Lab08 - RecyclerView

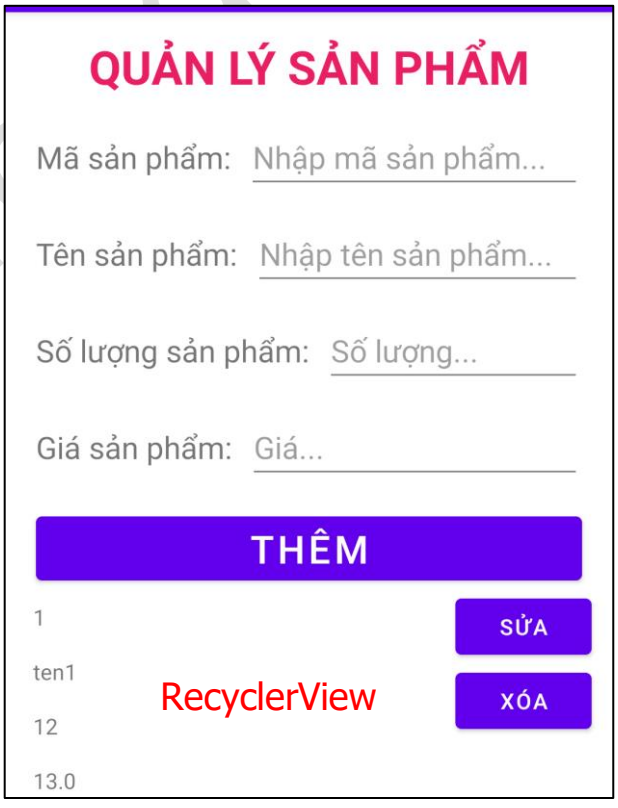
Trong Android, RecyclerView là một thành phần giao diện người dùng (UI) được sử dụng để hiển thị danh sách dữ liệu trong một cách linh hoạt và hiệu quả. Nó là một phiên bản cải tiến của ListView và GridView, được thiết kế để giải quyết các vấn đề hiệu suất khi làm việc với danh sách dữ liệu lớn.

RecyclerView cho phép hiển thị danh sách dữ liệu theo dạng danh sách (list) hoặc theo dạng lưới (grid), và cung cấp các tính năng như tái sử dụng các thành phần giao diện (View), quản lý việc cuộn (scrolling) một cách hiệu quả, và tùy chỉnh linh hoạt hơn.

Để sử dụng RecyclerView trong ứng dụng Android, cần tạo một Adapter để kết nối dữ liệu với RecyclerView và một ViewHolder để hiển thị từng mục dữ liệu trong danh sách. Sau đó, cấu hình RecyclerView trong layout XML và liên kết nó với Adapter tương ứng để hiển thị dữ liệu.

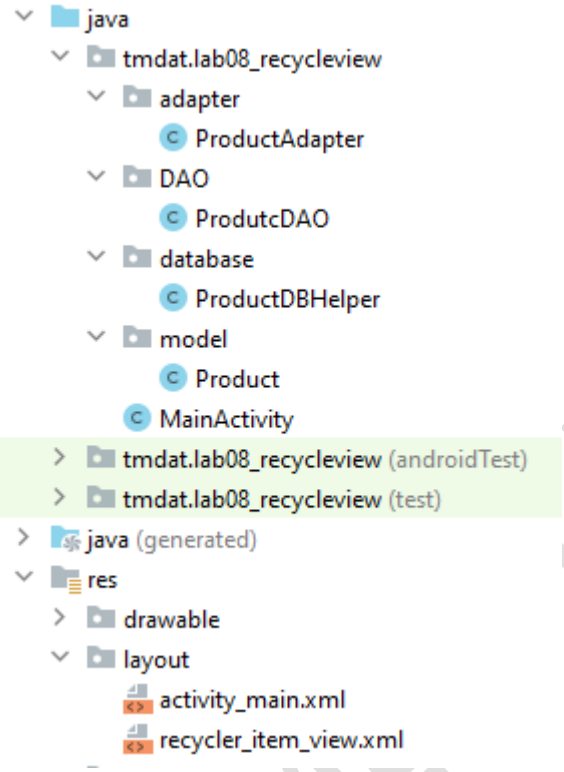
Bài toán:

Tạo ứng dụng quản lý sản phẩm có chức năng thêm, sửa, xóa thông tin, dữ liệu về sản phẩm được hiển thị bằng RecyclerView:



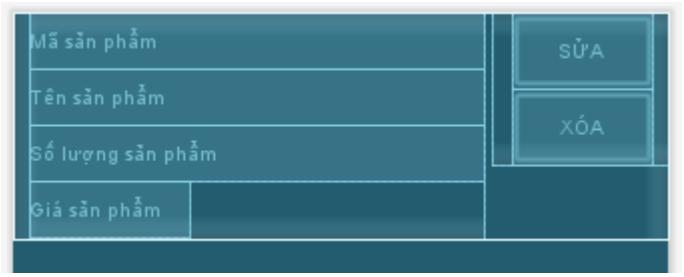
Hướng dẫn thực hiện:

Cấu trúc dự án trong ví dụ này bố trí như sau:



1. Xây dựng layout chính activity_main.xml

2. Xây dựng layout tùy chỉnh cho RecyclerView: recycler_item_view.xml



3. Xây dựng lớp Product.java

```
//Lớp Product.java
public class Product {
    3 usages
    private int maSP;
    3 usages
    private String tenSP;
    3 usages
    private int soSP;
    3 usages
    private double giaSP;
    public Product() {}
    2 usages
    public Product(int maSP, String tenSP, int soSP, double giaSP) {
        this.maSP = maSP;
        this.tenSP = tenSP;
        this.soSP = soSP;
        this.giaSP = giaSP;
    }
    4 usages
    public int getMaSP() { return maSP; }
    public void setMaSP(int maSP) { this.maSP = maSP; }
    3 usages
    public String getTenSP() { return tenSP; }
    public void setTenSP(String tenSP) { this.tenSP = tenSP; }
    3 usages
    public int getSoSP() { return soSP; }
    public void setSoSP(int soSP) { this.soSP = soSP; }
    3 usages
    public double getGiaSP() { return giaSP; }
    public void setGiaSP(double giaSP) { this.giaSP = giaSP; }
}
```

4. Xây dựng lớp ProductDBHelper.java

```
public class ProductDBHelper extends SQLiteOpenHelper {
    private static final String DB_NAME = "ProductDB";
    private static final int DB_VERSION = 1;
    public ProductDBHelper(Context context){
        //Tạo cơ sở dữ liệu
        super(context,DB_NAME,null,DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        //Lệnh SQL tạo bảng dữ liệu
        String create_table_sql="CREATE TABLE tableProduct(\n" +
                "\tmaSP integer PRIMARY KEY,\n" +
                "  tenSP text,\n" +
                "  soSP integer,\n" +
                "  giaSP double\n" +
                ")";
        //Thực thi câu lệnh tạo bảng
        sqLiteDatabase.execSQL(create_table_sql);
        //Lệnh SQL thêm dữ liệu mẫu vào bảng
        // Thêm dữ liệu mẫu nếu bảng rỗng
        //Ban đầu hoàn toàn có thể chỉ cần tạo bảng, không cần có đoạn thêm dữ liệu
mẫu
        //Khi insert từ layout cần thực hiện kiểm tra xem ID đã tồn tại hay chưa
        if (isTableEmpty(sqLiteDatabase, "tableProduct")) {
            String insert_table_sql = "INSERT INTO tableProduct VALUES
(1,'ten1',12,13),(2,'ten2',22,23)";
            sqLiteDatabase.execSQL(insert_table_sql);
        }
    }
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int
newVersion) {
        if(oldVersion!=newVersion){
            sqLiteDatabase.execSQL("DROP TABLE IF EXISTS tableProduct");
            onCreate(sqLiteDatabase);
        }
    }
    private boolean isTableEmpty(SQLiteDatabase db, String tableName) {
        Cursor cursor = db.rawQuery("SELECT count(*) FROM " + tableName, null);
        if (cursor != null) {
            cursor.moveToFirst();
            int count = cursor.getInt(0);
```

```
            cursor.close();
            return count == 0;
        }
        return true;
    }
}
```

5. Xây dựng lớp ProductDAO:

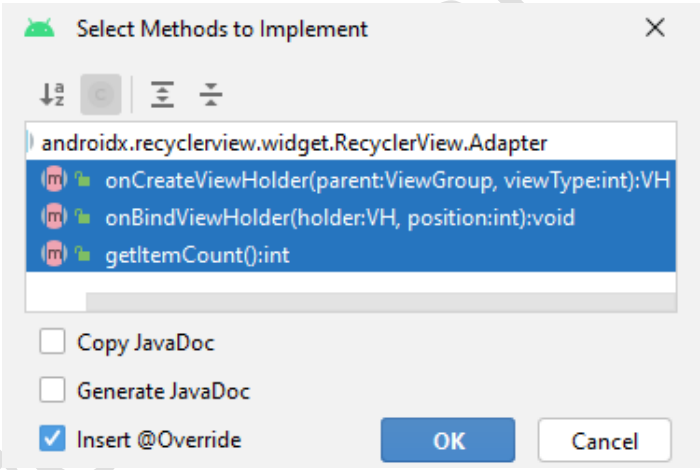
```
public class ProdutcDAO {
    private ProductDBHelper productDBHelper;
    private SQLiteDatabase sqLiteDatabase;
    public ProdutcDAO(Context context){
        productDBHelper = new ProductDBHelper(context);//Gọi lệnh tạo DB
    }
    // Phương thức Mở - Đóng CSDL
    public void open(){
        sqLiteDatabase=productDBHelper.getWritableDatabase();
    }
    public void close(){
        productDBHelper.close();
    }
    //Hàm Thêm: CREATE (C trong CRUD)
    public long addProdut(Product product){
        ContentValues contentValues = new ContentValues();
        contentValues.put("maSP", product.getMaSP());
        contentValues.put("tenSP", product.getTenSP());
        contentValues.put("soSP", product.getSoSP());
        contentValues.put("giaSP", product.getGiaSP());
        return sqLiteDatabase.insert("tableProduct", null, contentValues);
    }
    //Hàm xóa: DELETE (D trong CRUD)
    public boolean deleteProduct(int maSP) {
        int rowsAffected = sqLiteDatabase.delete("tableProduct", "maSP=?", new
String[]{String.valueOf(maSP)});
        return rowsAffected > 0;
    }
    //Hàm sửa: UPDATE (U trong CRUD)
    public boolean updateProduct(Product product){
        SQLiteDatabase db = productDBHelper.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("tenSP", product.getTenSP());
        values.put("soSP", product.getSoSP());
        values.put("giaSP", product.getGiaSP());
```

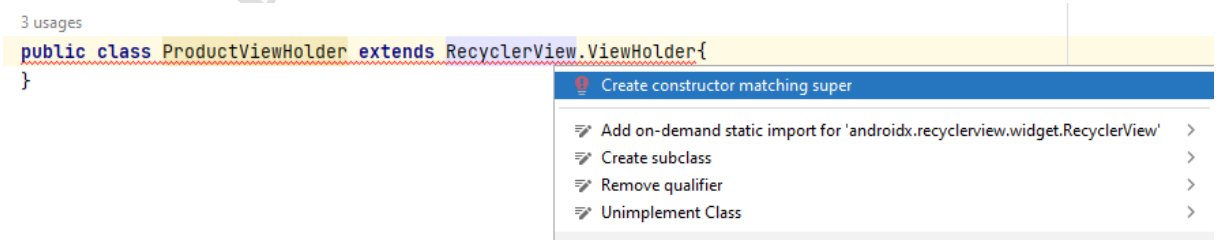
```
        // Sử dụng phương thức update() để cập nhật dữ liệu
        int rowsAffected = db.update("tableProduct", values, "maSP = ?", new
String[] { String.valueOf(product.getMaSP()) });
        db.close();

        // Kiểm tra xem dữ liệu có được cập nhật thành công hay không
        return rowsAffected > 0;
    }
    //Hàm đọc dữ liệu:READ (R trong CRUD)
    public List<Product> getListProduct() {
        List<Product> productList = new ArrayList<>();
        Cursor cursor = null;
        try {
            cursor = sqLiteDatabase.rawQuery("SELECT * FROM tableProduct", null);
            if (cursor != null && cursor.moveToFirst()) {
                int maSPIndex = cursor.getColumnIndex("maSP");
                int tenSPIndex = cursor.getColumnIndex("tenSP");
                int soSPIndex = cursor.getColumnIndex("soSP");
                int giaSPIndex = cursor.getColumnIndex("giaSP");
                do {
                    if (maSPIndex != -1 && tenSPIndex != -1 && soSPIndex != -1 &&
giaSPIndex != -1) {
                        int maSP = cursor.getInt(maSPIndex);
                        String tenSP = cursor.getString(tenSPIndex);
                        int soSP = cursor.getInt(soSPIndex);
                        double giaSP = cursor.getDouble(giaSPIndex);
                        Product product = new Product(maSP, tenSP, soSP, giaSP);
                        productList.add(product);
                    } else {
                        Log.e("ProdutcDAO", "Không tìm thấy chỉ mục cột");
                    }
                } while (cursor.moveToNext());
            }
        } catch (Exception e) {
            Log.e("ProdutcDAO", "Lỗi khi lấy thông tin sản phẩm từ CSDL", e);
        } finally {
            if (cursor != null) {
                cursor.close();
            }
        }
        return productList;
    }
}
```

6. Xây dựng lớp ProductAdapter.java

Chú ý thực hiện theo trình tự hướng dẫn để hoàn thành ProductAdapter:

```
21  //Gõ xong dòng public class... bên dưới thì nó sẽ có gạch đỏ
22  //Click vào dòng đó rồi chọn "implement method" để tự thêm vào các phương thức bên trong public
23  //Sau đó mới hoàn thiện các phương thức
    5 usages
24  public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {
25
26      @NonNull
27      @Override
28      public ProductAdapter.ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
29          return null;
30      }
31
32      @Override
33      public void onBindViewHolder(@NonNull ProductAdapter.ProductViewHolder holder, int position) {
34
35      }
36
37      @Override
38      public int getItemCount() {
39          return 0;
40      }
41  }
```






```
3 usages
public class ProductViewHolder extends RecyclerView.ViewHolder{
    public ProductViewHolder(@NonNull View itemView) {
        super(itemView);
    }
}
```

```java
public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {

    @NonNull
    @Override
    public ProductAdapter.ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull ProductAdapter.ProductViewHolder holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }

    3 usages
    public class ProductViewHolder extends RecyclerView.ViewHolder{
        public ProductViewHolder(@NonNull View itemView) {
            super(itemView);
        }
    }
}
```

Sau khi hoàn thành xong trình tự trên mới hoàn thiện lớp ProductAdapter:

```java
//Gõ xong dòng public class... bên dưới thì nó sẽ có gạch đỏ
//Click vào dòng đó rồi chọn "implement method" để tự thêm vào các phương thức bên trong public
//Sau đó mới hoàn thiện các phương thức
public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {
    private Context context;
    private List<Product> productList;
    private OnItemClickListener listener;
    public interface OnItemClickListener {
        void onEditClick(int position);
        void onDeleteClick(int position);
    }
    public void setOnItemClickListener(OnItemClickListener listener){
        this.listener = listener;
    }
    public ProductAdapter(Context context,List<Product>productList){
        this.context = context;
        this.productList = productList;
    }

    @NonNull
    @Override
    public ProductAdapter.ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.recycler_item_view,parent,false);
        return new ProductViewHolder(view);
    }

    @Override
```

```
    public void onBindViewHolder(@NonNull ProductAdapter.ProductViewHolder holder, int position) {
        Product product = productList.get(position);
        holder.bind(product);
    }

    @Override
    public int getItemCount() {
        return productList.size();
    }

    public class ProductViewHolder extends RecyclerView.ViewHolder{
        private TextView
textViewItemID,textViewItemProductName,textViewItemQuantityProduct,textViewItemPriceProduct;
        private Button buttonEdit, buttonDelete;
        public ProductViewHolder(@NonNull View itemView) {
            super(itemView);
            textViewItemID = itemView.findViewById(R.id.textViewItemID);
            textViewItemProductName=itemView.findViewById(R.id.textViewItemProductName);
            textViewItemQuantityProduct=itemView.findViewById(R.id.textViewItemProductQuantity);
            textViewItemPriceProduct=itemView.findViewById(R.id.textViewItemProductPrice);
            buttonEdit = itemView.findViewById(R.id.buttonEdit);
            buttonDelete = itemView.findViewById(R.id.buttonDelete);
            //Sự kiện Sửa - Xóa cũng được xử lý ở đây
            buttonEdit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (listener != null) {
```

```
                    int position = getAdapterPosition();
                    if (position != RecyclerView.NO_POSITION) {
                        listener.onEditClick(position);
                    }
                }
            }
        });
        //Sự kiện xóa
        buttonDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (listener != null) {
                    int position = getAdapterPosition();
                    if (position != RecyclerView.NO_POSITION) {
                        listener.onDeleteClick(position);
                    }
                }
            }
        });
    }
    public void bind(Product product){
        textViewItemID.setText(String.valueOf(product.getMaSP()));
        textViewItemProductName.setText(product.getTenSP());//Name là kiểu String nên không cần convert
        textViewItemQuantityProduct.setText(String.valueOf(product.getSoSP()));
        textViewItemPriceProduct.setText(String.valueOf(product.getGiaSP()));
    }
}
```

```
}
```

6. Xây dựng lớp xử lý MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private EditText editTextMaSP, editTextTenSP, editTextSoSP, editTextGiaSP;
    private Button buttonThem;
    private RecyclerView recyclerViewListProduct;
    private ProductAdapter productAdapter;
    private List<Product> productList;
    private ProdutcDAO productDAO;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Ánh xạ các view
        editTextMaSP = findViewById(R.id.editTextMaSP);
        editTextTenSP = findViewById(R.id.editTextTenSP);
        editTextSoSP = findViewById(R.id.editTextSoSP);
        editTextGiaSP = findViewById(R.id.editTextGiaSP);
        buttonThem = findViewById(R.id.buttonThem);
        recyclerViewListProduct = findViewById(R.id.recyclerViewListProduct);

        // Khởi tạo DAO và danh sách sản phẩm
        productDAO = new ProdutcDAO(this);
        productList = new ArrayList<>();
```

```
// Khởi tạo adapter và thiết lập RecyclerView
productAdapter = new ProductAdapter(this, productList);
recyclerViewListProduct.setLayoutManager(new LinearLayoutManager(this));
recyclerViewListProduct.setAdapter(productAdapter);

// Mở cơ sở dữ liệu trước khi tải danh sách sản phẩm
productDAO.open();
// Load danh sách sản phẩm từ cơ sở dữ liệu và cập nhật RecyclerView
loadProductList();

// Xử lý sự kiện khi nhấn nút "Thêm"
buttonThem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Lấy dữ liệu từ các EditText
        int maSP = Integer.parseInt(editTextMaSP.getText().toString());
        String tenSP = editTextTenSP.getText().toString();
        int soSP = Integer.parseInt(editTextSoSP.getText().toString());
        double giaSP = Double.parseDouble(editTextGiaSP.getText().toString());

        // Tạo đối tượng Product mới
        Product product = new Product(maSP, tenSP, soSP, giaSP);

        // Mở cơ sở dữ liệu trước khi thêm sản phẩm
        productDAO.open();
```

```
                // Thêm sản phẩm vào cơ sở dữ liệu và cập nhật RecyclerView
                long result = productDAO.addProdut(product);
                if (result != -1) {
                    productList.add(product);
                    productAdapter.notifyDataSetChanged();
                }
                // Đóng cơ sở dữ liệu sau khi thêm sản phẩm
                productDAO.close();
                // Xóa dữ liệu trong các EditText
                editTextMaSP.setText("");
                editTextTenSP.setText("");
                editTextSoSP.setText("");
                editTextGiaSP.setText("");
            }
        });
        // Xử lý sự kiện khi nhấn nút "Sửa" – “Xóa”
    }
    // Load danh sách sản phẩm từ cơ sở dữ liệu
    private void loadProductList() {
        productList.clear();
        productList.addAll(productDAO.getListProduct());
        productAdapter.notifyDataSetChanged();
        // Đóng cơ sở dữ liệu sau khi tải danh sách sản phẩm
        productDAO.close();
    }
}
```

Hoàn thiện chức năng Sửa và Xóa

1. Xây dựng layout dùng khi sửa thông tin sản phẩm: dialog_edit_product.xml

2. Hoàn thiện Adapter.

Trong ProductAdapter, thêm vào phương thức ProductViewHolder xử lý sự kiện Sửa – Xóa:

```
public ProductViewHolder(@NonNull View itemView) {
    super(itemView);
    textViewItemID = itemView.findViewById(R.id.textViewItemID);
    textViewItemProductName=itemView.findViewById(R.id.textViewItemProductName);
    textViewItemQuantityProduct=itemView.findViewById(R.id.textViewItemProductQuantity);
    textViewItemPriceProduct=itemView.findViewById(R.id.textViewItemProductPrice);
    buttonEdit = itemView.findViewById(R.id.buttonEdit);
    buttonDelete = itemView.findViewById(R.id.buttonDelete);
    //Sự kiện Sửa - Xóa cũng được xử lý ở đây
    buttonEdit.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            if (listener != null) {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    listener.onEditClick(position);
                }
            }
        }
    });
    //Sự kiện xóa
    buttonDelete.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (listener != null) {
                int position = getAdapterPosition();
                if (position != RecyclerView.NO_POSITION) {
                    listener.onDeleteClick(position);
                }
            }
        }
    });
}
```

3. Hoàn thiện MainActivity.

Trong chức năng Sửa thông tin sản phẩm, có sử dụng AlertDialog, AlertDialog là một loại Dialog cơ bản trong Android, được sử dụng để hiển thị một thông điệp và một hoặc nhiều lựa chọn cho người dùng. Nó là một cách phổ biến để yêu cầu xác nhận hoặc thông báo cho người dùng về một hành động cụ thể.

Cấu trúc của AlertDialog:

- AlertDialog bao gồm các thành phần chính sau:
- Tiêu đề (Title): Tiêu đề của Dialog, thường là một chuỗi văn bản ngắn để mô tả mục đích hoặc nội dung của Dialog.
- Nội dung (Content): Phần nội dung chính của Dialog, thường là một thông điệp hoặc mô tả.
- Các lựa chọn (Options): Các lựa chọn mà người dùng có thể chọn, thường là các nút như "OK", "Cancel", "Yes", "No"...

Cách sử dụng AlertDialog:

- Để tạo và sử dụng AlertDialog trong ứng dụng Android của bạn, bạn có thể làm như sau:
- Tạo một instance của AlertDialog.Builder: Bạn sẽ sử dụng AlertDialog.Builder để xây dựng và cấu hình AlertDialog.
- Thiết lập các thuộc tính cho AlertDialog: Bạn có thể thiết lập tiêu đề, nội dung, các lựa chọn và các tính năng khác của AlertDialog thông qua phương thức của AlertDialog.Builder như setTitle(), setMessage(), setPositiveButton(), setNegativeButton()...
- Gọi phương thức create() của AlertDialog.Builder để tạo ra AlertDialog.
- Gọi phương thức show() trên AlertDialog để hiển thị nó lên màn hình.

Ví dụ về cách sử dụng AlertDialog chứa 2 nút Yes và No

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);
builder.setTitle("Confirmation");
builder.setMessage("Are you sure you want to delete this item?");
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // Xử lý khi người dùng chọn "Yes"
    }
});
builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // Xử lý khi người dùng chọn "No"
        dialog.dismiss(); // Đóng Dialog
    }
});
AlertDialog dialog = builder.create();
dialog.show();
```

Áp dụng vào bài thực hành.

Trong phương thức onCreate, bổ sung phần xử lý khi click các nút Sửa và Xóa:

```
// Xử lý sự kiện click cho adapter
productAdapter.setOnItemClickListener(new ProductAdapter.OnItemClickListener() {
    @Override
    public void onEditClick(int position) {
        // Lấy sản phẩm cần sửa từ danh sách sản phẩm
        Product productToEdit = productList.get(position);
        // Hiển thị dialog sửa sản phẩm
        showEditProductDialog(productToEdit, position);
    }
    @Override
    public void onDeleteClick(int position) {
        // Xác nhận xóa sản phẩm
        showDeleteConfirmationDialog(position);
    }
});
```

Trong lớp MainActivity xây dựng các phương thức showEditProductDialog(productToEdit, position) và showDeleteConfirmationDialog(position);

```
// Hiển thị dialog sửa sản phẩm
private void showEditProductDialog(Product productToEdit, int position) {
    //Khai báo đối tượng Dialog
    AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);
```

```
    LayoutInflater inflater = getLayoutInflater();
    //Dialog được tạo dựa trên dialog_edit_product.xml gồm:
    //textViewMaSP, editTextTenSP, editTextSoSP, editTextGiaSP
    View dialogView = inflater.inflate(R.layout.dialog_edit_product, null);
    dialogBuilder.setView(dialogView);
    //Thông tin mã sản phẩm không cần sửa nên dùng TextView
    final TextView textViewMaSP = dialogView.findViewById(R.id.textViewMaSP);
    final EditText editTextTenSP = dialogView.findViewById(R.id.editTextTenSP);
    final EditText editTextSoSP = dialogView.findViewById(R.id.editTextSoSP);
    final EditText editTextGiaSP = dialogView.findViewById(R.id.editTextGiaSP);

    // Hiển thị thông tin sản phẩm hiện tại trong dialog
    textViewMaSP.setText(String.valueOf(productToEdit.getMaSP()));
    editTextTenSP.setText(productToEdit.getTenSP());
    editTextSoSP.setText(String.valueOf(productToEdit.getSoSP()));
    editTextGiaSP.setText(String.valueOf(productToEdit.getGiaSP()));

    //Cấu hình Dialog
    dialogBuilder.setTitle("Sửa sản phẩm");
    //Tạo các nút và thiết lập sự kiện cho các nút trên Dialog
    //Nút Save
    dialogBuilder.setPositiveButton("Save", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            // Lấy thông tin đã sửa từ dialog
            int maSP = Integer.parseInt(textViewMaSP.getText().toString());
            String tenSP = editTextTenSP.getText().toString();
            int soSP = Integer.parseInt(editTextSoSP.getText().toString());
```

```
            double giaSP = Double.parseDouble(editTextGiaSP.getText().toString());

            // Tạo sản phẩm mới
            Product editedProduct = new Product(maSP, tenSP, soSP, giaSP);

            // Cập nhật sản phẩm trong cơ sở dữ liệu
            productDAO.open();
            boolean updated = productDAO.updateProduct(editedProduct);
            productDAO.close();

            // Nếu cập nhật thành công, cập nhật lại RecyclerView
            if (updated) {
                productList.set(position, editedProduct);
                productAdapter.notifyItemChanged(position);
                Toast.makeText(MainActivity.this, "Sửa sản phẩm thành công", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(MainActivity.this, "Không thể sửa sản phẩm", Toast.LENGTH_SHORT).show();
            }
        }
    });
    //Nút Cancel
    dialogBuilder.setNegativeButton("Cancel", null);
    //Tạo đối tượng AlertDialog từ AlertDialog.Builder đã được cấu hình trước đó và hiển thị lên
    AlertDialog alertDialog = dialogBuilder.create();
    alertDialog.show();
}
// Hiển thị dialog xác nhận xóa sản phẩm
```

```
private void showDeleteConfirmationDialog(final int position) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Xác nhận xóa sản phẩm");
    builder.setMessage("Bạn có chắc chắn muốn xóa sản phẩm này?");
    builder.setPositiveButton("Xóa", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            // Xóa sản phẩm khỏi cơ sở dữ liệu
            productDAO.open();
            boolean deleted = productDAO.deleteProduct(productList.get(position).getMaSP());
            productDAO.close();

            // Nếu xóa thành công, cập nhật RecyclerView
            if (deleted) {
                productList.remove(position);
                productAdapter.notifyItemRemoved(position);
                Toast.makeText(MainActivity.this, "Đã xóa sản phẩm", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(MainActivity.this, "Không thể xóa sản phẩm", Toast.LENGTH_SHORT).show();
            }
        }
    });
    builder.setNegativeButton("Hủy", null);
    builder.create().show();
}
```