Lab09 – Xử lý nhiều bảng dữ liệu

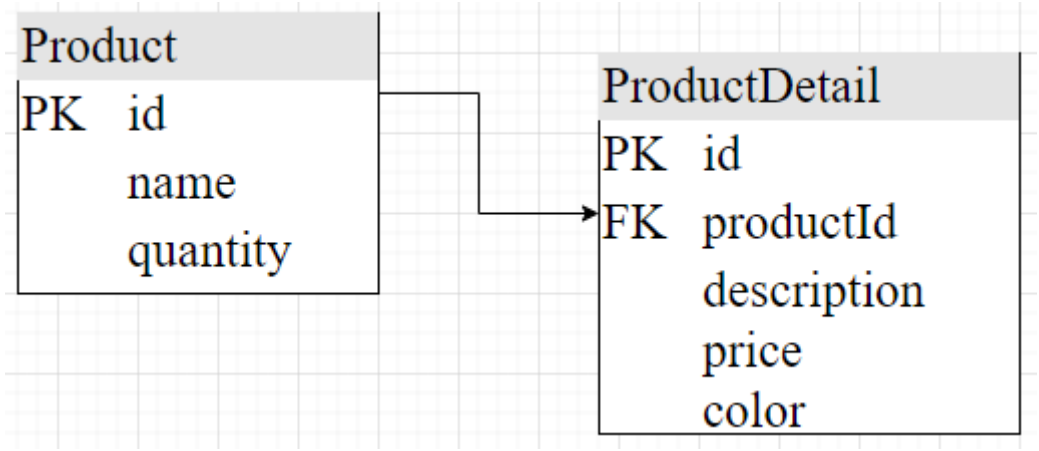
Bài toán:

Tạo ứng dụng quản lý sản phẩm gồm các thông tin: Mã sản phẩm, tên sản phẩm, số lượng, mô tả, màu sắc, giá sản phẩm. Chi tiết demo trong link:

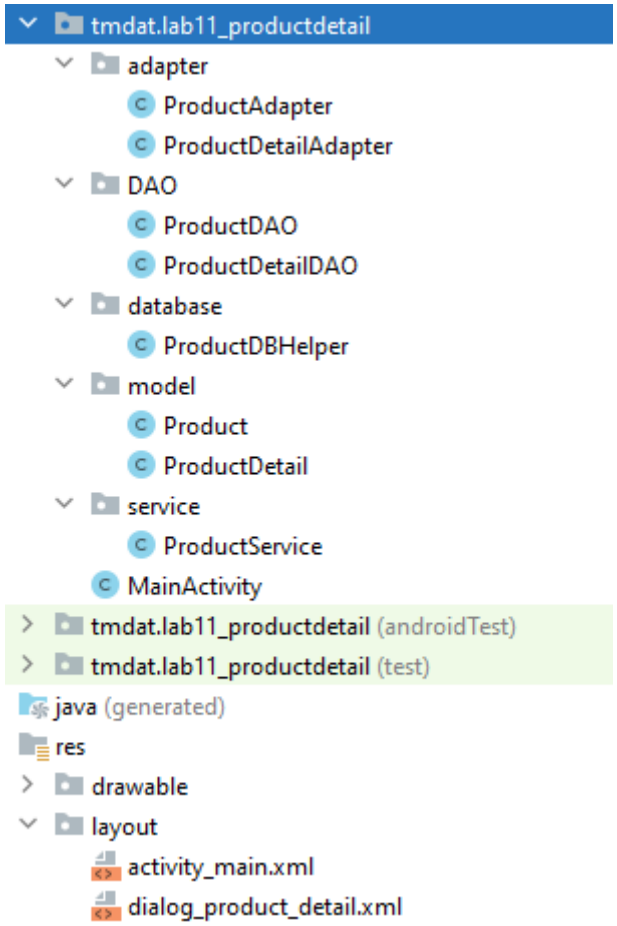
https://youtu.be/a08UtjirktA

Hướng dẫn thực hiện:

Dự án có 2 bảng dữ liệu, giữa 2 bảng có quan hệ với nhau:



Cấu trúc của dự án gồm các lớp và layout:



Chức năng cơ bản của mỗi thành phần:

- **Product và ProductDetail**: Đây là các đối tượng dữ liệu đại diện cho sản phẩm và chi tiết sản phẩm trong ứng dụng của bạn. Chúng chứa thông tin như tên sản phẩm, mô tả, giá cả, số lượng, màu sắc và các thông tin khác liên quan đến sản phẩm và chi tiết sản phẩm.
- **ProductDBHelper**: Lớp này là trợ lý cơ sở dữ liệu SQLite trong ứng dụng của bạn. Nó chứa định nghĩa cho cơ sở dữ liệu, bao gồm bảng products và product_details, và các phương thức để tạo, nâng cấp và quản lý cơ sở dữ liệu.

- **ProductDAO và ProductDetailDAO**: Đây là các lớp truy cập dữ liệu (DAO) cho sản phẩm và chi tiết sản phẩm. Chúng cung cấp các phương thức để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên cơ sở dữ liệu cho các đối tượng tương ứng.
- **ProductAdapter và ProductDetailAdapter**: Các Adapter này được sử dụng để hiển thị danh sách sản phẩm và chi tiết sản phẩm trong giao diện người dùng của bạn, chẳng hạn như danh sách ListView hoặc RecyclerView. Chúng tạo ra các item view để hiển thị thông tin của mỗi sản phẩm và chi tiết sản phẩm.
- **ProductService**: Lớp này là một lớp trung gian giữa các thành phần quản lý dữ liệu và cơ sở dữ liệu. Nó cung cấp các phương thức để thực hiện các thao tác liên quan đến dữ liệu sản phẩm và chi tiết sản phẩm, như thêm mới, lấy danh sách, cập nhật và xóa.
- **MainActivity**: Đây là hoạt động chính của ứng dụng của bạn, nơi bạn tương tác với người dùng và quản lý luồng làm việc chính. Trong MainActivity, bạn sẽ tạo ra giao diện người dùng, xử lý sự kiện và gọi các phương thức từ các lớp DAO hoặc Service để thao tác với dữ liệu.

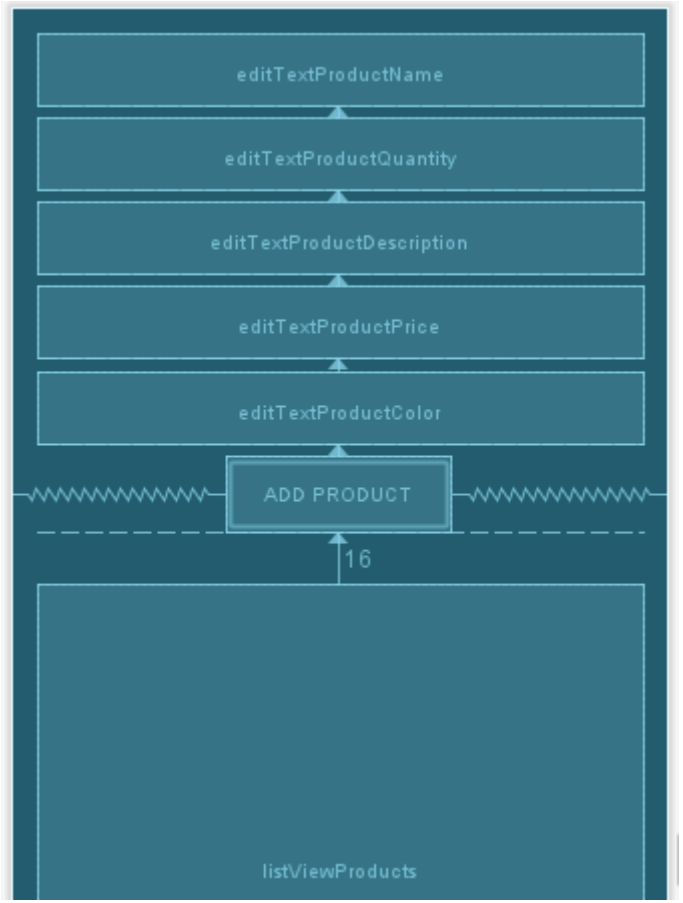
Luồng dữ liệu trong ứng dụng của bạn có thể được mô tả như sau:

- MainActivity: MainActivity là hoạt động chính của ứng dụng, nơi bạn tạo giao diện người dùng và quản lý sự tương tác với người dùng.
    - Trong MainActivity, bạn gọi các phương thức từ ProductService để thực hiện các thao tác liên quan đến dữ liệu sản phẩm và chi tiết sản phẩm.
    - Khi người dùng tương tác với giao diện, chẳng hạn nhấn nút để thêm hoặc xem danh sách sản phẩm, MainActivity sẽ gọi các phương thức tương ứng trong ProductService để xử lý yêu cầu.
- ProductService: Lớp ProductService là lớp trung gian giữa MainActivity và các lớp DAO (ProductDAO và ProductDetailDAO).
    - Trong ProductService, bạn triển khai các phương thức để thực hiện các thao tác CRUD trên cơ sở dữ liệu, như thêm mới sản phẩm, lấy danh sách sản phẩm, cập nhật thông tin sản phẩm, xóa sản phẩm, và tương tự cho chi tiết sản phẩm.

    - Khi được gọi từ MainActivity, ProductService sẽ gọi các phương thức tương ứng trong các lớp DAO để thao tác với cơ sở dữ liệu.
- ProductDAO và ProductDetailDAO: Các lớp DAO làm nhiệm vụ truy cập cơ sở dữ liệu và thực hiện các thao tác cụ thể trên các bảng tương ứng.
    - Trong ProductDAO và ProductDetailDAO, bạn triển khai các phương thức để thực hiện các thao tác CRUD cụ thể trên cơ sở dữ liệu, bao gồm thêm mới, lấy danh sách, cập nhật và xóa các mục dữ liệu.
    - Các phương thức này sẽ trả về kết quả hoặc dữ liệu cần thiết cho ProductService để truyền lại cho MainActivity.
- ProductAdapter và ProductDetailAdapter: Các Adapter này là để hiển thị dữ liệu sản phẩm và chi tiết sản phẩm trong giao diện người dùng.
- MainActivity sẽ sử dụng các Adapter này để hiển thị danh sách sản phẩm và chi tiết sản phẩm trong ListView hoặc RecyclerView.
- Adapter sẽ lấy dữ liệu từ danh sách sản phẩm được cung cấp bởi MainActivity và sử dụng nó để hiển thị thông tin tương ứng trong giao diện người dùng.

Tóm lại, luồng dữ liệu sẽ đi từ MainActivity đến ProductService, từ ProductService đến các lớp DAO, và cuối cùng đến cơ sở dữ liệu. Khi có kết quả hoặc dữ liệu trả về, nó sẽ đi ngược lại từ DAO đến ProductService, từ ProductService đến MainActivity, và từ MainActivity đến Adapter để hiển thị trong giao diện người dùng.

1. Xây dựng layout chính activity_main.xml



2. Xây dựng model

ProductDetail.java

```java
//ProductDetail.java
public class ProductDetail {
    3 usages
    private int id;
    3 usages
    private int productId; // Khóa ngoại tham chiếu đến bảng Product
    3 usages
    private String description;
    3 usages
    private double price;
    3 usages
    private String color;
    2 usages
    public ProductDetail(int id, int productId, String description, double price, String color) {
        this.id = id;
        this.productId = productId;
        this.description = description;
        this.price = price;
        this.color = color;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    1 usage
    public int getProductId() { return productId; }
    1 usage
    public void setProductId(int productId) { this.productId = productId; }
    4 usages
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }
    4 usages
    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }
    public String getColor() { return color; }
    public void setColor(String color) { this.color = color; }
}
```

Product.java

```java
//Product.java
public class Product {
    3 usages
    private int id;
    3 usages
    private String name;
    3 usages
    private int quantity;
    2 usages
    private ProductDetail productDetail;
    2 usages
    public Product(int id, String name, int quantity) {
        this.id = id;
        this.name = name;
        this.quantity = quantity;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) {this.name = name; }
    4 usages
    public int getQuantity() {return quantity;}
    public void setQuantity(int quantity) {this.quantity = quantity;}
    4 usages
    public ProductDetail getProductDetail() {return productDetail;}
    1 usage
    public void setProductDetail(ProductDetail productDetail) {this.productDetail = productDetail;}
}
```

3. Xây dựng lớp ProductDBHelper.java

```java
public class ProductDBHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "product_database";
    private static final int DATABASE_VERSION = 1;

    // Table names
    public static final String TABLE_PRODUCT = "products";
    public static final String TABLE_PRODUCT_DETAIL = "product_details";
```

```
    // Common column names
    public static final String COLUMN_ID = "id";

    // Product table column names
    public static final String COLUMN_PRODUCT_NAME = "name";
    public static final String COLUMN_PRODUCT_QUANTITY = "quantity";

    // Product detail table column names
    public static final String COLUMN_PRODUCT_ID = "product_id";
    public static final String COLUMN_PRODUCT_DESCRIPTION = "description";
    public static final String COLUMN_PRODUCT_PRICE = "price";
    public static final String COLUMN_PRODUCT_COLOR = "color";

    // Create table statements
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCT + "(" +
            COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCT_NAME + " TEXT," +
            COLUMN_PRODUCT_QUANTITY + " INTEGER" + ")";

    private static final String CREATE_TABLE_PRODUCT_DETAIL = "CREATE TABLE " + TABLE_PRODUCT_DETAIL + "(" +
            COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCT_ID + " INTEGER," +
            COLUMN_PRODUCT_DESCRIPTION + " TEXT," +
            COLUMN_PRODUCT_PRICE + " REAL," +
            COLUMN_PRODUCT_COLOR + " TEXT" + "," +
            "FOREIGN KEY(" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCT + "(" + COLUMN_ID + ") ON DELETE
CASCADE)";

    public ProductDBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_PRODUCT_DETAIL);

    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older tables if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_PRODUCT);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_PRODUCT_DETAIL);

        // Create tables again
        onCreate(db);
    }
}
```

4. Xây dựng các DAO:

ProductDAO:

```
public class ProductDAO {
    private SQLiteDatabase database;
    private ProductDBHelper dbHelper;
    public ProductDAO(Context context) {
        dbHelper = new ProductDBHelper(context);
    }
    public void open() {
        database = dbHelper.getWritableDatabase();
    }
    public void close() {
        dbHelper.close();
    }
```

```
    // Tạo bảng mới
    public long addProduct(Product product) {
        ContentValues values = new ContentValues();
        values.put(ProductDBHelper.COLUMN_PRODUCT_NAME, product.getName());
        values.put(ProductDBHelper.COLUMN_PRODUCT_QUANTITY, product.getQuantity());
        return database.insert(ProductDBHelper.TABLE_PRODUCT, null, values);
    }
    // Lấy toàn bộ thông tin bảng product
    public List<Product> getAllProducts() {
        List<Product> products = new ArrayList<>();
        Cursor cursor = database.query(ProductDBHelper.TABLE_PRODUCT, null,
                null, null, null, null, null);

        if (cursor != null) {
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
                Product product = cursorToProduct(cursor);
                products.add(product);
                cursor.moveToNext();
            }
            cursor.close();
        }
        return products;
    }
    // Cập nhật bảng product
    public int updateProduct(Product product) {
        ContentValues values = new ContentValues();
        values.put(ProductDBHelper.COLUMN_PRODUCT_NAME, product.getName());
        values.put(ProductDBHelper.COLUMN_PRODUCT_QUANTITY, product.getQuantity());

        return database.update(ProductDBHelper.TABLE_PRODUCT, values,
                ProductDBHelper.COLUMN_ID + " = ?",
```

```
                new String[]{String.valueOf(product.getId())});
    }
    // Xóa trong bảng product theo id
    public void deleteProduct(int productId) {
        database.delete(ProductDBHelper.TABLE_PRODUCT,
                ProductDBHelper.COLUMN_ID + " = ?",
                new String[]{String.valueOf(productId)});
    }
    // Chuyển đổi con trỏ thành đối tượng Sản phẩm
    private Product cursorToProduct(Cursor cursor) {
        int idIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_ID);
        int nameIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_PRODUCT_NAME);
        int quantityIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_PRODUCT_QUANTITY);

        int id = cursor.getInt(idIndex);
        String name = cursor.getString(nameIndex);
        int quantity = cursor.getInt(quantityIndex);

        return new Product(id, name, quantity);
    }
}
```

ProductDetailDAO:

```
public class ProductDetailDAO {
    private SQLiteDatabase database;
    private ProductDBHelper dbHelper;

    public ProductDetailDAO(Context context) {
        dbHelper = new ProductDBHelper(context);
    }

    public void open() {
        database = dbHelper.getWritableDatabase();
```

```
    }

    public void close() {
        dbHelper.close();
    }

    // Thêm một chi tiết sản phẩm vào cơ sở dữ liệu
    public long addProductDetail(ProductDetail productDetail) {
        ContentValues values = new ContentValues();
        values.put(ProductDBHelper.COLUMN_PRODUCT_ID, productDetail.getProductId());
        values.put(ProductDBHelper.COLUMN_PRODUCT_DESCRIPTION, productDetail.getDescription());
        values.put(ProductDBHelper.COLUMN_PRODUCT_PRICE, productDetail.getPrice());
        values.put(ProductDBHelper.COLUMN_PRODUCT_COLOR, productDetail.getColor());
        return database.insert(ProductDBHelper.TABLE_PRODUCT_DETAIL, null, values);
    }

    // Cập nhật thông tin chi tiết sản phẩm
    public int updateProductDetail(ProductDetail productDetail) {
        ContentValues values = new ContentValues();
        values.put(ProductDBHelper.COLUMN_PRODUCT_DESCRIPTION, productDetail.getDescription());
        values.put(ProductDBHelper.COLUMN_PRODUCT_PRICE, productDetail.getPrice());
        values.put(ProductDBHelper.COLUMN_PRODUCT_COLOR, productDetail.getColor());
        return database.update(ProductDBHelper.TABLE_PRODUCT_DETAIL, values,
                ProductDBHelper.COLUMN_ID + " = ?",
                new String[]{String.valueOf(productDetail.getId())});
    }

    // Xóa một chi tiết sản phẩm theo ID
    public void deleteProductDetail(int productDetailId) {
        database.delete(ProductDBHelper.TABLE_PRODUCT_DETAIL,
                ProductDBHelper.COLUMN_ID + " = ?",
                new String[]{String.valueOf(productDetailId)});
```

```
    }
    // Lấy chi tiết sản phẩm dựa trên ID
    public ProductDetail getProductDetailByProductId(int productId) {
        Cursor cursor = database.query(ProductDBHelper.TABLE_PRODUCT_DETAIL, null,
                ProductDBHelper.COLUMN_PRODUCT_ID + " = ?",
                new String[]{String.valueOf(productId)}, null, null, null);

        if (cursor != null) {
            cursor.moveToFirst();
            ProductDetail productDetail = cursorToProductDetail(cursor);
            cursor.close();
            return productDetail;
        } else {
            return null;
        }
    }

    // Chuyển đổi dữ liệu từ Cursor thành đối tượng ProductDetail
    private ProductDetail cursorToProductDetail(Cursor cursor) {
        int idIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_ID);
        int productIdIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_PRODUCT_ID);
        int descriptionIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_PRODUCT_DESCRIPTION);
        int priceIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_PRODUCT_PRICE);
        int colorIndex = cursor.getColumnIndex(ProductDBHelper.COLUMN_PRODUCT_COLOR);

        int id = cursor.getInt(idIndex);
        int productId = cursor.getInt(productIdIndex);
        String description = cursor.getString(descriptionIndex);
        double price = cursor.getDouble(priceIndex);
        String color = cursor.getString(colorIndex);

        return new ProductDetail(id, productId, description, price, color);
```

```
    }
}
```

5. Xây dựng Service

Lớp ProductService có tác dụng trung gian giữa các hoạt động quản lý sản phẩm và chi tiết sản phẩm và cung cấp các phương thức để thực hiện các thao tác liên quan đến dữ liệu sản phẩm và chi tiết sản phẩm

```
public class ProductService {
    private ProductDAO productDAO;
    private ProductDetailDAO productDetailDAO;

    public ProductService(Context context) {
        productDAO = new ProductDAO(context);
        productDetailDAO = new ProductDetailDAO(context);
    }

    public void open() {
        productDAO.open();
        productDetailDAO.open();
    }

    public void close() {
        productDAO.close();
        productDetailDAO.close();
    }

    // Thêm một sản phẩm mới kèm theo chi tiết vào cả hai bảng
    public long addProductWithDetail(Product product, ProductDetail productDetail) {
        long productId = productDAO.addProduct(product);
        if (productId != -1) {
            productDetail.setProductId((int) productId);
```

```
            return productDetailDAO.addProductDetail(productDetail);
        }
        return -1;
    }
    // Lấy tất cả sản phẩm từ cả hai bảng
    public List<Product> getAllProducts() {
        List<Product> products = new ArrayList<>();
        List<Product> allProducts = productDAO.getAllProducts();
        for (Product product : allProducts) {
            ProductDetail productDetail = productDetailDAO.getProductDetailByProductId(product.getId());
            if (productDetail != null) {
                product.setProductDetail(productDetail);
            }
            products.add(product);
        }
        return products;
    }
    // Sửa thông tin sản phẩm
    public long updateProductWithDetail(Product product, ProductDetail productDetail) {
        long rowsAffected = productDAO.updateProduct(product);
        if (rowsAffected > 0) {
            productDetail.setProductId(product.getId()); // Cập nhật productId cho productDetail
            return productDetailDAO.updateProductDetail(productDetail);
        }
        return -1;
    }

    // Xóa sản phẩm
    public void deleteProduct(int productId) {
        // Trước khi xóa sản phẩm, bạn có thể kiểm tra và xóa các chi tiết sản phẩm liên quan
        // ở bảng product_details
        productDetailDAO.deleteProductDetail(productId);
```

```
        // Sau đó, gọi phương thức deleteProduct từ ProductDAO để xóa sản phẩm
        productDAO.deleteProduct(productId);
    }
}
```

5. Xây dựng các Adapter:

ProductAdapter:

```
public class ProductAdapter extends ArrayAdapter<Product> {
    private Context context;
    private List<Product> products;

    public ProductAdapter(Context context, List<Product> products) {
        super(context, 0, products);
        this.context = context;
        this.products = products;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(android.R.layout.simple_list_item_2, parent, false);
            //simple_list_item_2 là layout sẵn có, chỉ cần gọi ra để sử dụng
        }

        Product currentProduct = products.get(position);

        TextView productName = convertView.findViewById(android.R.id.text1);
        productName.setText(currentProduct.getName());
```

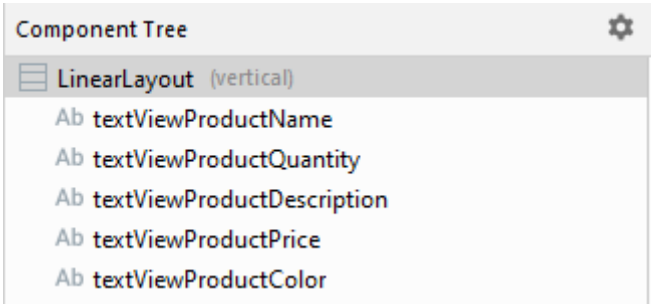
```
        TextView productQuantity = convertView.findViewById(android.R.id.text2);
        productQuantity.setText("Quantity: " + currentProduct.getQuantity());

        return convertView;
    }
}
```

ProductDetailAdapter:

```
public class ProductDetailAdapter extends ArrayAdapter<ProductDetail> {
    private Context context;
    private List<ProductDetail> productDetails;

    public ProductDetailAdapter(Context context, List<ProductDetail> productDetails) {
        super(context, 0, productDetails);
        this.context = context;
        this.productDetails = productDetails;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(android.R.layout.simple_list_item_2, parent, false);
        }

        ProductDetail currentDetail = productDetails.get(position);

        TextView description = convertView.findViewById(android.R.id.text1);
        description.setText(currentDetail.getDescription());

        TextView priceAndColor = convertView.findViewById(android.R.id.text2);
        priceAndColor.setText("Price: $" + currentDetail.getPrice() + " | Color: " + currentDetail.getColor());
```

```
        return convertView;
    }
}
```

6. Xây dựng layout chi tiết sản phẩm dialog_product_detail.xml:



7. Xây dựng MainActivity:

```
public class MainActivity extends AppCompatActivity {
    private EditText editTextProductName, editTextProductQuantity, editTextProductDescription,
            editTextProductPrice, editTextProductColor;
    private Button buttonAddProduct;
    private ListView listViewProducts;
    private ProductDAO productDAO;
    private ProductAdapter productAdapter;
    private List<Product> productList;
    private ProductService productService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Khởi tạo views
```

```
        editTextProductName = findViewById(R.id.editTextProductName);
        editTextProductQuantity = findViewById(R.id.editTextProductQuantity);
        editTextProductDescription = findViewById(R.id.editTextProductDescription);
        editTextProductPrice = findViewById(R.id.editTextProductPrice);
        editTextProductColor = findViewById(R.id.editTextProductColor);
        buttonAddProduct = findViewById(R.id.buttonAddProduct);
        listViewProducts = findViewById(R.id.listViewProducts);

        // Khởi tạo ProductService và ListView
        productService = new ProductService(this);
        productService.open();
        productList = new ArrayList<>();
        productAdapter = new ProductAdapter(this, productList);
        listViewProducts.setAdapter(productAdapter);

        // Đăng ký Context Menu cho ListView
        registerForContextMenu(listViewProducts);

        // Nút thêm sản phẩm
        buttonAddProduct.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addProduct();
            }
        });
        // Click vào item trên ListView để hiển thị product detail
        listViewProducts.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Product product = productList.get(position);
                showProductDetailDialog(product);
            }
```

```
        });

        // Tải sản phẩm từ cơ sở dữ liệu và hiển thị trong ListView
        loadProducts();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        productDAO.close();
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        if (v.getId() == R.id.listViewProducts) {
            AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) menuInfo;
            menu.setHeaderTitle("Tùy chọn");
            menu.add(Menu.NONE, 1, Menu.NONE, "Sửa");
            menu.add(Menu.NONE, 2, Menu.NONE, "Xóa");
        }
    }

    @Override
    public boolean onContextItemSelected(MenuItem item) {
        AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
        int position = info.position;
        switch (item.getItemId()) {
            case 1:
                // Xử lý khi người dùng chọn Edit
                editProduct(productList.get(position));
                return true;
            case 2:
```

```
                // Xử lý khi người dùng chọn Delete
                deleteProduct(productList.get(position));
                return true;
            default:
                return super.onContextItemSelected(item);
        }
    }
    // Phương thức thêm sản phẩm
    private void addProduct() {
        // Nhận thông tin chi tiết về sản phẩm và sản phẩm từ các trường EditText
        String name = editTextProductName.getText().toString().trim();
        String quantityStr = editTextProductQuantity.getText().toString().trim();
        String description = editTextProductDescription.getText().toString().trim();
        String priceStr = editTextProductPrice.getText().toString().trim();
        String color = editTextProductColor.getText().toString().trim();

        if (name.isEmpty() || quantityStr.isEmpty() || description.isEmpty()
                || priceStr.isEmpty() || color.isEmpty()) {
            Toast.makeText(this, "Hãy điền đầy đủ thông tin", Toast.LENGTH_SHORT).show();
            return;
        }
        int quantity = Integer.parseInt(quantityStr);
        double price = Double.parseDouble(priceStr);

        // Tạo các đối tượng Product và ProductDetail mới
        Product product = new Product(0, name, quantity);
        ProductDetail productDetail = new ProductDetail(0, 0, description, price, color);

        // Thêm sản phẩm và chi tiết sản phẩm vào cơ sở dữ liệu
        long result = productService.addProductWithDetail(product, productDetail);

        if (result != -1) {
```

```
            // Tải lại sản phẩm từ cơ sở dữ liệu và cập nhật ListView
            loadProducts();

            // Clear input fields
            clearFields();

            Toast.makeText(this, "Thêm sản phẩm thành công", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Thêm sản phẩm thất bại", Toast.LENGTH_SHORT).show();
        }
    }

    // Phương thức tải sản phẩm từ cơ sở dữ liệu và cập nhật ListView
    private void loadProducts() {
        productList.clear();
        productList.addAll(productService.getAllProducts());
        productAdapter.notifyDataSetChanged();
    }

    // Xóa các trường dữ liệu
    private void clearFields() {
        editTextProductName.setText("");
        editTextProductQuantity.setText("");
        editTextProductDescription.setText("");
        editTextProductPrice.setText("");
        editTextProductColor.setText("");
    }
    // Hiển thị chi tiết sản phẩn lên dialog
    private void showProductDetailDialog(Product product) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Chi tiết sản phẩm");
```

```
        View dialogView = LayoutInflater.from(this).inflate(R.layout.dialog_product_detail, null);
        builder.setView(dialogView);

        TextView textViewProductName = dialogView.findViewById(R.id.textViewProductName);
        TextView textViewProductQuantity = dialogView.findViewById(R.id.textViewProductQuantity);
        TextView textViewProductDescription = dialogView.findViewById(R.id.textViewProductDescription);
        TextView textViewProductPrice = dialogView.findViewById(R.id.textViewProductPrice);
        TextView textViewProductColor = dialogView.findViewById(R.id.textViewProductColor);

        textViewProductName.setText("Tên sản phẩm: " + product.getName());
        textViewProductQuantity.setText("Số lượng: " + String.valueOf(product.getQuantity()));
        if (product.getProductDetail() != null) {
            textViewProductDescription.setText("Mô tả: " + product.getProductDetail().getDescription());
            textViewProductPrice.setText("Giá: " + String.valueOf(product.getProductDetail().getPrice()));
            textViewProductColor.setText("Màu sắc: " + product.getProductDetail().getColor());
        }

        builder.setPositiveButton("Đóng", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });

        AlertDialog dialog = builder.create();
        dialog.show();
    }
    private void editProduct(Product product) {
        // Xử lý khi người dùng chọn Edit, có thể mở một Activity để chỉnh sửa sản phẩm
        Toast.makeText(this, "Bạn vừa chọn Sửa", Toast.LENGTH_SHORT).show();
    }
```

```
    private void deleteProduct(Product product) {
        // Xử lý khi người dùng chọn Delete, có thể xóa sản phẩm từ cơ sở dữ liệu và cập nhật lại ListView
        Toast.makeText(this, "Bạn vừa chọn xóa", Toast.LENGTH_SHORT).show();
    }
}
```

Trong ví dụ chưa xử lý hoàn chỉnh các chức năng sửa và xóa, các bạn có thể làm tương tự như bài học trước.