Banking App

Reverse Engineering

Code Quality

Code Quality:

Based on the idea that the code is as close as possible to a type of literary literature. And the code over time needs maintenance, reuse, and so on It can be said that the code is read more than it is written, so I say that the quality specifications of the code are to be readable (clear).

The code (the code itself, I don't mean the comments) should be clear.

Also, the code must be economical so that it does not consume a lot of system resources (RAM, memory, CPU, etc.)

Why Code Quality Matters ?

It is important to pay attention to the code quality, since it impacts the overall quality of your codebase. And, quality impacts how safe, secure, and reliable your codebase is.

Aspects of code quality that should be measured:

Here are some of the key traits to measure for higher quality. The source code of (Alinma bank) have medium quality cause it is apply some of four measure for higher quality .

Reliability:

Maintainability:

In my opinion our code source doesn’t support principle Maintainability because Our code doesn't support method (Builder Pattern).

This picture is an external example of how to make (Builder Pattern) show in figure1 .

Figure1 This picture See how the object is configured show in figure2:

```java
// Pizza Construction using Builder Pattern

public class PizzaBuilderPattern {
   public static void main(String[] args) {
      Pizza pizza = new Pizza.Builder(10).cheese(true).peppernoi(true).build();
   }
}

class Pizza {
   // required
   private int size;

   // optionals
   private boolean cheese;
   private boolean hotdog;
   private boolean peppernoi;

   public static class Builder {
      private int size;

      private boolean cheese;
      private boolean hotdog;
      private boolean peppernoi;

      public Builder(int size) { this.size = size; }

      public Builder cheese (boolean cheese) { this.cheese = cheese; return this; }
      public Builder hotdog (boolean hotdog) { this.hotdog = hotdog; return this; }
      public Builder peppernoi (boolean pep) { this.peppernoi = pep; return this; }

      public Pizza build () { return new Pizza(this); }
   }

   private Pizza (Builder builder) {
      this.size = builder.size;
      this.cheese = builder.cheese;
      this.hotdog = builder.hotdog;
      this.peppernoi = builder.peppernoi;
   }
}
```

Figure1

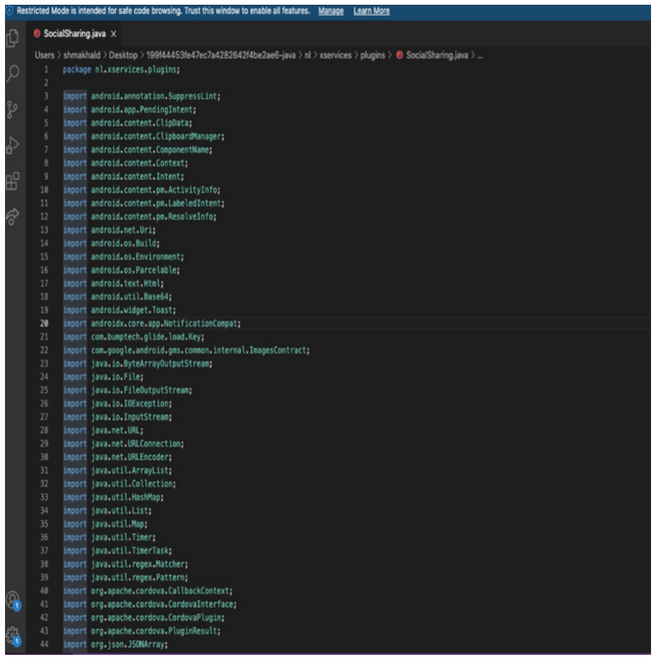
```java
Pizza pizza = new Pizza.Builder(10).cheese(true).peppernoi(true).build();
```

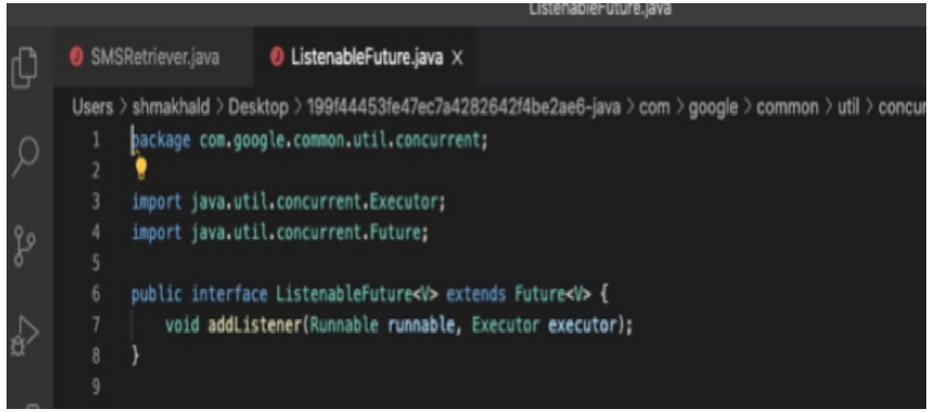
Figure2

Portability

Reusability:

In my opinion our code source support principle Reusability There are many relationships between classes Extended from each other and have a lot of subclasses Code that does not reinvent the wheel and uses the available libraries as much as possible import a lot of other libraries class from different environment Example: Import a lot of libraries





Some step to make quality code high :

-Use a Coding Standard.
-Analyze Code –Before Code Reviews.
-Follow Code Review Best Practices
-In order to verify the intent of the code.
-Refactor Legacy Code (When Necessary).

CleanCode

Clean code is a set of rules and principles that helps to keep our code readable, maintainable, and extendable. It's one of the most important aspects of writing quality software. We (developers) spend way more time reading the code than actually writing it, which is why it's important that we write good code.

In this workshop I analyzed our bank code in 3 criteria which are : DYR “Don’t Repeat Yourself” .

```
Object newInstance = MediaDescriptionCompatApi21.Builder.newInstance();
MediaDescriptionCompatApi21.Builder.setMediaId(newInstance, this.mMediaId);
MediaDescriptionCompatApi21.Builder.setTitle(newInstance, this.mTitle);
MediaDescriptionCompatApi21.Builder.setSubtitle(newInstance, this.mSubtitle);
MediaDescriptionCompatApi21.Builder.setDescription(newInstance, this.mDescription);
MediaDescriptionCompatApi21.Builder.setIconBitmap(newInstance, this.mIcon);
MediaDescriptionCompatApi21.Builder.setIconUri(newInstance, this.mIconUri);
Bundle bundle = this.mExtras;
```

Boy Scot Rule .

```
                }
                imageButton.setContentDescription("Close Button");
                imageButton.setId(Integer.valueOf(i).intValue());
                imageButton.setOnClickListener(new View.OnClickListener() {
```

SRP “Single Responsibility Principal” .

```
public boolean isSecretEstablished() {
    return this.expectedBridgeSecret != -1;
}
```

And In some group of codes they were following these principles .

Code Structure

mess build in the system was the variables style and the vertical alignment

Variable Style:

- Underscor:

```
public static final String APPLICATION_ID = "android.support.constraint";
public static final String BUILD_TYPE = "release";
public static final boolean DEBUG = false;
public static final String FLAVOR = "";
public static final int VERSION_CODE = -1;
public static final String VERSION_NAME = "";
```

- CamelCase:

```
public static final int cardBackgroundColor = 2130968667;
public static final int cardCornerRadius = 2130968668;
public static final int cardElevation = 2130968669;
public static final int cardMaxElevation = 2130968670;
```

vertical alignment:

```
public static final int constraint_referenced_ids = 2130968727
public static final int content = 2130968728;
public static final int emptyVisibility = 2130968768;
public static final int layout_constrainedHeight = 2130968875;
public static final int layout_constrainedWidth = 2130968876;
```

```
super(context, cursor);
this.mDropDownLayout = i;
this.mLayout = i;
this.mInflater = (LayoutInflater) context.getSystemService("layout_inflater");
```

Code Documentation :

```
/* access modifiers changed from: protected */
public Filter.FilterResults performFiltering(CharSequence charSequence) {
    Cursor runQueryOnBackgroundThread = this.mClient.runQueryOnBackgroundThread(charSequence);
    Filter.FilterResults filterResults = new Filter.FilterResults();
    if (runQueryOnBackgroundThread != null) {
        filterResults.count = runQueryOnBackgroundThread.getCount();
        filterResults.values = runQueryOnBackgroundThread;
    } else {
        filterResults.count = 0;
        filterResults.values = null;
    }
    return filterResults;
}
```

```
@Override // androidx.cursoradapter.widget.CursorAdapter
public Cursor swapCursor(Cursor cursor) {
    findColumns(cursor, this.mOriginalFrom);
    return super.swapCursor(cursor);
}
```

Security Issues

Why security is important?

-secure Sodtware development will consistently generate the following benefits for the company:

• Lower costs, to early detection and elimination of security flaws
• Stakeholders will embrace the importance of investment in secure methodologies, and will not push developers to release software more quickly

Issues we found in the project and how to avoid it:

• External storage perrmissions: it can access the device storage at any time. means it can upload personal files or even delete sensitive information from the device, so it’s better to think twice before giving storage permission, as it can be harmful.

• The use of dangerous and special permissions should be avoided when not strictly required, and an app should work even when a permission is not granted.

• Allowing data backup: Allowing app data to be backed up could be useful for the software developers but somehow it may lead to app hacking

• If your app holds sensitive data, you can stop allowing users from making a backup of the app. This can be done by placing more security files and methods

Good security feature in our project:

- Mobile man-in-the-middle attack prevention:

There are a lot of prevention techniques such as encryption for the sensitive information

Design

Design patterns are largely about communication and expressiveness. By using the standard pattern names, the names of the classes that implement those patterns, you can succinctly describe your design to other developers.

In the system they use two type of design patterns. Adapter pattern and observe pattern.

Team Member

Al-juhara Al-sadoun

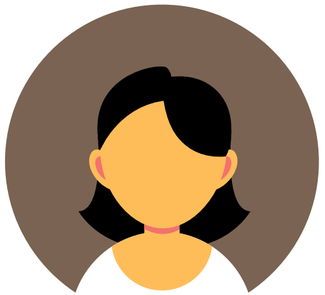
440023448

Alya Al-shaikh

440022910

Layan Al-jabali

440022562



Shamaa Elewa

440022232

Thanks For Listening