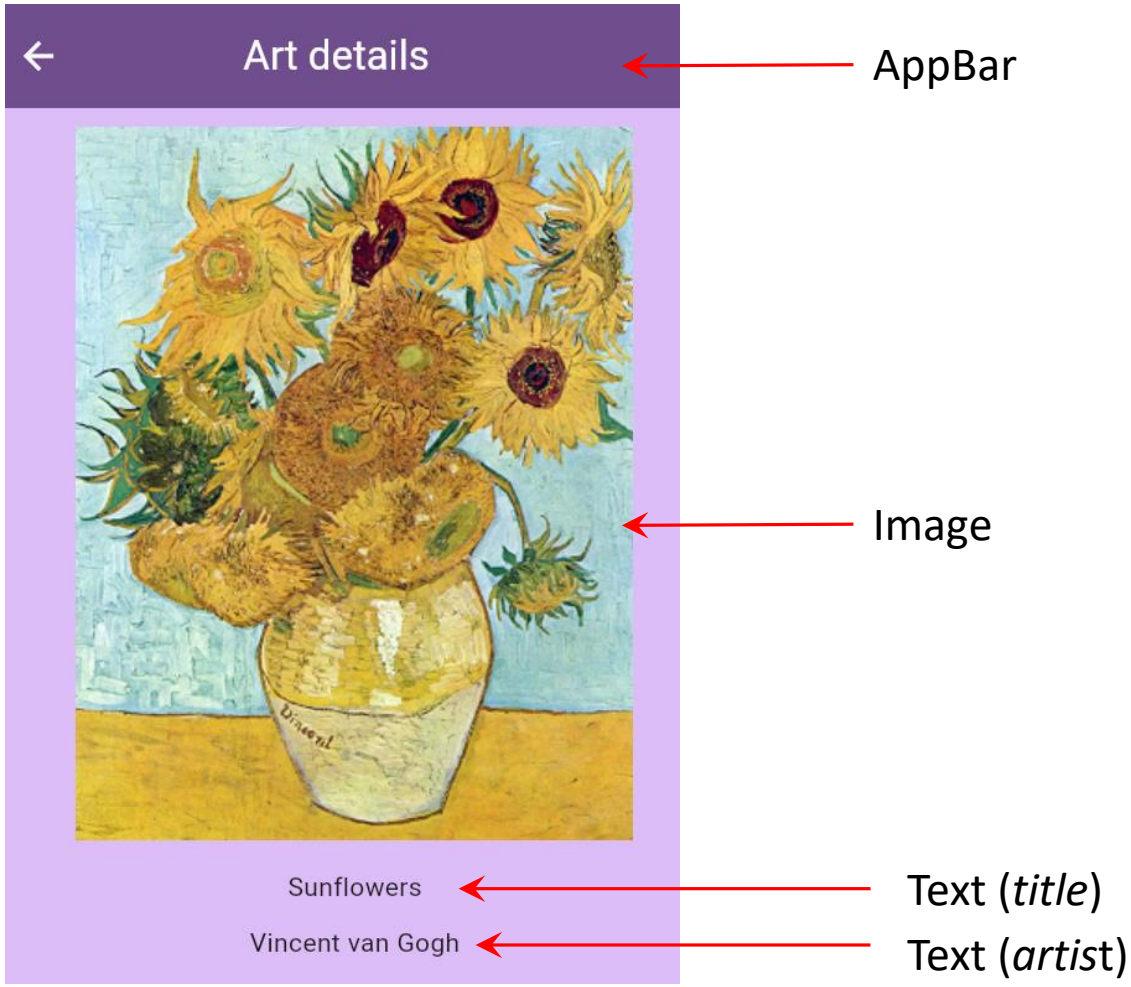


copy all 5 files from previous project **art_add_new**

(due to change in folder names, delete old-imports and re-import with the guide from Quick Fix)

In Visual Studio Code

new file **artdetails.dart**



GestureDetector Widget

To detect and respond to various gestures, such as:

(this will be used in ***artsummary.dart*** to detect the tapping of the item in the ListView)

- **onTap**: Called when the user taps the widget.
- **onDoubleTap**: Triggered when the user double-taps the widget.
- **onLongPress**: Fires when the user presses and holds the widget.
- **onPanUpdate**: Detects dragging gestures and provides updates on movement.
- **onHorizontalDragUpdate / onVerticalDragUpdate**: Detects horizontal or vertical drag gestures.
- **onScaleUpdate**: Recognizes pinch-to-zoom gestures

etc

Enhance the app – show Art Details

For images:

- Prepare the necessary image files in assets/img
- Update pubspec.yaml

For ***new*** artdetails.dart file (ArtDetails()):

- Prepare the Scaffold: Column with Image and Text to display art details (pull dynamic data from ArtDataService)

For main.dart file:

- Add in a new named route to facilitate navigating to the new ArtDetails() page

For artsummary.dart file:

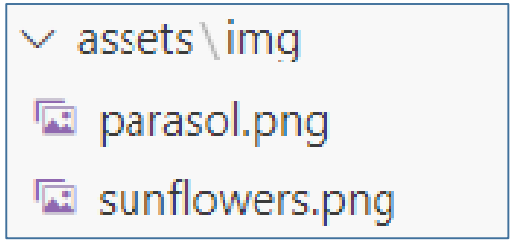
- Detect the tapping on art piece in the ListView (requires **GestureDetector Widget**)
- Registered the title, artist and image of the art piece being tapped

For artdataservice.dart file:

- Create new variables to register the title, artist and image of the art piece being tapped

For images:

- Prepare the necessary image files in assets/img
- Update pubspec.yaml



pubspec.yaml

```
59  # To add assets to your application, add an assets section, like this:
60  assets:
61    - assets/img/parasol.png
62    - assets/img/sunflowers.png
63  #   - images/a_dot_ham.jpeg
```

parasol.png

sunflowers.png

** alternatively, use web pictures*

For artdataservice.dart file:

- Create 3 new variables to register the *title*, *artist* and *image* of the art piece being tapped

Edit and add in these 3 variables.

Will be updated when item is being tapped in the list in ArtSummary()

Later, to be used in the new ArtDetails() page

artdataservice.dart

```
class ArtDataService {
  static String tappedTitle="";
  static String tappedArtist="";
  static String tappedImage="";

    static   List<Art> z = [];

    // original
    // code in artdataservice.dart
    :
    :
    :
}
```

* All the **static** keywords are crucial

For artsummary.dart file:

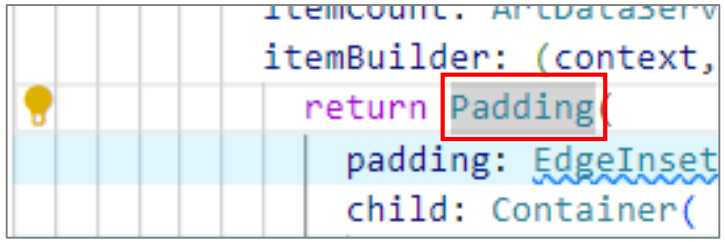
- Detect the tapping on art piece in the ListView (needs **GestureDetector Widget**)
- Registered the title, artist and image of the art piece being tapped

Wrap the current `Padding` with a **GestureDetector** Widget to detect the tapping (*steps in next slide*)

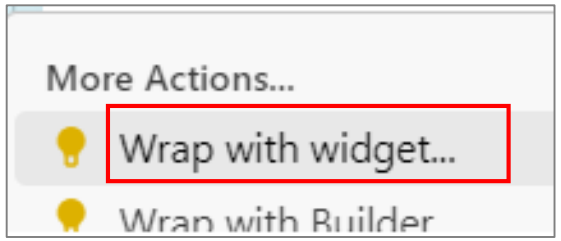
artsummary.dart

```
body: Column(children: [
  SizedBox(
    height: 200,
    child: ListView.builder(
      itemCount: ArtDataService.getCount(),
      itemBuilder: (context, index) {
        return Padding(
          padding: EdgeInsets.all(10),
          child: Container(
            padding: EdgeInsets.all(0),
            color: Colors.white,
            child: Text(ArtDataService.getArtAt(index).title),
          :
          :
```

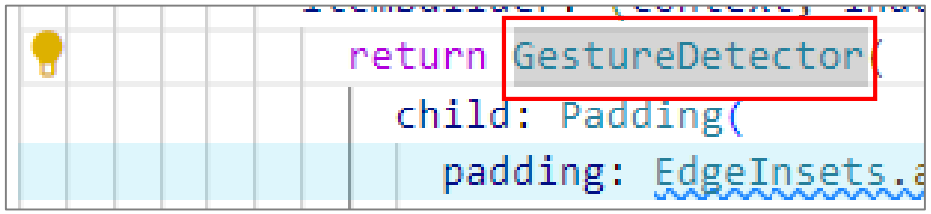
Wrap the current Padding with a Gesture Detector Widget



Click on Padding until appears



Click on and select Wrap with widget...



Replace default widget with **GestureDetector**

(The original Padding will automatically becomes the child of this new **GestureDetector**)

Add in **onTap()** inside **GestureDetector** - (the code will run when the item is tapped)

Add in **onTap()**

When an item in the ListView is tapped, update the 3 new variables in **ArtDataService** and Navigate to the new **ArtDetails** page

(*There will be temporary errors as the new page has not been set up yet. See following slides*)

artsummary.dart

```
body: Column(children: [
        SizedBox(
          height: 200,
          child: ListView.builder(
            itemCount: ArtDataService.getCount(),
            itemBuilder: (context, index) {
              return GestureDetector(
                onTap: () {
                  ArtDataService.tappedTitle = ArtDataService.getArtAt(index).title;
                  ArtDataService.tappedArtist = ArtDataService.getArtAt(index).artist;
                  ArtDataService.tappedImage = ArtDataService.getArtAt(index).image;
                  Navigator.pushNamed(context, "/artdetailspage");
                },
                child: Padding(
                  padding: EdgeInsets.all(10),
                  child: Container(
                    padding: EdgeInsets.all(10),
                    color: Colors.white,
                    child: Text(ArtDataService.getArtAt(index).title),
                  :
                  :
```

For main.dart file:

- Add in named route to facilitate navigating to the new ArtDetails() page

Add this named route.

(There will be temporary errors as the new page has not been set up yet. See following slides)

main.dart

```
void main() {
  runApp(MaterialApp(home: ArtSummary(), routes: {
    "/newartpage": (context) => NewArt(),
    "/artsummarypage": (context) => ArtSummary(),
    "/artdetailspage": (context) => ArtDetails(),
  }));
}
```

artdetails.dart (New)(Full code)

- Create a new file **artdetails.dart**
- In there, create a class **ArtDetails**

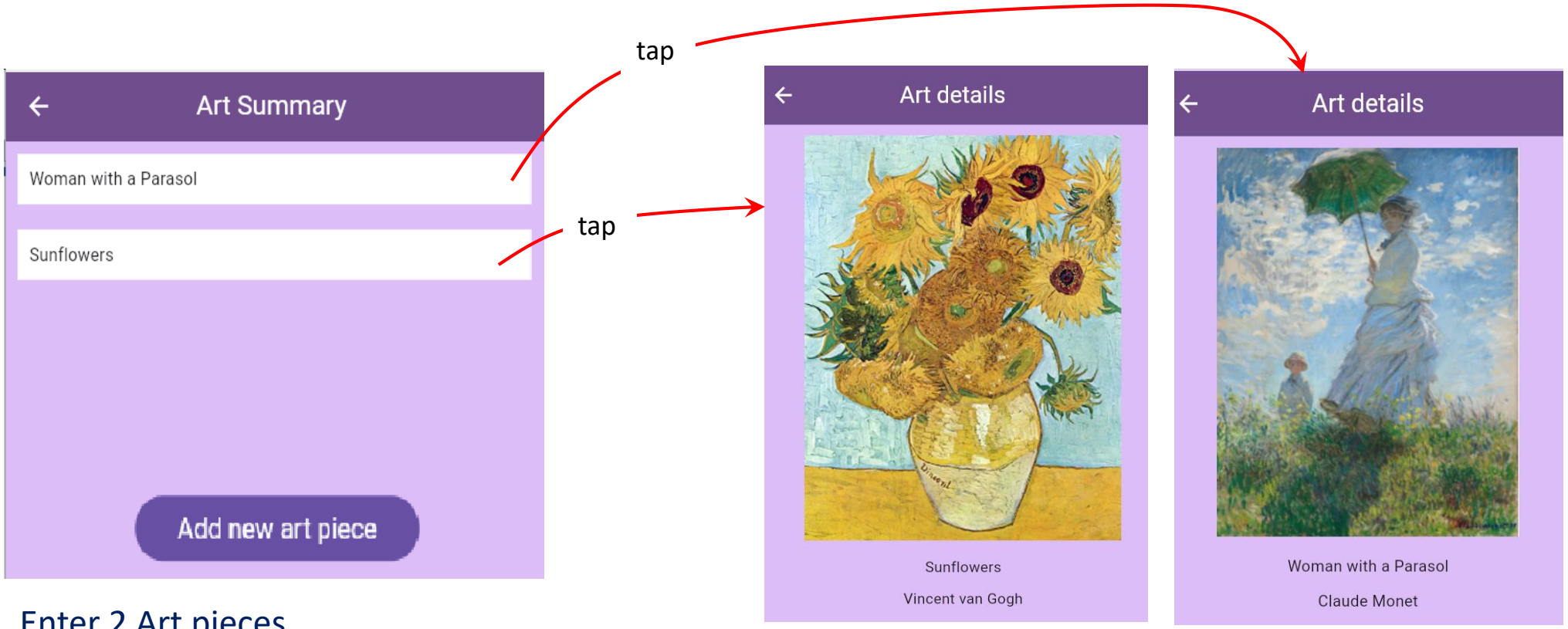
```dart
import 'package:flutter/material.dart';
class ArtDetails extends StatefulWidget {
  const ArtDetails({super.key});

  @override
  State<ArtDetails> createState() => _ArtDetailsState();
}

class _ArtDetailsState extends State<ArtDetails> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color.fromARGB(255, 221, 189, 247),
      appBar: AppBar(
        title: const Text("Art details"),
        backgroundColor: Color.fromARGB(255, 112, 78, 142),
        foregroundColor: Colors.white,
        centerTitle: true,
      ),
```

artdetails.dart

```dart
body: Center(
  child: Column(children: [
    SizedBox(height: 10),
    Image.asset("assets/img/${ArtDataService.tappedImage}"),
    const SizedBox(height: 15),
    Text(ArtDataService.tappedTitle),
    const SizedBox(height: 10),
    Text(ArtDataService.tappedArtist),
    SizedBox(height: 10),
            ]),
),   );  } }
```

pull dynamic data from **ArtDataService**

Run the app



Enter 2 Art pieces.
Tapping on an art title in the list will bring up the details of the respective art piece.

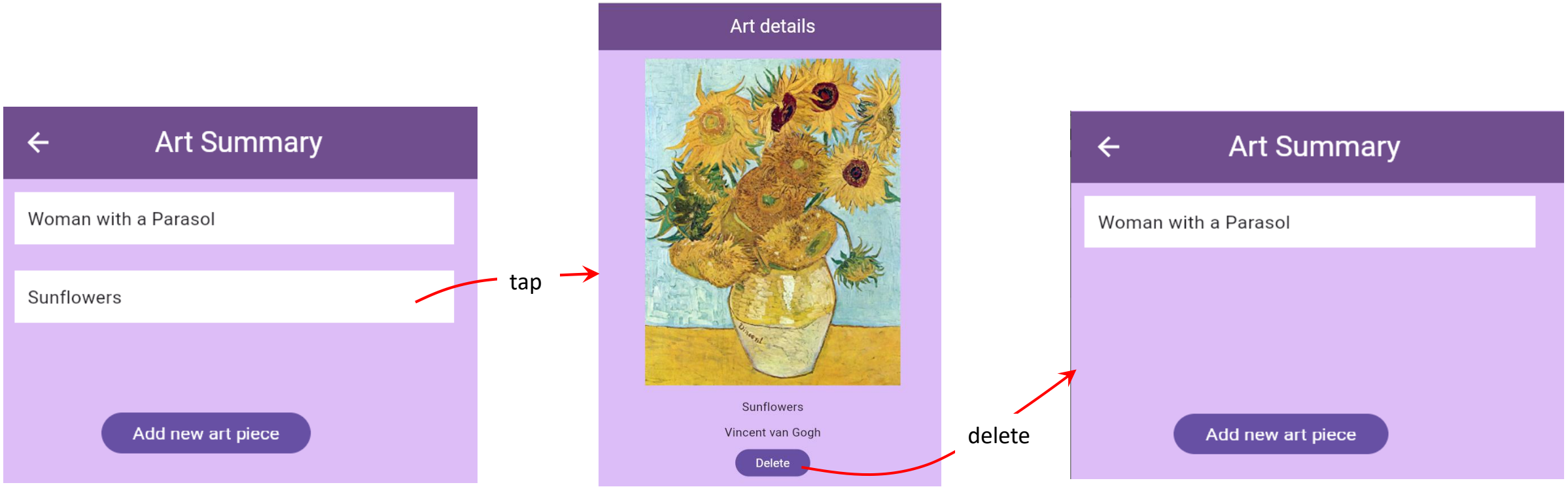
artdetails.dart

Art App with Delete Button in Art Details

Project: art_delete

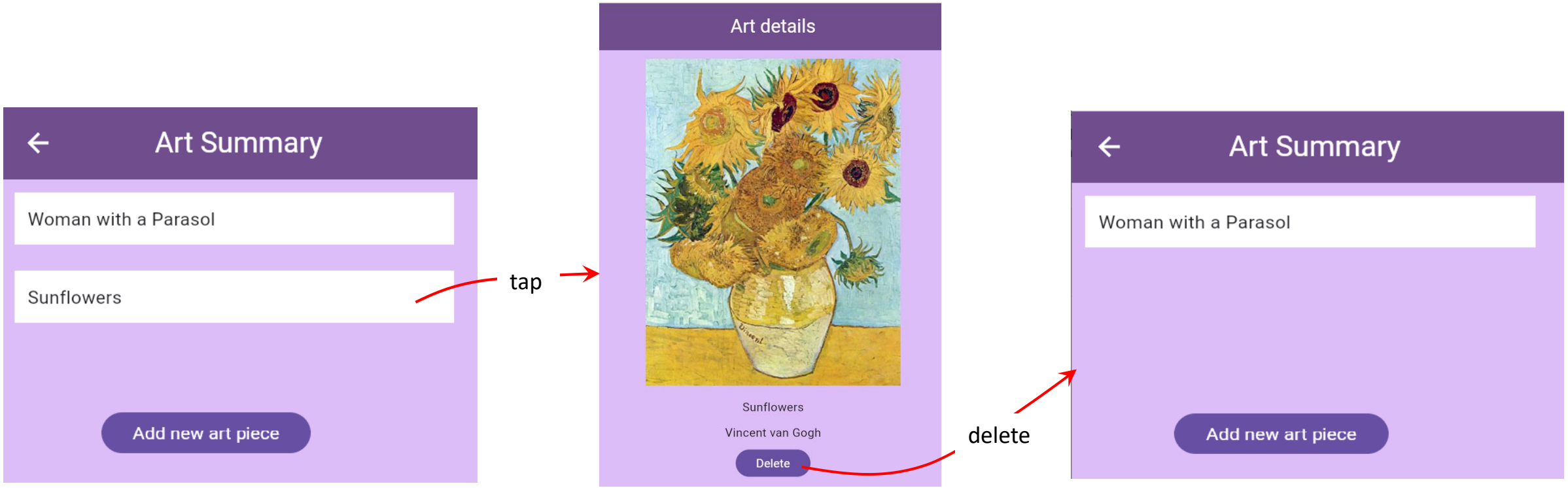
Enhance the app – Delete Art

Add a **Delete button** in ArtDetails.
Click to the delete the art and return to ArtSummary



Sunflowers deleted from the ListView

The data



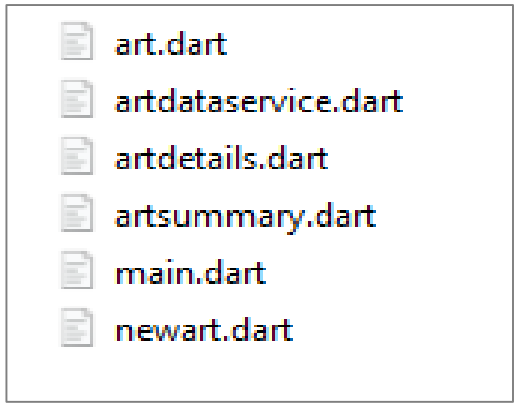
Store away the artId of the tapped item also

Retrieve the stored artId to delete the correct Art object from List<Art>

In Visual Studio Code

Create a new Flutter project **art_delete**

Copy all 6 files from previous project **art_details**



(due to change in folder names, delete old-imports and re-import with the guide from Quick Fix)

Next, copy **assets** folder from previous project **art_details**

Next, edit **pubspec.yaml** to include the necessary image files

```
60  assets:
61    - assets/img/parasol.png
62    - assets/img/sunflowers.png
```

Enhance the app – Delete Art

For artdataservice.dart file:

- Create a new variable *tappedId* to register the Art Id for the Art object being tapped (this is needed in order to delete the correct Art from List<Art>)

For artsummary.dart file:

- Expand the current tap() to register the *tappedId* of the Art being tapped

For artdeatils.dart file (ArtDetails()):

- Add in a **new Delete Button**
- In onPressed(), call **ArtDataService**.removeArtByArtID() to delete the Art object based on *tappedId*

For artdataservice.dart file:

- Create a new variable to register the tapped Id for the Art object to be deleted

Edit and add in 1 more variable, *tappedId*

This is the main variable that facilitate the deletion of the correct Art object

artdataservice.dart

```
class ArtDataService {
  static String tappedTitle="";
  static String tappedArtist="";
  static String tappedImage="";
  static String tappedId="";

   static   List<Art> z = [];

   // original
   // code in artdataservice.dart
   :
   :
   :
}
```

* All the **static** keywords are crucial

In artsummary.dart, expand the current tap() to register the tapped Id of the Art being tapped.

Add this statement to capture the Art Id being tapped

artsummary.dart

```
body: Column(children: [
        SizedBox(
          height: 200,
          child: ListView.builder(
            itemCount: ArtDataService.getCount(),
            itemBuilder: (context, index) {
              return GestureDetector(
                onTap: () {
                  ArtDataService.tappedTitle = ArtDataService.getArtAt(index).title;
                  ArtDataService.tappedArtist = ArtDataService.getArtAt(index).artist;
                  ArtDataService.tappedImage = ArtDataService.getArtAt(index).image;
                  ArtDataService.tappedId =  ArtDataService.getArtAt(index).artId;

                  Navigator.pushNamed(context, "/artdetailspage");
                },
                child: Padding(
                  padding: EdgeInsets.all(10),
                  child: Container(
                    padding: EdgeInsets.all(10),
                    color: Colors.white,
                    child: Text(ArtDataService.getArtAt(index).title),
                  :
```

Add new Delete button - artdetails.dart (Full code)

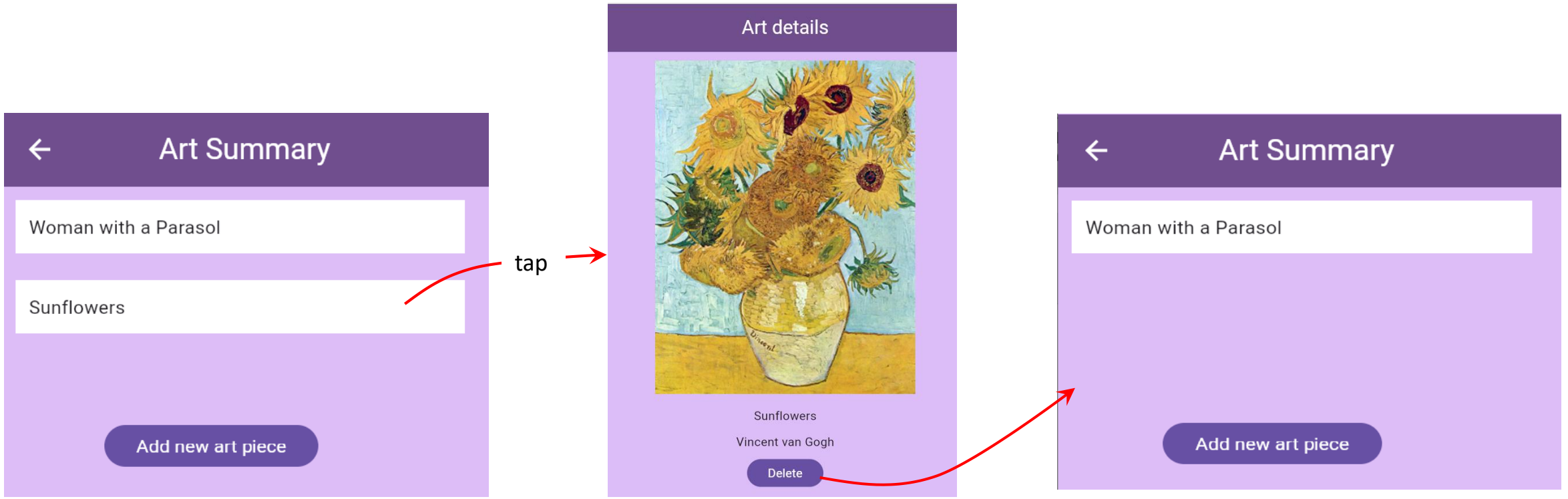
```dart
class ArtDetails extends StatefulWidget {
  const ArtDetails({super.key});

  @override
  State<ArtDetails> createState() => _ArtDetailsState();
}

class _ArtDetailsState extends State<ArtDetails> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Color.fromARGB(255, 221, 189, 247),
      appBar: AppBar(
        title: const Text("Art details"),
        backgroundColor: Color.fromARGB(255, 112, 78, 142),
        foregroundColor: Colors.white,
        centerTitle: true,
      ),
```

artdetails.dart

```dart
body: Center(
  child: Column(children: [
    SizedBox(height: 10),
    Image.asset("assets/img/${ArtDataService.tappedImage}"),
    const SizedBox(height: 15),
    Text(ArtDataService.tappedTitle),
    const SizedBox(height: 10),
    Text(ArtDataService.tappedArtist),
    SizedBox(height: 10),

    FilledButton(
      child: Text("Delete"),
      onPressed: () {
       ArtDataService.removeArtByArtId(ArtDataService.tappedId);
       Navigator.pushNamed(context, "/artsummarypage");
      })

    ]),
),   );  } }
```

pull **tappedId** from **ArtDataService** and call **removeArtByArtId()** to delete

Run the App



Add a **Delete button** in ArtDetails. Click to the delete the art and return to artsummary.dart

Sunflowers deleted from the summary

Dialog

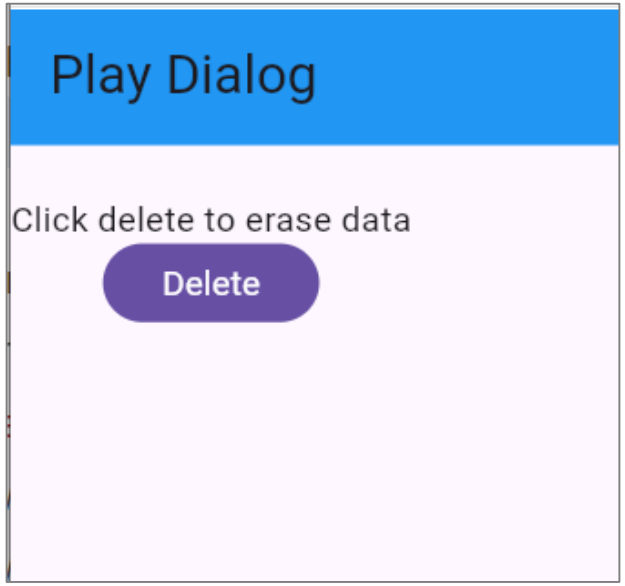
Exercise 5.1

Exercise 5.1

- Currently, the Art object will be deleted the moment the Delete button is clicked, without further confirmation with the user.
- Study the next few slides on pop-up Dialog Widget.
- Create a new project: **art_dialog**
- Copy the files from previous project , **art_delete** and enhance it to include a confirmation **Dialog pop-up** when user clicks on the **Delete** button.

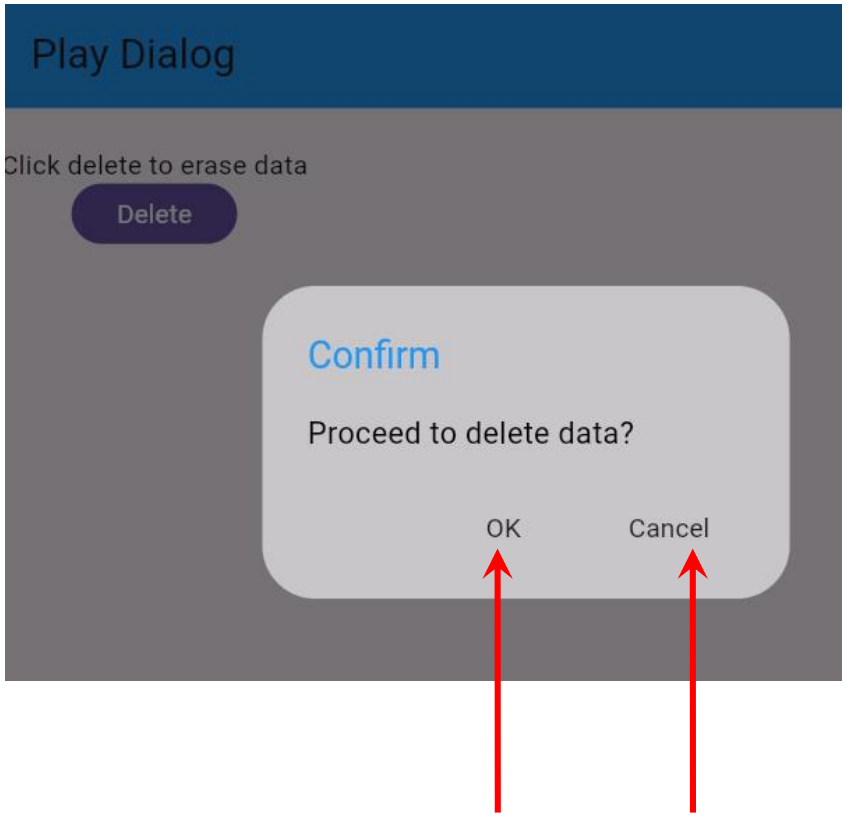
- If the user clicked Cancel in Dialog, go back to Art details.
- If the user clicked OK in Dialog, delete the Art object and go back to Art Summary (home page).

Pop up dialog for confirmation



A simple experimental app.

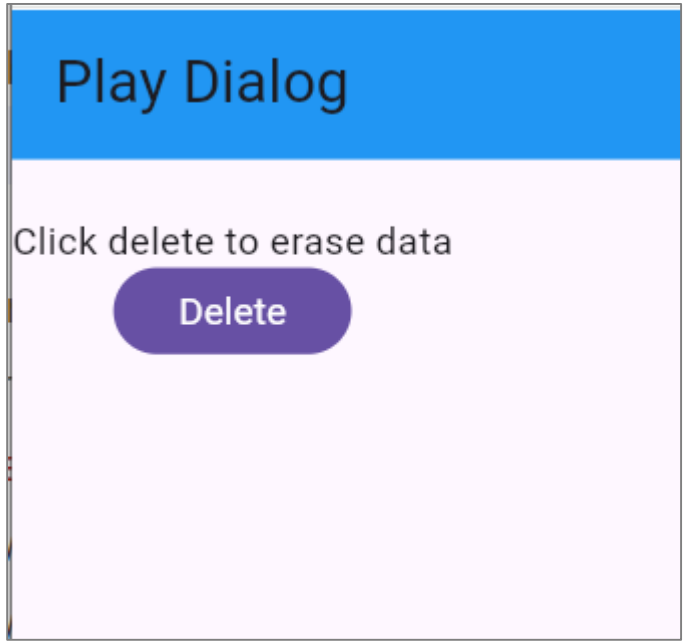
Clicking Delete button



Need to capture the response from the user

Pop up dialog for confirmation

Create a new project. Create a new stateless class. Feed in the Scaffold as:




```
 return Scaffold(
        appBar: AppBar(
          title: const Text("Play Dialog"),
          backgroundColor: Colors.blue,
        ),
        body: Column(children: [
          const SizedBox(height: 20),
          const Text("Click delete to erase data"),
          FilledButton(
            onPressed: () { },
           child: const Text("Delete"),
          )
         ])
);
```

Dialog Widget

To create pop-up dialogs that appear over the main content.

It uses a function call to **showDialog()** that displays a dialog (such as **AlertDialog**) and returns its result.

AlertDialog: Displays a message with optional actions like "OK" or "Cancel".

```
showDialog(
    context: context,
    builder: (context) {
        return AlertDialog(
            backgroundColor:_____________,
            title:_____________________,
            content:_____________________,
            actions: [_____________________],

        );
    }
);
```

Next edit the onPressed() { } for the button

to pop up the dialog (**AlertDialog**), with the message of 'Proceed to delete data?'

and

also capture the user's response via the OK and Cancel buttons.

```
onPressed: () {
    showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
              backgroundColor: Colors.white60,
              title: const Text(
                'Confirm',
                style: TextStyle(color: Colors.blue, fontSize: 20),
              ),
              content: const Text(
                'Proceed to delete data?',
                style: const TextStyle(
                    color: Colors.black, fontSize: 16),
              ),
              actions: [
                MaterialButton(
                    child: const Text('OK'), onPressed: () {
                       print ("OK clicked.");
                       print ("Delete data...");
                       Navigator.of(context).pop();
                    }),
                MaterialButton(
                    child: const Text('Cancel'), onPressed: () {
                      print ("Cancel clicked.");
                       Navigator.of(context).pop();
                    }),
              ]);
        });
},
```

Before Topic 6

Before the Topic 6

- whenever the Art app is re-started, all the art data entered previously, is gone. To have the entered data shown, upon starting the app, requires the use of database (for persistent data storage, even after the app is closed)
- Topic 6 will focus on the use of Google's Firebase to store the Art info entered by the user
- Topic 6 builds on Ex 5-1. Most of the files in the Art App project remain unchanged.
- In Topic 6, the artdataservice.dart (class ArtDataService), besides updating local storage List<Art>, some CRUD functions will be expanded to update the Google's Firebase as well. For instance:

```
static void addArt (String artId, String artTitle, String artArtist, String artImage){

   // update local storage
   z.add( Art( artId, artTitle, artArtist, artImage));

   (code to update google's Firebase)

}
```