



# Retrieve a list of contacts

<https://developer.android.com/training/contacts-provider/retrieve-names>

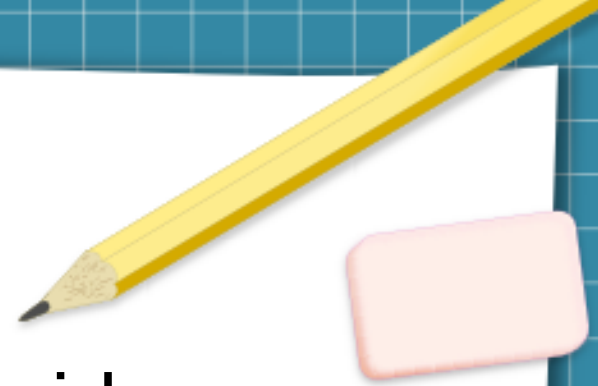
# Cont ...



- This lesson shows you how to retrieve a list of contacts whose data matches all or part of a search string, using the following techniques:
  - Match contact names
  - Match a specific type of data, such as a phone number
  - Match any type of data

-

# Cont ...



- Request permission to read the provider
  - `<uses-permission  
android:name="android.permission.READ_CONTA  
CTS" />`

# Match a contact by name and list the results



- This technique tries to match a search string to the name of a contact or contacts in the Contact Provider's ContactsContract.Contacts table.
- Define ListView/RecyclerView and item layouts
  - `<ListView`  
    `xmlns:android="http://schemas.android.com/apk/res/android"`  
        `android:id="@android:id/list"`  
        `android:layout_width="match_parent"`  
        `android:layout_height="match_parent"/>`

# Cont ...



- To help you write queries against the Contacts Provider, the Android framework provides a contracts class called `ContactsContract`, which defines useful constants and methods for accessing the provider.
- When you use this class, you don't have to define your own constants for content URIs, table names, or columns.
- To use this class, include the following statement:
  - `import android.provider.ContactsContract;`

# Cont ...



- Since the code uses a CursorLoader to retrieve data from the provider, you must specify that it implements the loader interface LoaderManager.LoaderCallbacks.
  - import  
android.support.v4.app.LoaderManager.LoaderCallbacks;
  - ...
  - public class ContactsFragment extends Fragment  
implements
  - LoaderManager.LoaderCallbacks<Cursor> {

# Cont ...

- Define global variables that are used in other parts of the code:

```
/*
```

```
* Defines an array that contains column names to move from
```

```
* the Cursor to the ListView.
```

```
*/
```

```
@SuppressWarnings("InlinedApi")
```

```
private final static String[] FROM_COLUMNS = {
```

```
    Build.VERSION.SDK_INT
```

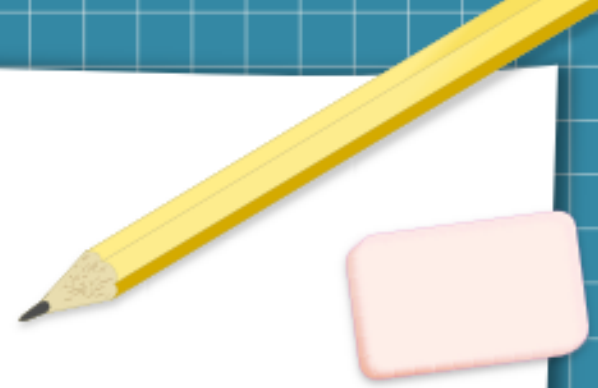
```
    >= Build.VERSION_CODES.HONEYCOMB ?
```

```
    ContactsContract.Contacts.DISPLAY_NAME_PRIMARY :
```

```
    ContactsContract.Contacts.DISPLAY_NAME
```

```
};
```

## Cont ...



// The contact's LOOKUP\_KEY

String mContactKey;

// A content URI for the selected contact

Uri mContactUri;



# Define a projection



- Define a constant that contains the columns you want to return from your query.
- In Android 3.0 (API version 11) and later, the name of this column is `Contacts.DISPLAY_NAME_PRIMARY`; in versions previous to that, its name is `Contacts.DISPLAY_NAME`.
- `Contacts._ID` and `LOOKUP_KEY` are used together to construct a content URI for the contact the user selects.

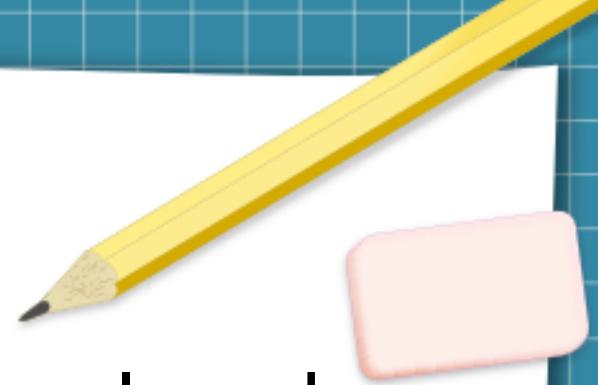
# Specify the selection criteria



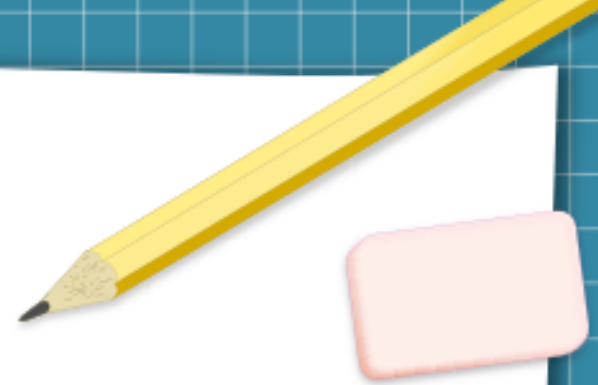
- To specify the data you want, create a combination of text expressions and variables that tell the provider the data columns to search and the values to find.
- For the text expression, define a constant that lists the search columns.
- Although this expression can contain values as well, the preferred practice is to represent the values with a "?" placeholder.

## Cont ...

- During retrieval, the placeholder is replaced with values from an array.
- Using "?" as a placeholder ensures that the search specification is generated by binding rather than by SQL compilation.
- This practice eliminates the possibility of malicious SQL injection.



# Cont ...



```
private static final String SELECTION =  
    Build.VERSION.SDK_INT >=  
    Build.VERSION_CODES.HONEYCOMB ?  
    Contacts.DISPLAY_NAME_PRIMARY + "  
LIKE ?" :  
    Contacts.DISPLAY_NAME + " LIKE ?";  
// Defines a variable for the search string  
private String mSearchString;  
// Defines the array to hold values that replace the ?  
private String[] mSelectionArgs = { mSearchString };
```

# Initialize the loader

- Since you're using a CursorLoader to retrieve data, you must initialize the background thread and other variables that control asynchronous retrieval.
- Do the initialization in `onActivity()`,



# Implement onCreateLoader()



```
public Loader<Cursor> onCreateLoader(int loaderId, Bundle args) {  
    // Makes search string into pattern and stores it in the selection array  
    mSelectionArgs[0] = "%" + mSearchString + "%";  
    // Starts the query  
    return new CursorLoader(  
        getActivity(),  
        ContactsContract.Contacts.CONTENT_URI,  
        PROJECTION,  
        SELECTION,  
        mSelectionArgs,  
        null  
    );  
}
```

# Implement onLoadFinished() and onLoaderReset()



- Implement the onLoadFinished() method.
- The loader framework calls onLoadFinished() when the Contacts Provider returns the results of the query.
- In this method, put the result Cursor in the SimpleCursorAdapter

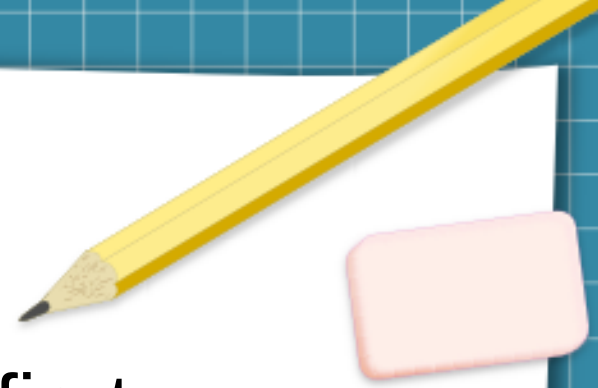
# Match a contact by a specific type of data



- This technique allows you to specify the type of data you want to match.
- Retrieving by name is a specific example of this type of query, but you can also do it for any of the types of detail data associated with a contact.
- For example, you can retrieve contacts that have a specific postal code; in this case, the search string has to match data stored in a postal code row.



# Cont ...



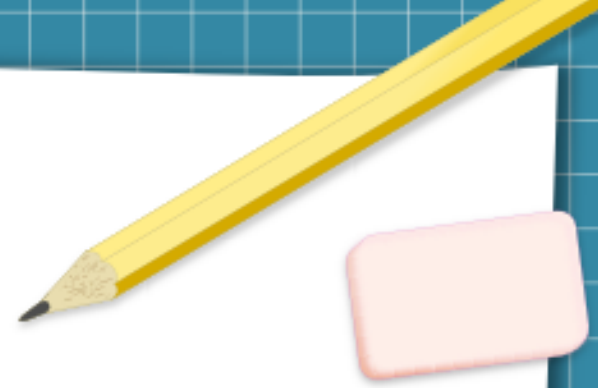
- To implement this type of retrieval, first implement the following code, as listed in previous sections:
  - Request Permission to Read the Provider.
  - Define ListView and item layouts.
  - Define a Fragment that displays the list of contacts.
  - Define global variables.
  - Initialize the Fragment.

# Cont ...



- Set up the CursorAdapter for the ListView.
- Set the selected contact listener.
- Define constants for the Cursor column indexes.
- Although you're retrieving data from a different table, the order of the columns in the projection is the same, so you can use the same indexes for the Cursor.
- Define the onItemClick() method.
- Initialize the loader.
- Implement onLoadFinished() and onLoaderReset().

# Cont ...



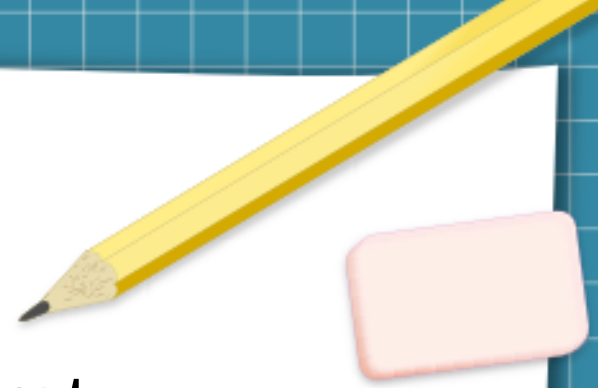
- In addition:
  - Choose the data type and table
  - Define a projection
  - Define search criteria
  - Implement onCreateLoader()

# Match a contact by any type of data



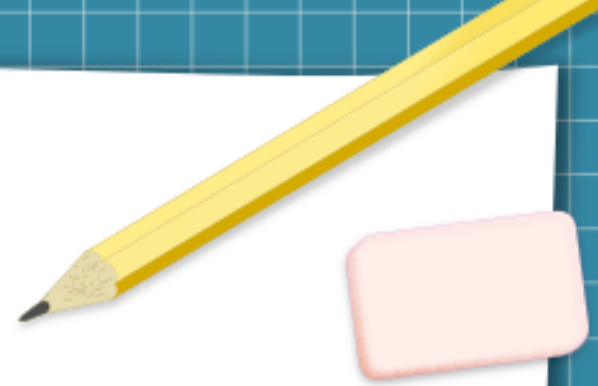
- Retrieving a contact based on any type of data returns contacts if any of their data matches a the search string, including name, email address, postal address, phone number, and so forth.
- This results in a broad set of search results.
- For example, if the search string is "Doe", then searching for any data type returns the contact "John Doe"; it also returns contacts who live on "Doe Street".

# Cont ...



- To implement this type of retrieval, first implement the following code, as listed in previous sections:
  - Request Permission to Read the Provider.
  - Define ListView and item layouts.
  - Define a Fragment that displays the list of contacts.
  - Define global variables.
  - Initialize the Fragment.
  - Set up the CursorAdapter for the ListView.
  - Set the selected contact listener.

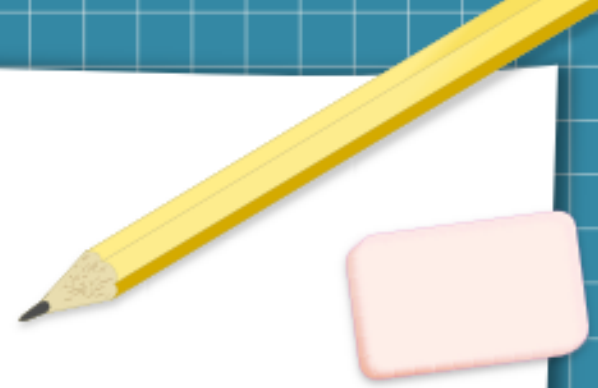
# Cont ...

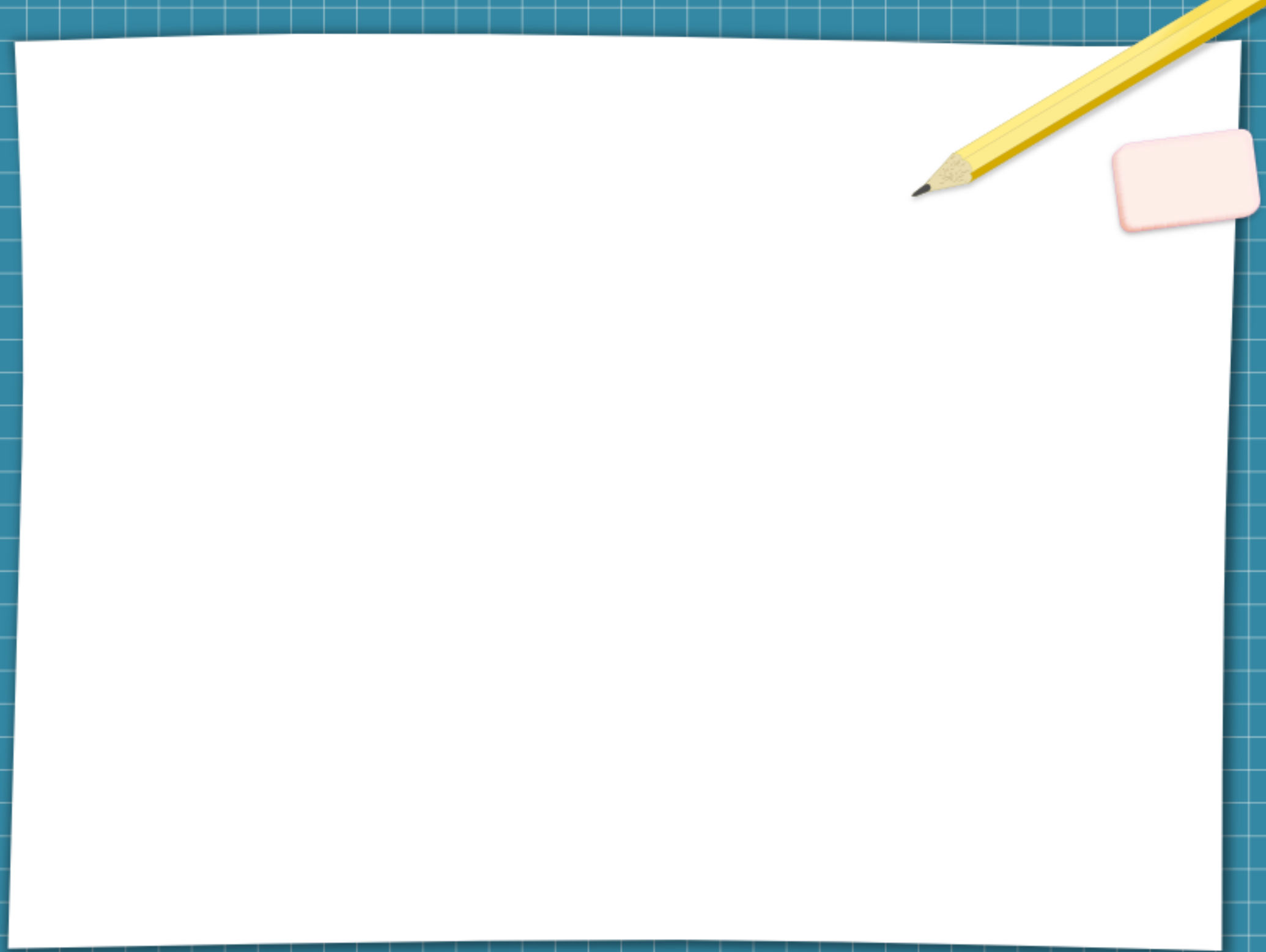


- Define a projection.
- Define constants for the Cursor column indexes.
- For this type of retrieval, you're using the same table you used in the section Match a contact by name and list the results. Use the same column indexes as well.
- Define the onItemClick() method.
- Initialize the loader.
- Implement onLoadFinished() and onLoaderReset().

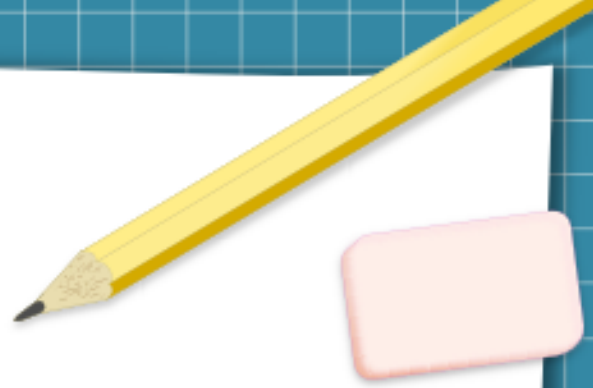
# Cont ...

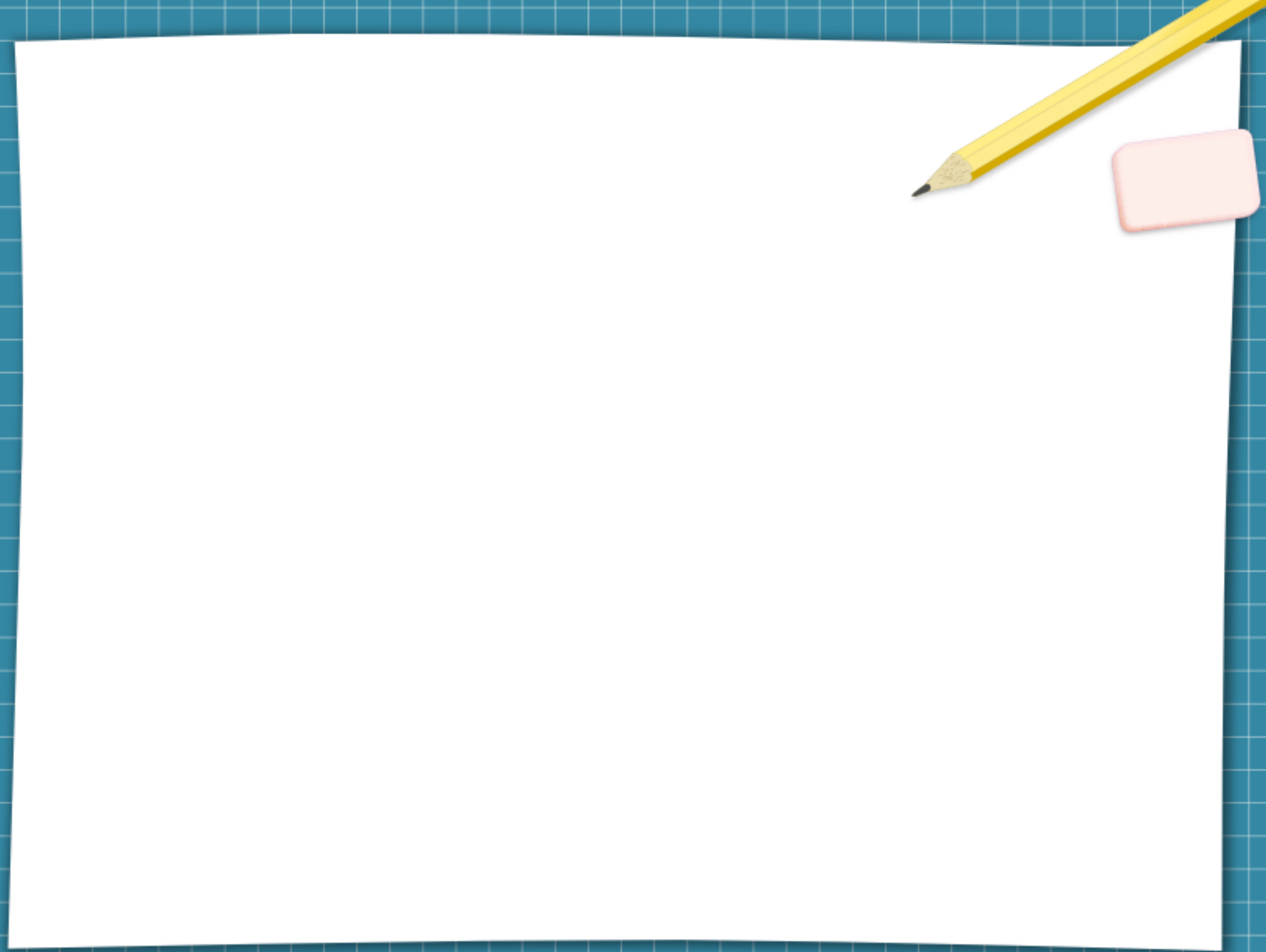
- In addition:
  - Remove selection criteria
  - Implement onCreateLoader()

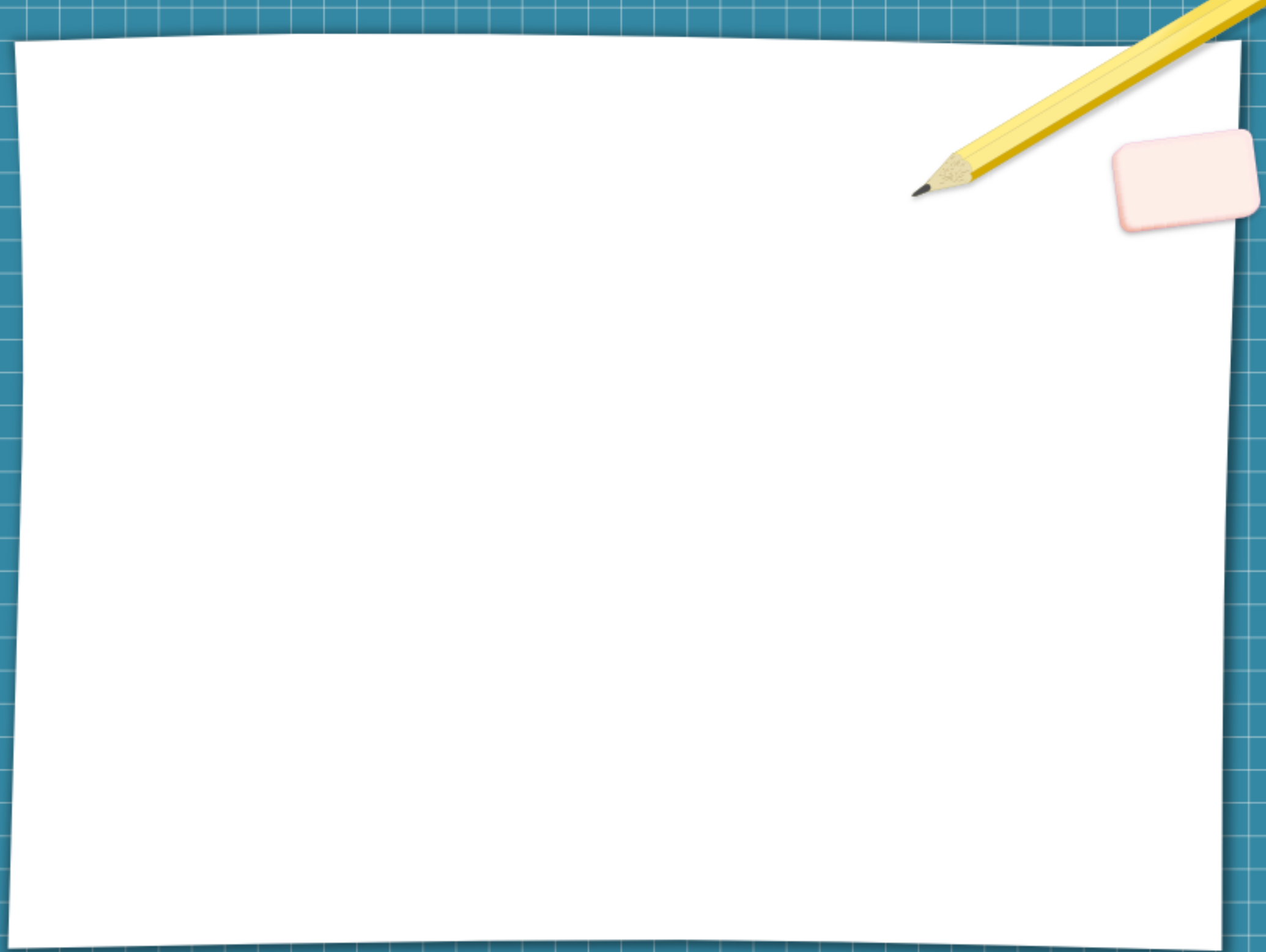


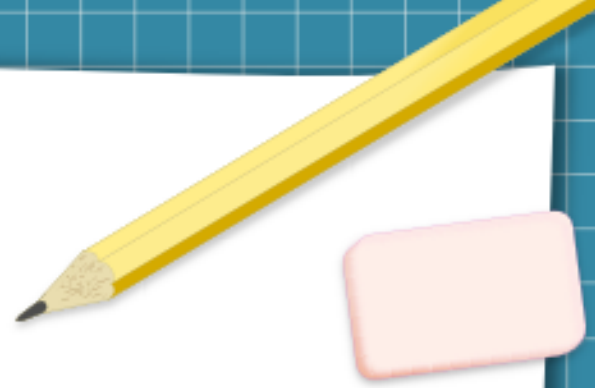


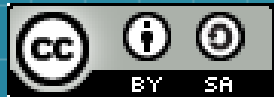
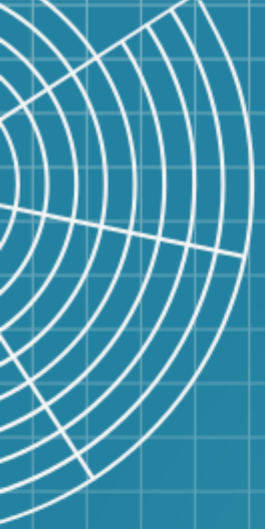




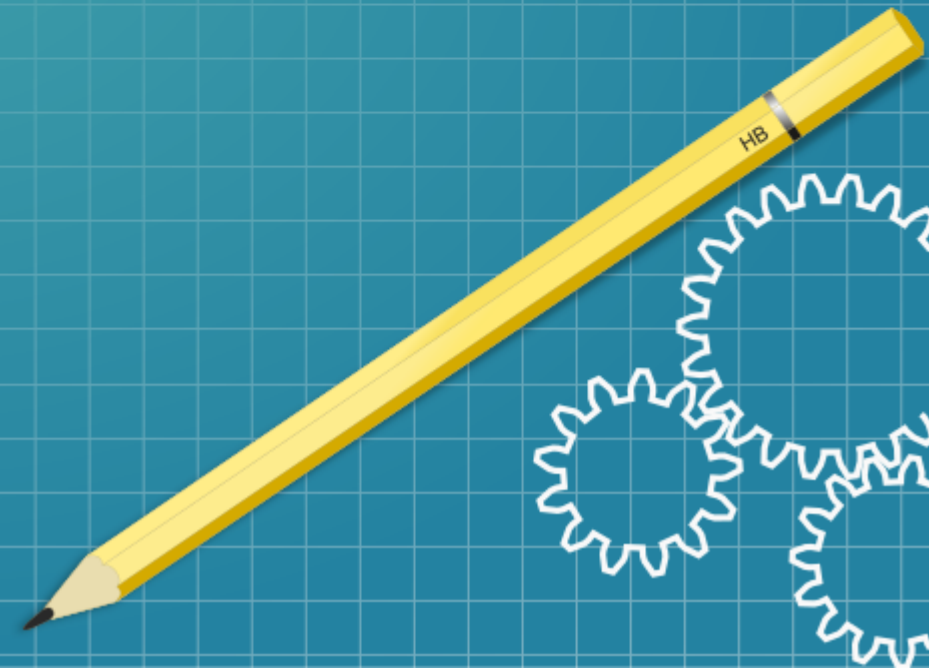
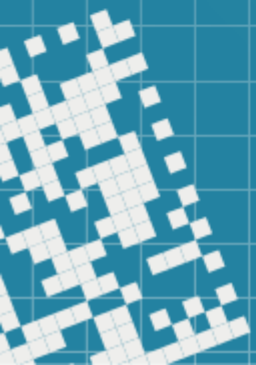








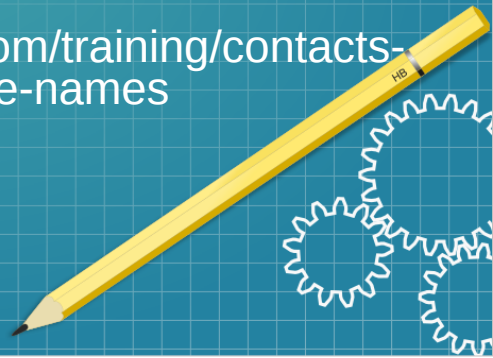
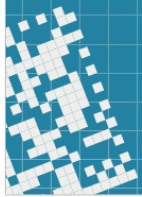
This work is licensed under a Creative Commons  
Attribution-ShareAlike 3.0 Unported License.  
It makes use of the works of Mateus Machado Luna.





# Retrieve a list of contacts

<https://developer.android.com/training/contacts-provider/retrieve-names>



## Cont ...



- This lesson shows you how to retrieve a list of contacts whose data matches all or part of a search string, using the following techniques:
  - Match contact names
  - Match a specific type of data, such as a phone number
  - Match any type of data
-

## Cont ...



- Request permission to read the provider
  - `<uses-permission  
android:name="android.permission.READ_CONTACTS" />`



## Match a contact by name and list the results



- This technique tries to match a search string to the name of a contact or contacts in the Contact Provider's ContactsContract.Contacts table.
- Define ListView/RecyclerView and item layouts
  - `<ListView`  
    `xmlns:android="http://schemas.android.com/apk/res/android"`  
        `android:id="@android:id/list"`  
        `android:layout_width="match_parent"`  
        `android:layout_height="match_parent"/>`

## Cont ...



- To help you write queries against the Contacts Provider, the Android framework provides a contracts class called `ContactsContract`, which defines useful constants and methods for accessing the provider.
- When you use this class, you don't have to define your own constants for content URIs, table names, or columns.
- To use this class, include the following statement:
  - `import android.provider.ContactsContract;`

## Cont ...



- Since the code uses a CursorLoader to retrieve data from the provider, you must specify that it implements the loader interface LoaderManager.LoaderCallbacks.
  - import  
android.support.v4.app.LoaderManager.LoaderCallbacks;
  - ...
  - public class ContactsFragment extends Fragment  
implements
  - LoaderManager.LoaderCallbacks<Cursor> {

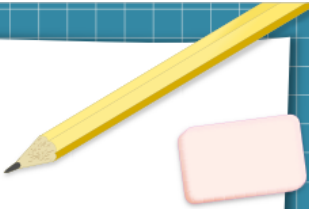
## Cont ...



- Define global variables that are used in other parts of the code:

```
/*
 * Defines an array that contains column names to move from
 * the Cursor to the ListView.
 */
@SuppressLint("InlinedApi")
private final static String[] FROM_COLUMNS = {
    Build.VERSION.SDK_INT
        >= Build.VERSION_CODES.HONEYCOMB ?
        ContactsContract.Contacts.DISPLAY_NAME_PRIMARY :
        ContactsContract.Contacts.DISPLAY_NAME
};
```

## Cont ...



```
// The contact's LOOKUP_KEY  
String mContactKey;  
// A content URI for the selected contact  
Uri mContactUri;
```

## Define a projection



- Define a constant that contains the columns you want to return from your query.
- In Android 3.0 (API version 11) and later, the name of this column is `Contacts.DISPLAY_NAME_PRIMARY`; in versions previous to that, its name is `Contacts.DISPLAY_NAME`.
- `Contacts._ID` and `LOOKUP_KEY` are used together to construct a content URI for the contact the user selects.

## Specify the selection criteria



- To specify the data you want, create a combination of text expressions and variables that tell the provider the data columns to search and the values to find.
- For the text expression, define a constant that lists the search columns.
- Although this expression can contain values as well, the preferred practice is to represent the values with a "?" placeholder.

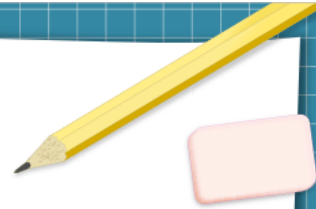
## Cont ...



- During retrieval, the placeholder is replaced with values from an array.
- Using "?" as a placeholder ensures that the search specification is generated by binding rather than by SQL compilation.
- This practice eliminates the possibility of malicious SQL injection.



## Cont ...



```
private static final String SELECTION =  
    Build.VERSION.SDK_INT >=  
    Build.VERSION_CODES.HONEYCOMB ?  
    Contacts.DISPLAY_NAME_PRIMARY + "  
    LIKE ?" :  
    Contacts.DISPLAY_NAME + " LIKE ?";  
    // Defines a variable for the search string  
    private String mSearchString;  
    // Defines the array to hold values that replace the ?  
    private String[] mSelectionArgs = { mSearchString };
```

## Initialize the loader



- Since you're using a CursorLoader to retrieve data, you must initialize the background thread and other variables that control asynchronous retrieval.
- Do the initialization in `onActivity()`,

# Implement onCreateLoader()



```
public Loader<Cursor> onCreateLoader(int loaderId, Bundle args) {  
    // Makes search string into pattern and stores it in the selection array  
    mSelectionArgs[0] = "%" + mSearchString + "%";  
    // Starts the query  
    return new CursorLoader(  
        getActivity(),  
        ContactsContract.Contacts.CONTENT_URI,  
        PROJECTION,  
        SELECTION,  
        mSelectionArgs,  
        null  
    );  
}
```

## Implement onLoadFinished() and onLoaderReset()



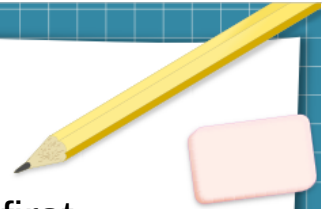
- Implement the onLoadFinished() method.
- The loader framework calls onLoadFinished() when the Contacts Provider returns the results of the query.
- In this method, put the result Cursor in the SimpleCursorAdapter

## Match a contact by a specific type of data



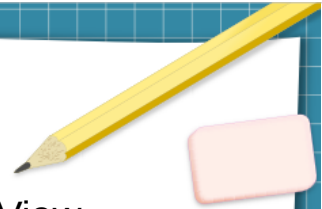
- This technique allows you to specify the type of data you want to match.
- Retrieving by name is a specific example of this type of query, but you can also do it for any of the types of detail data associated with a contact.
- For example, you can retrieve contacts that have a specific postal code; in this case, the search string has to match data stored in a postal code row.

## Cont ...



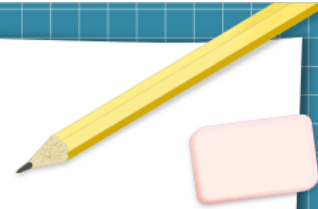
- To implement this type of retrieval, first implement the following code, as listed in previous sections:
  - Request Permission to Read the Provider.
  - Define ListView and item layouts.
  - Define a Fragment that displays the list of contacts.
  - Define global variables.
  - Initialize the Fragment.

## Cont ...



- Set up the CursorAdapter for the ListView.
- Set the selected contact listener.
- Define constants for the Cursor column indexes.
- Although you're retrieving data from a different table, the order of the columns in the projection is the same, so you can use the same indexes for the Cursor.
- Define the onItemClick() method.
- Initialize the loader.
- Implement onLoadFinished() and onLoaderReset().

## Cont ...



- In addition:
  - Choose the data type and table
  - Define a projection
  - Define search criteria
  - Implement onCreateLoader()



## Match a contact by any type of data

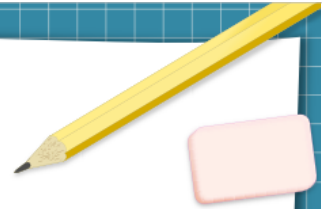


- Retrieving a contact based on any type of data returns contacts if any of their data matches a the search string, including name, email address, postal address, phone number, and so forth.
- This results in a broad set of search results.
- For example, if the search string is "Doe", then searching for any data type returns the contact "John Doe"; it also returns contacts who live on "Doe Street".

## Cont ...

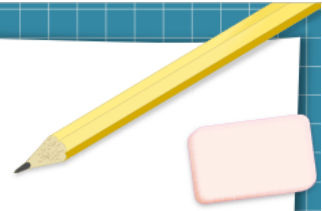
- To implement this type of retrieval, first implement the following code, as listed in previous sections:
  - Request Permission to Read the Provider.
  - Define ListView and item layouts.
  - Define a Fragment that displays the list of contacts.
  - Define global variables.
  - Initialize the Fragment.
  - Set up the CursorAdapter for the ListView.
  - Set the selected contact listener.

## Cont ...

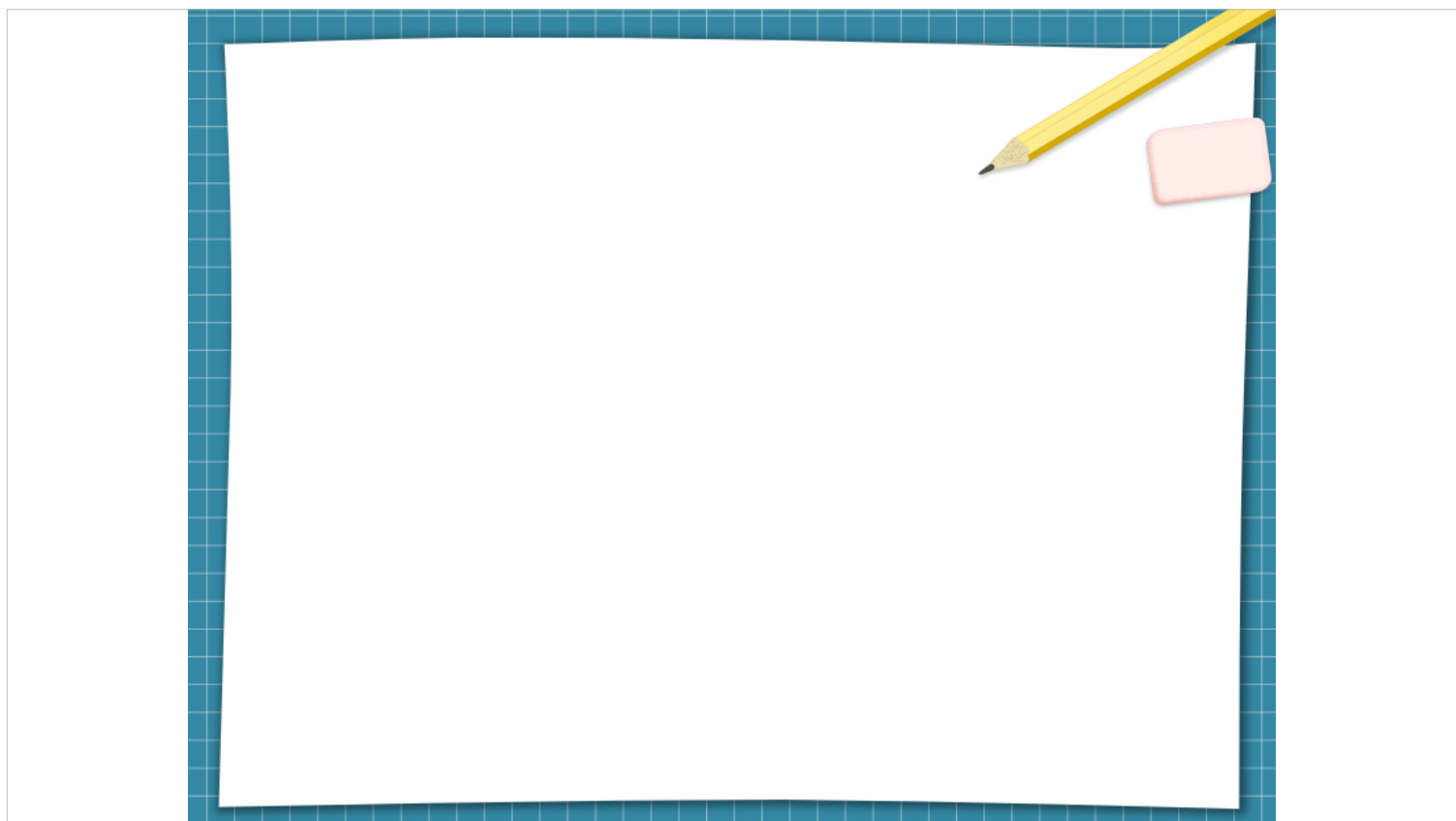


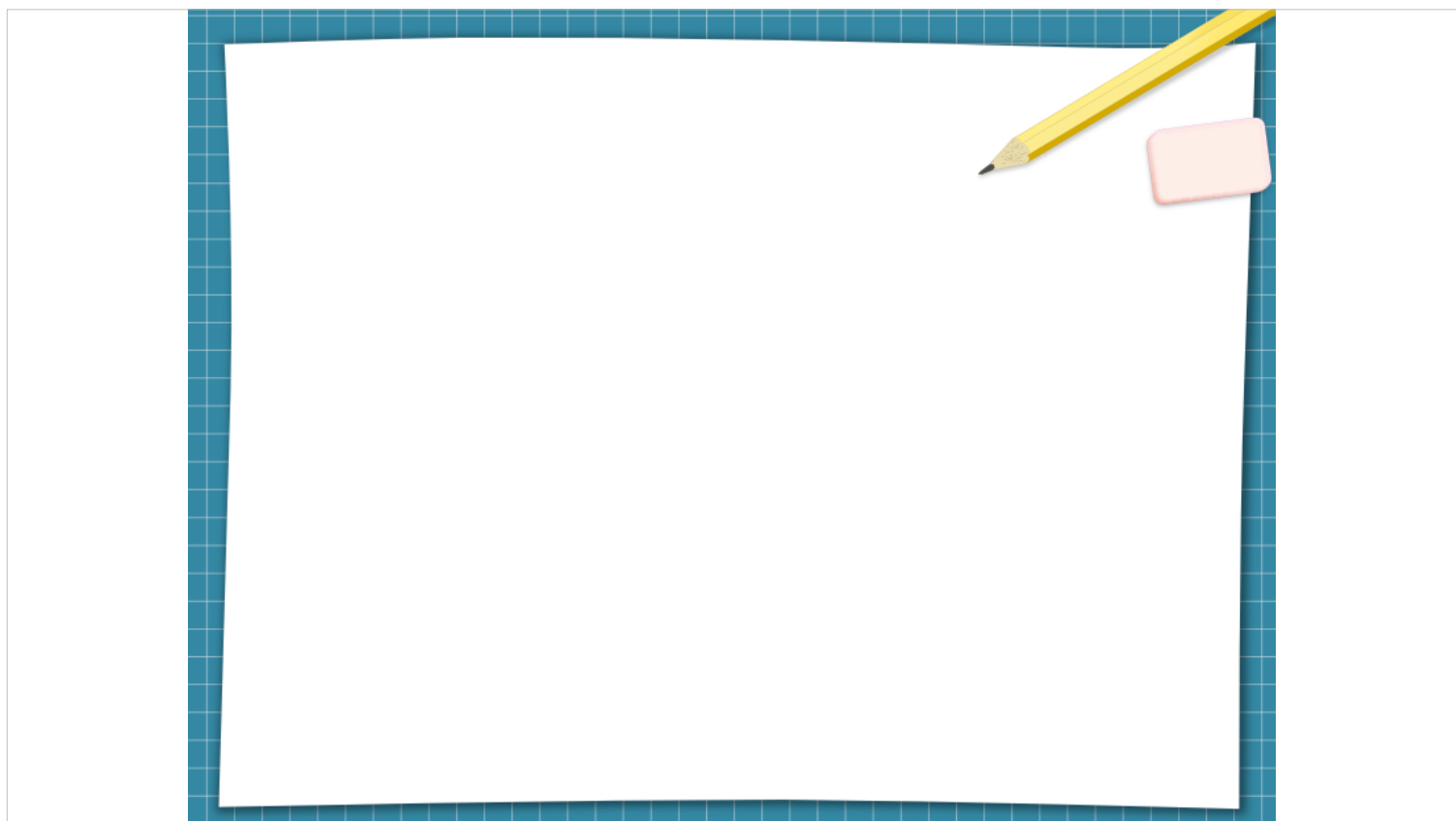
- Define a projection.
- Define constants for the Cursor column indexes.
- For this type of retrieval, you're using the same table you used in the section Match a contact by name and list the results. Use the same column indexes as well.
- Define the onItemClick() method.
- Initialize the loader.
- Implement onLoadFinished() and onLoaderReset().

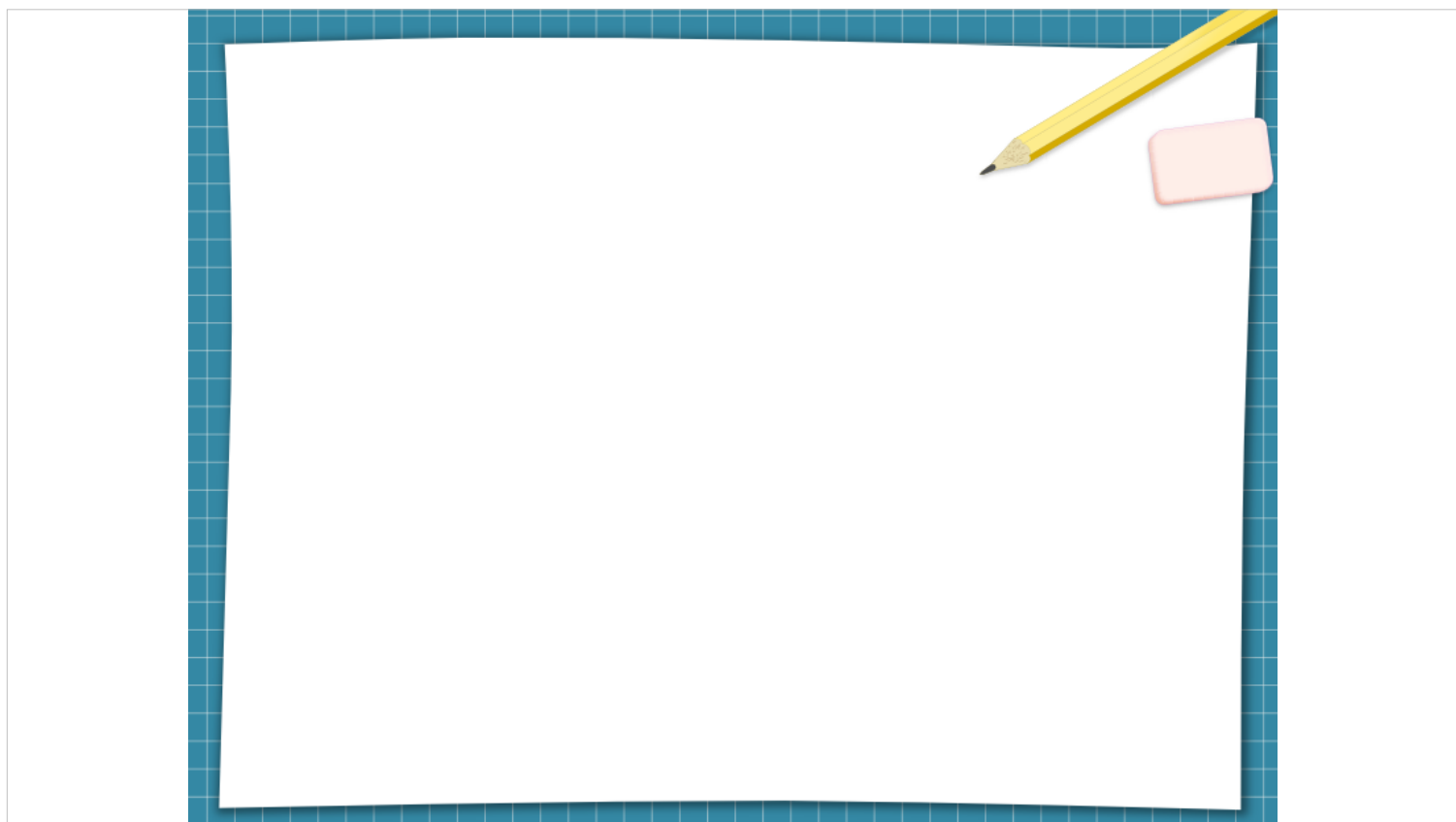
## Cont ...

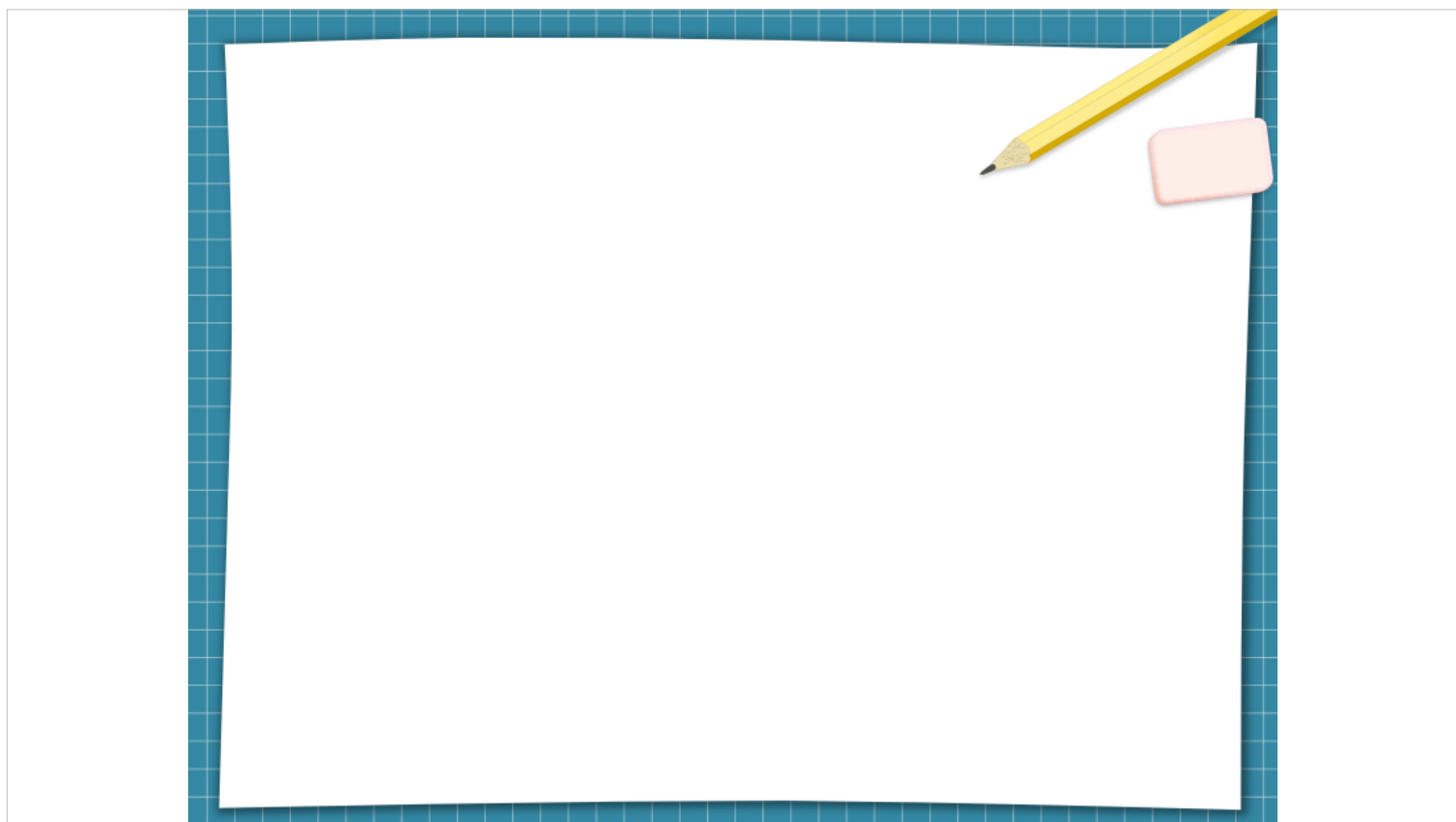


- In addition:
  - Remove selection criteria
  - Implement onCreateLoader()

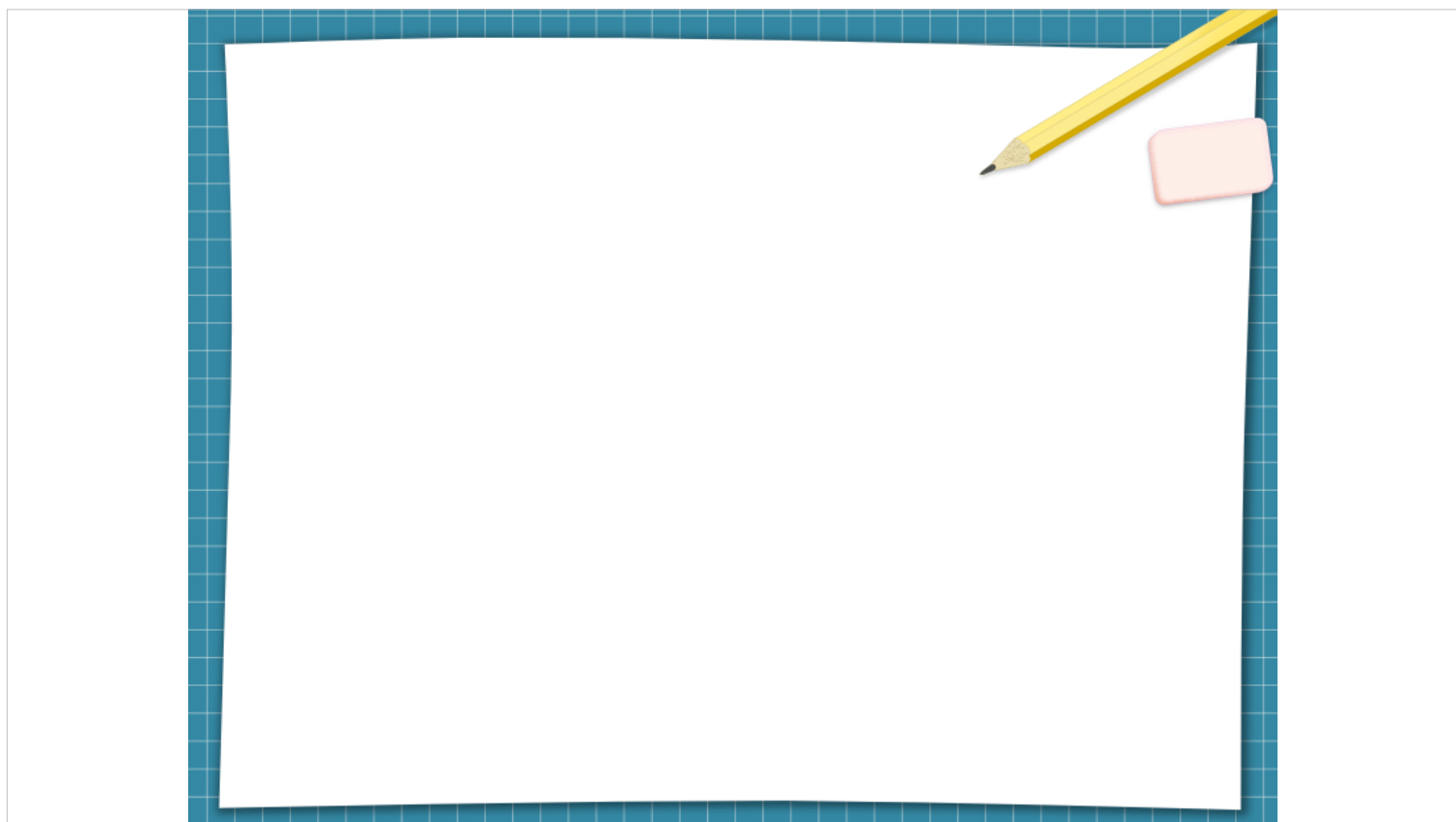


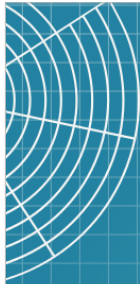












This work is licensed under a Creative Commons  
Attribution-ShareAlike 3.0 Unported License.  
It makes use of the works of Mateus Machado Luna.

